ModelSim®

Actel

Tutorial

Version 5.8d

Published: 10/Mar/04

The world's most popular HDL simulator

ModelSim support

Support for ModelSim is available from your FPGA vendor. See the About ModelSim dialog box (accessed via the Help menu) for contact information.

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.

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

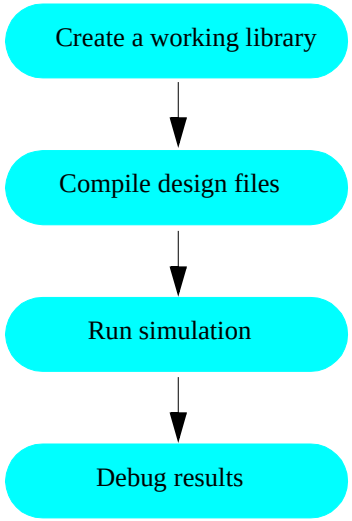
ModelSim is a simulation and debugging tool for VHDL, Verilog, and mixed-language designs.

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 a combination of the two, are compiled into a library. You typically start a new simulation in ModelSim by creating a working library called "work". "Work" is the library name used by the compiler as the default destination for compiled design units.

Compiling your design

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

Running the simulation

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

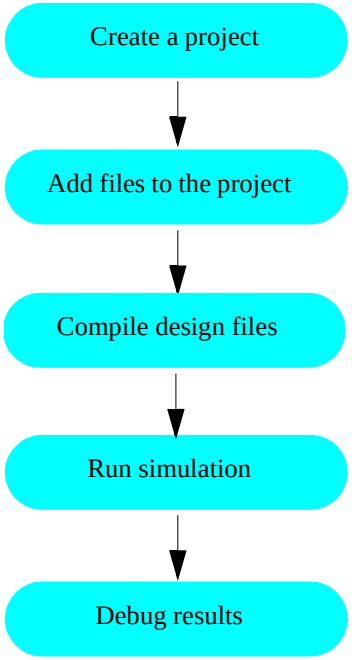
Debugging your results

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

Project flow

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

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



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

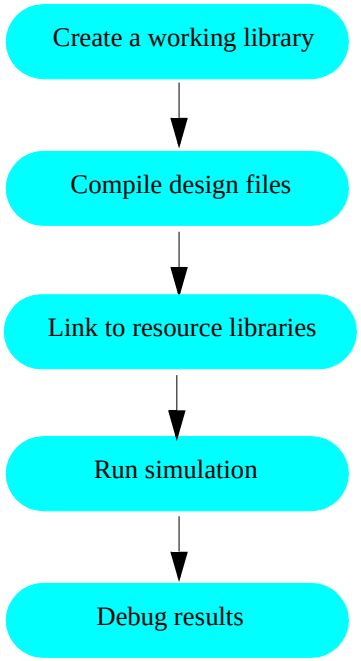
- You do not have to create a working library in the project flow; it is done for you automatically.
- Projects are persistent. In other words, they will open every time you invoke ModelSim unless you specifically close them.

Multiple library flow

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

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

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



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

Debugging tools

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

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

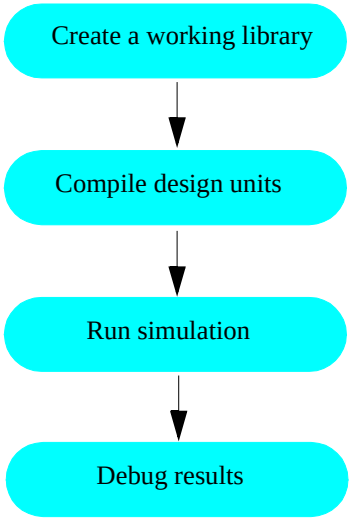
Lesson 2 - Basic simulation

Topics

The following topics are covered in this lesson:

Introduction

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-43), *Chapter 5 - Verilog simulation* (UM-83), *Chapter 4 - VHDL simulation* (UM-57)

ModelSim Command Reference (**vlib** (CR-203), **vmap** (CR-212), **vlog** (CR-204), **vcom** (CR-167), **vsim** (CR-213), **view** (CR-179), and **run** (CR-133) 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). Create the directory, then copy *counter.v* and *tcounter.v* files from */<install_dir>/examples* to the new directory.

2 Start ModelSim if necessary.

a 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 Dialog

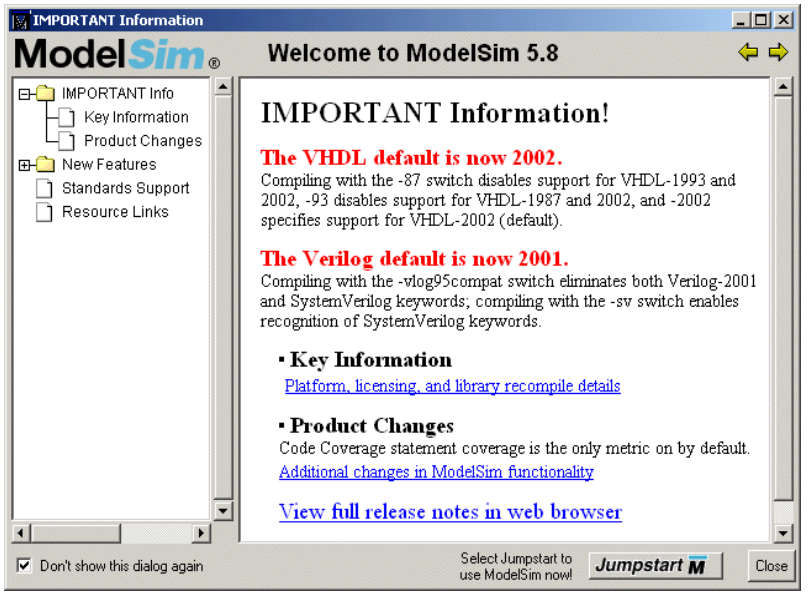
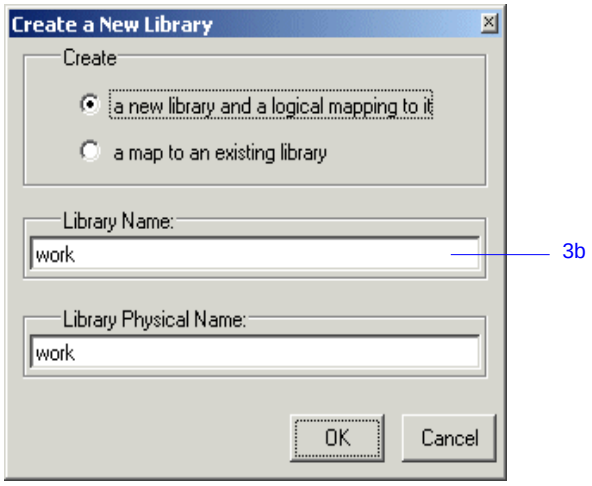


Figure 2: The Create a New Library dialog



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.

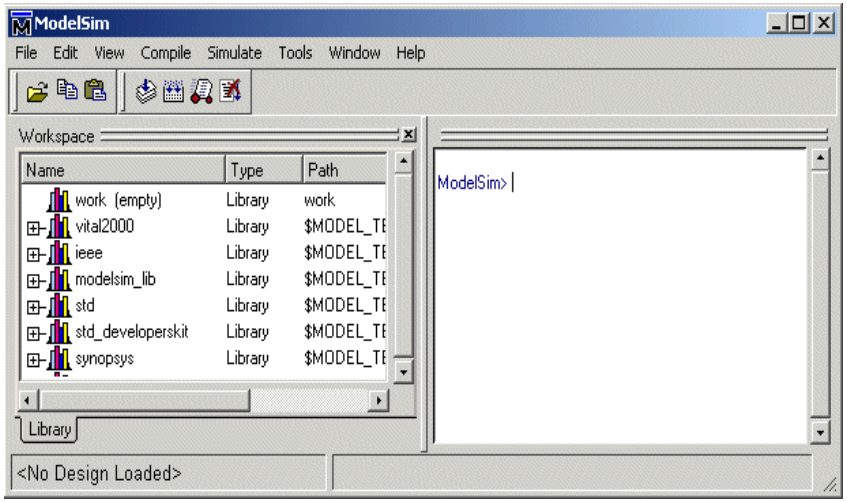
ModelSim also adds the library to the list in the Workspace (Figure 3) and records the library mapping for future reference in the ModelSim initialization file (*modelsim.ini*).

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.

1 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 > Project**.

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**.

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). If you scroll to the right, you will see their types (modules in this case) and the path to the underlying source files.

Figure 4: The Compile HDL Source Files dialog

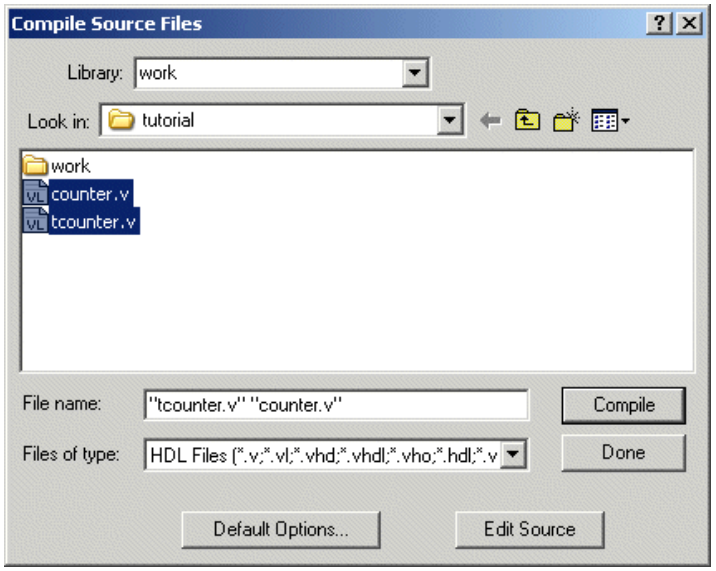
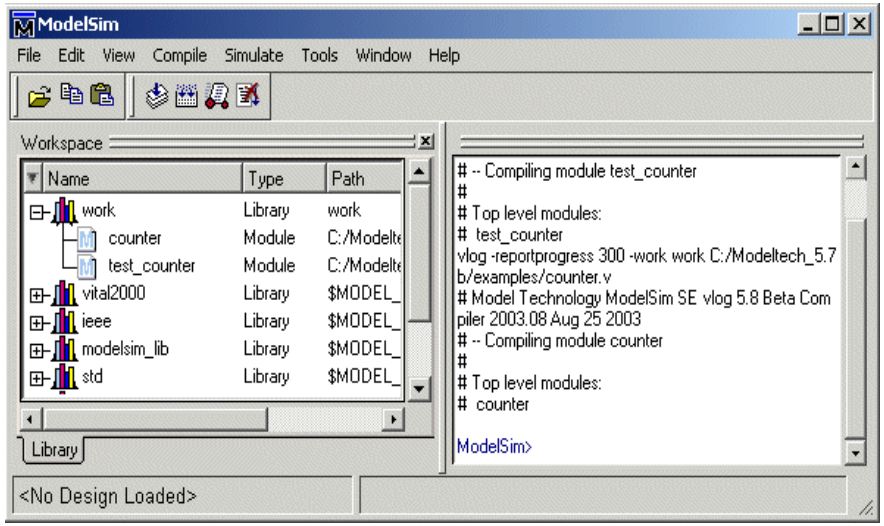


Figure 5: Verilog modules compiled into the work library

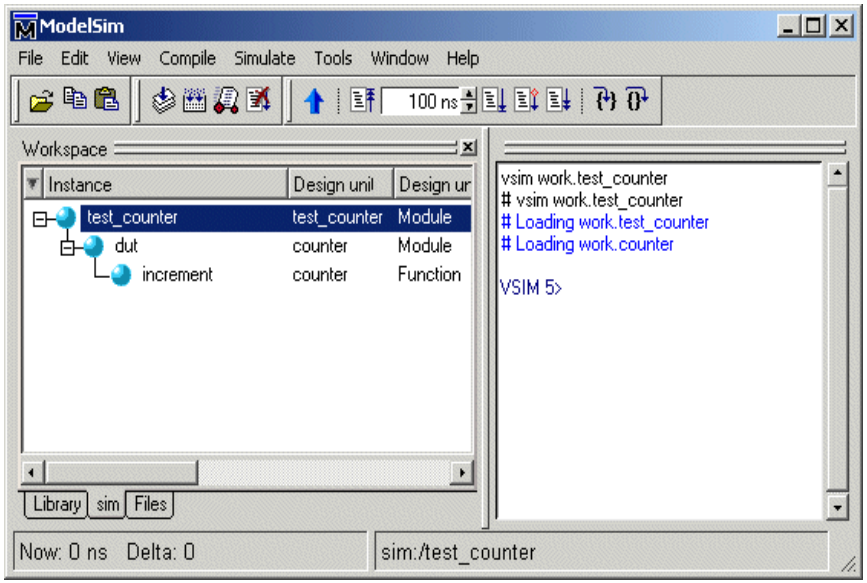


3 Load the *test_counter* module.

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

You will see a new tab named *sim* that displays the hierarchical structure of the design (Figure 6). 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: Workspace tab showing a Verilog design



Running the simulation

Now you will run the simulation.

1 View all windows.

a Select **View > All Windows**.

This opens all ModelSim windows, giving you different views of your design data and a variety of debugging tools. You may need to move or resize the windows to your liking.

2 Add signals to the Wave window.

a In the Signals window, select **Add > Wave > Signals in Region** (Figure 7).

Three signals are added to the Wave window.

3 Run the simulation.

a Click the Run icon on the Main, Wave, or Source window toolbar.

The simulation runs for 100 ns (the default simulation length) and waves are drawn in the Wave window (Figure 8).

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 7: Adding signals to the Wave window

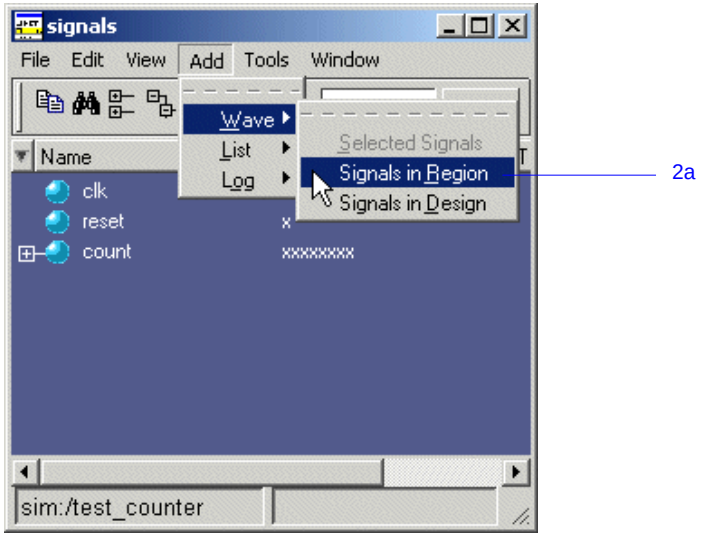
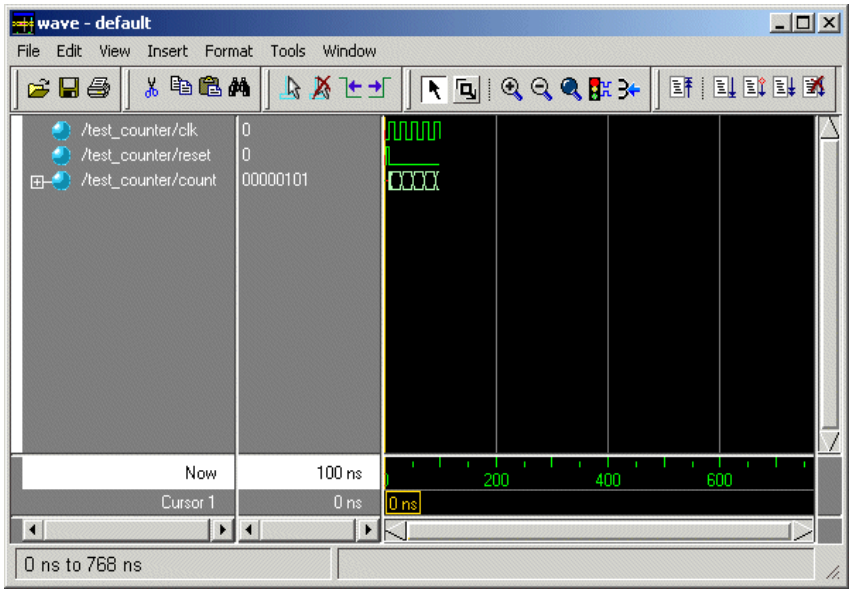


Figure 8: Waves being drawn in the Wave window



c Click the Run -All icon on the Main, Wave, or Source 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 blue line numbers.

1 Set a breakpoint on line 28 of *counter.v* (if you are simulating the VHDL files, use line 30 instead).

   a Select *dut* in the *sim* tab of the Main window Workspace.

   This opens *counter.v* in the Source window.

   b Scroll to line 28 and click on or to the left of the line number.

   A red diamond appears next to the line (Figure 9).

2 Disable, enable, and delete the breakpoint.

   a Click the red diamond to disable the breakpoint.

   b Click the red diamond again to re-enable the breakpoint.

   c Click the red diamond with your right mouse button and select **Remove Breakpoint 28**.

   d Click on line 28 again to re-create the breakpoint.

3 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 10).

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

Figure 9: A breakpoint in the Source window

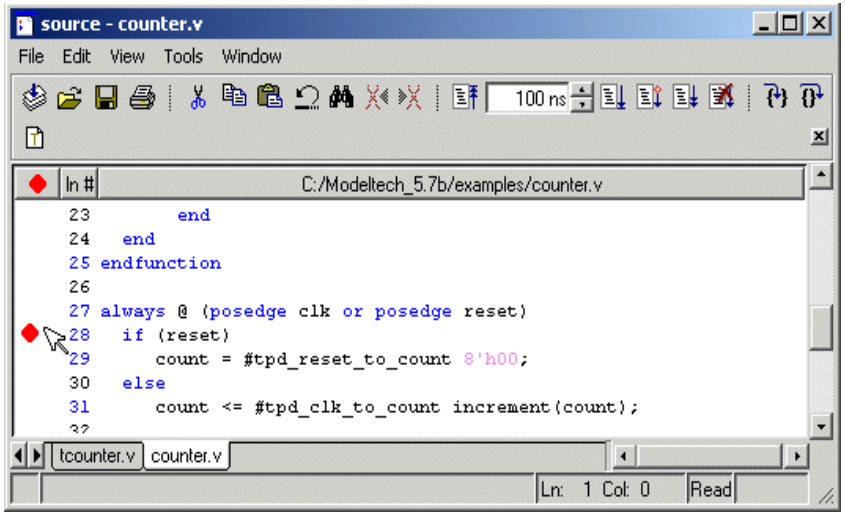
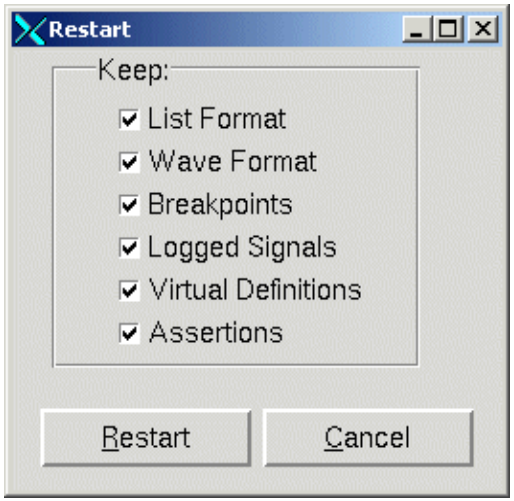


Figure 10: The Restart dialog



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

The simulation runs until the breakpoint is hit. When the simulation hits the breakpoint, it stops running, highlights the line with an arrow in the Source window (Figure 11), and issues a Break message in the Main window.

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 Signals window
- set your mouse pointer over the *count* variable in the Source window, and a "balloon" will pop up with the value (Figure 11)
- 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`)

4 Try out the step commands.

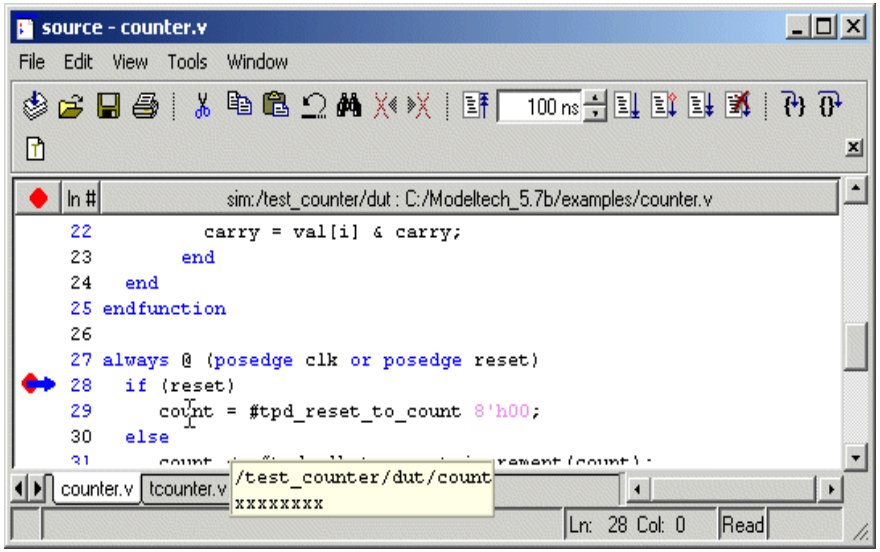
a Click the Step icon on the Source window toolbar.

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.

b When you are done experimenting, close the Source window by selecting **File > Close**.

Figure 11: Resting the mouse pointer on a variable in the Source 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-21)

Creating a new project

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

   a 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 12). The default library is where compiled design units will reside.

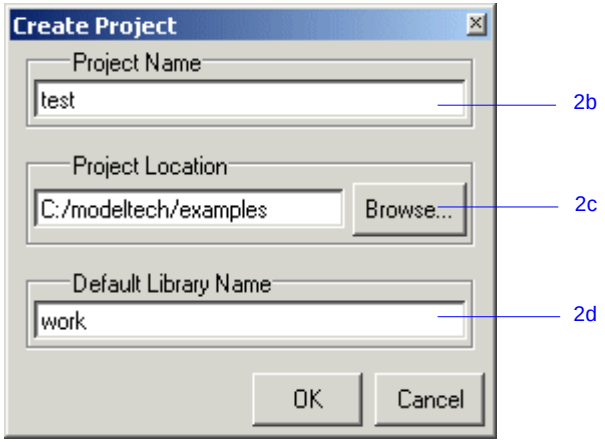
   b Type **test** in the Project Name field.

   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**.

Figure 12: The Create Project dialog



Adding items 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 13). 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 14). 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 Select *counter.v*, hold the <Ctrl> key down, and then select *tcounter.v*.

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

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

Figure 13: Adding new items to a project

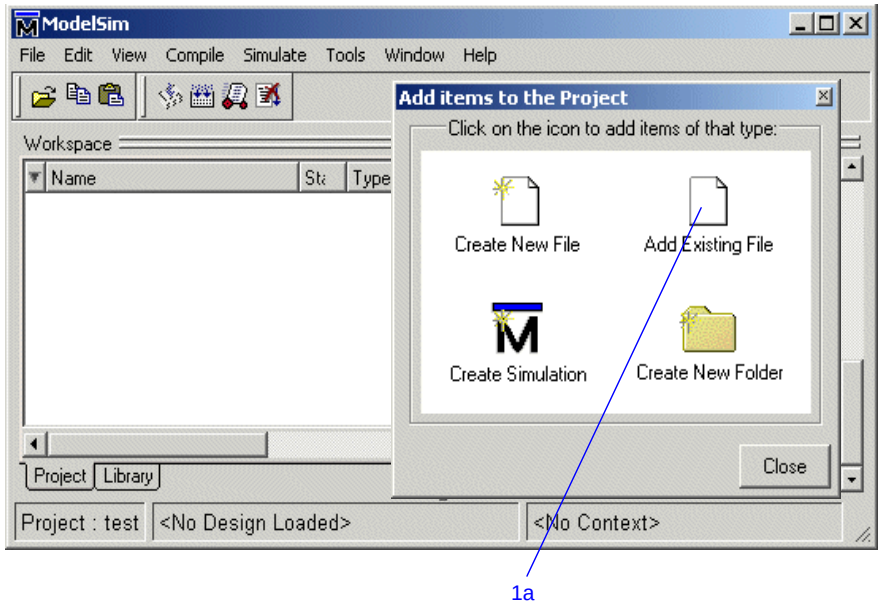
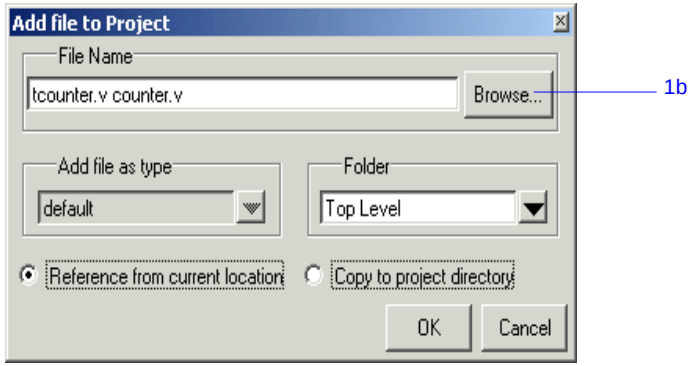


Figure 14: The Add File to Project dialog



You should now see two files listed in the Project tab of the Main window workspace (Figure 15).

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 16).

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 15: Newly added project files display a '?' for status

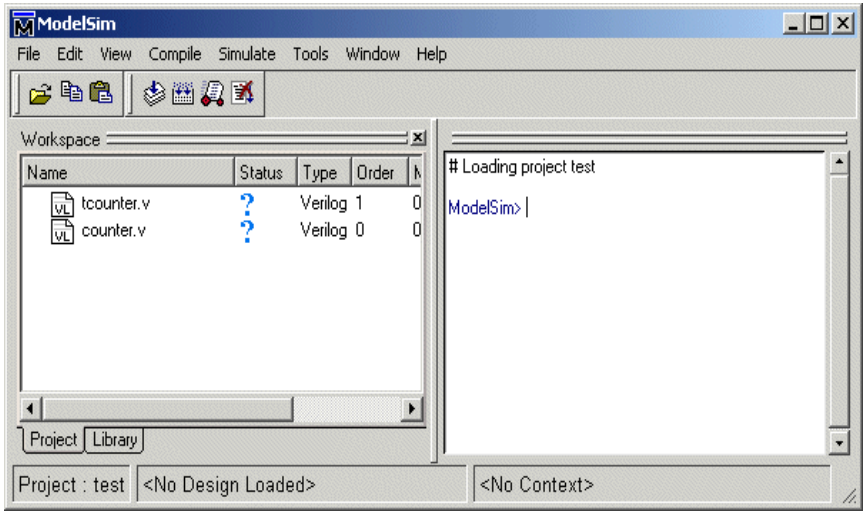
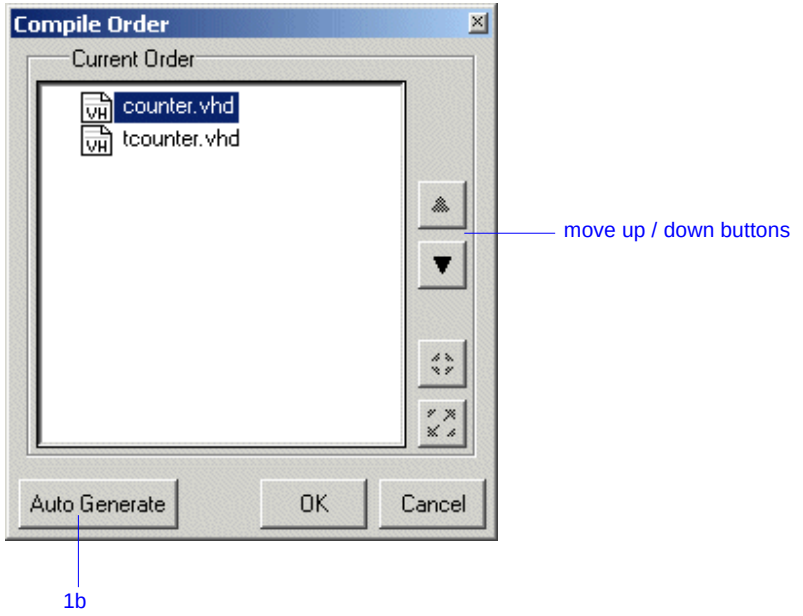


Figure 16: The Compile Order dialog box



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 window on the right.

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 17).

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 18). 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 17: The Library tab with an expanded library

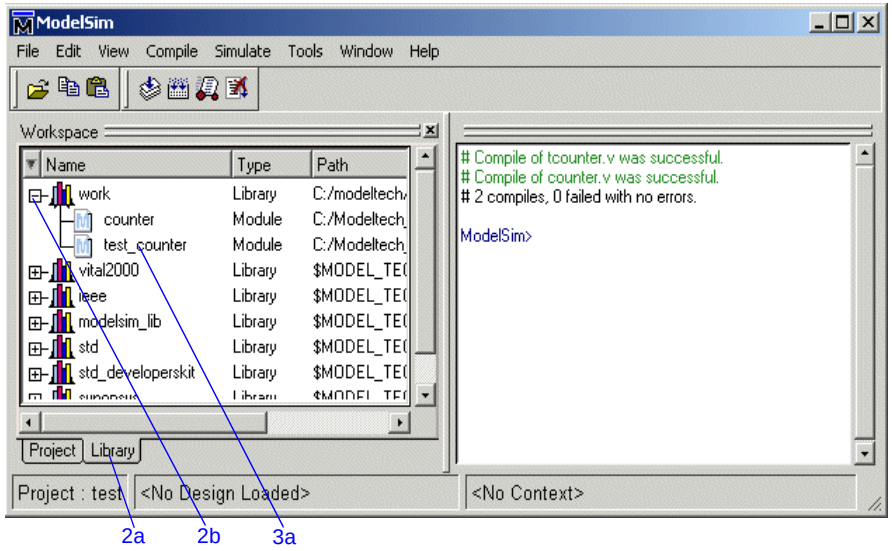
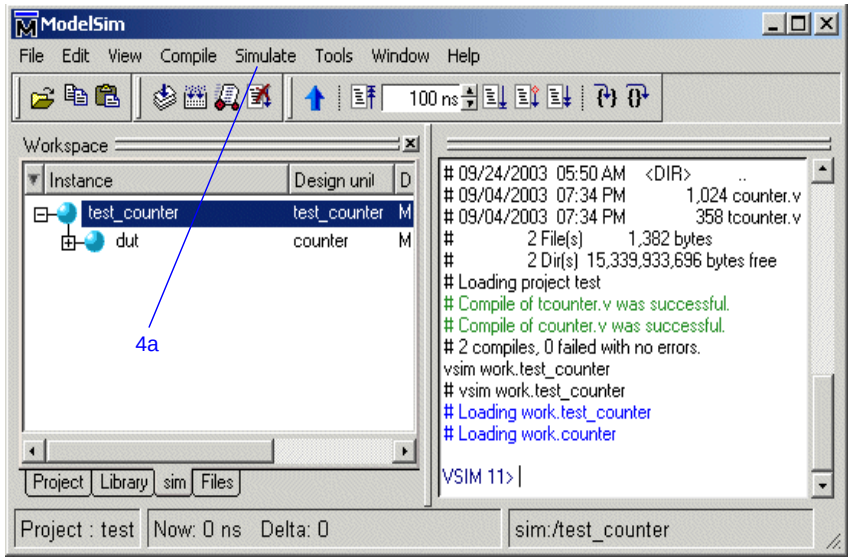


Figure 18: The structure tab for the *counter* design unit



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 drop-down in Figure 14). 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 13, 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 19).

c Click **OK**.

You'll now see a folder in the Project tab (Figure 20).

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 21).

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

d Click OK.

Figure 19: Adding a new folder to the project

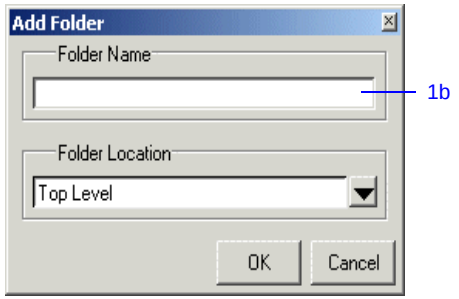


Figure 20: A folder in a project

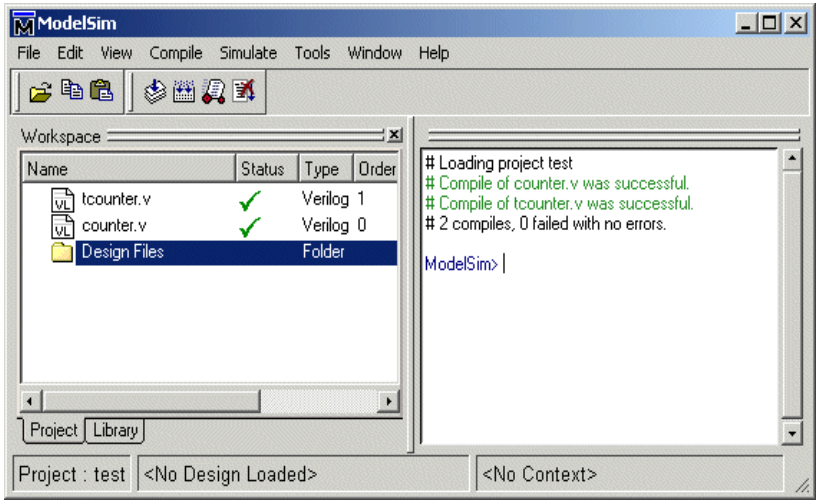
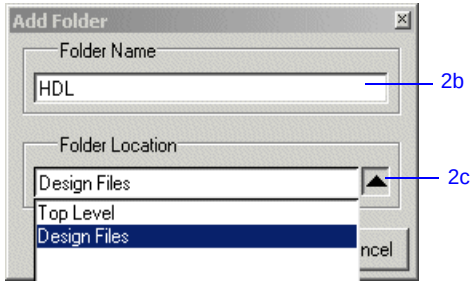


Figure 21: Creating a subfolder



You'll now see a '+' icon next to the *Design Files* folder in the Project tab (Figure 22).

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 23), 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 22: A folder with a sub-folder

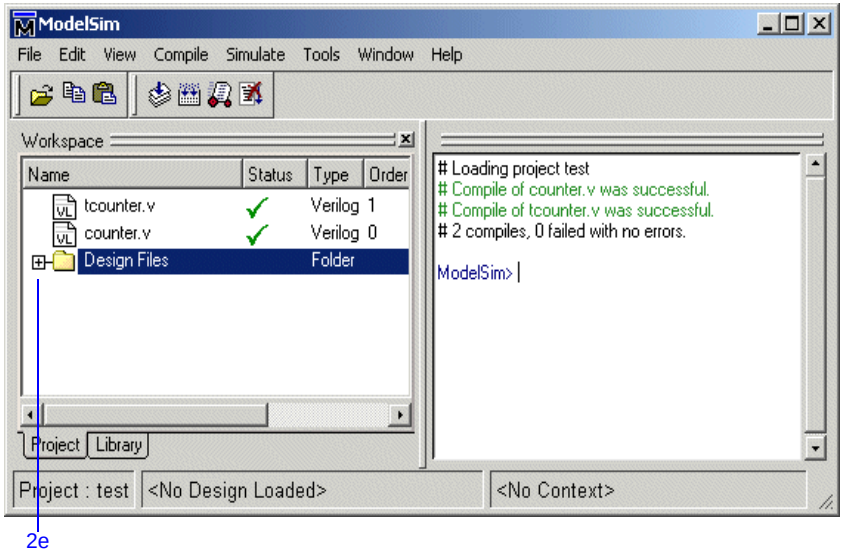
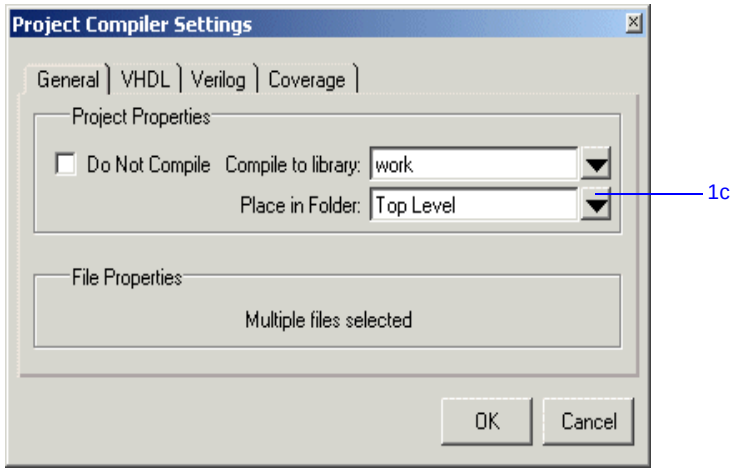


Figure 23: Changing file location via the project settings dialog



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 24). 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 25).

Figure 24: The Simulation Configuration dialog

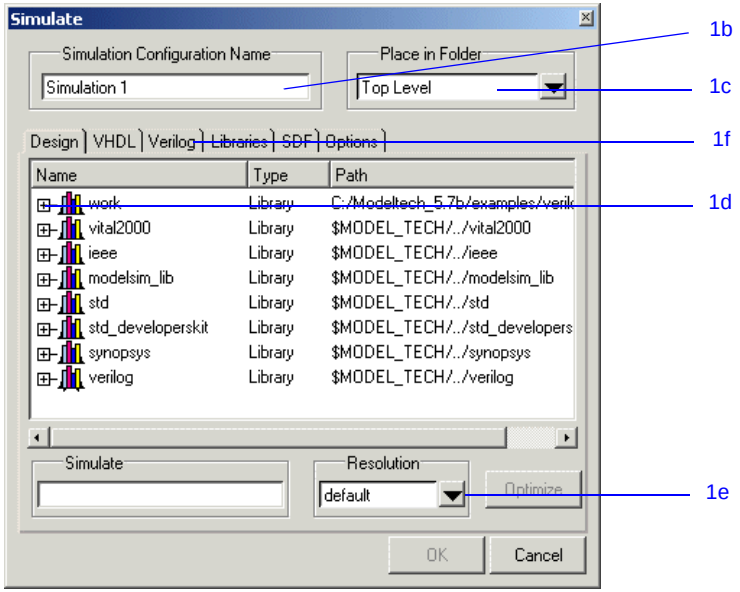
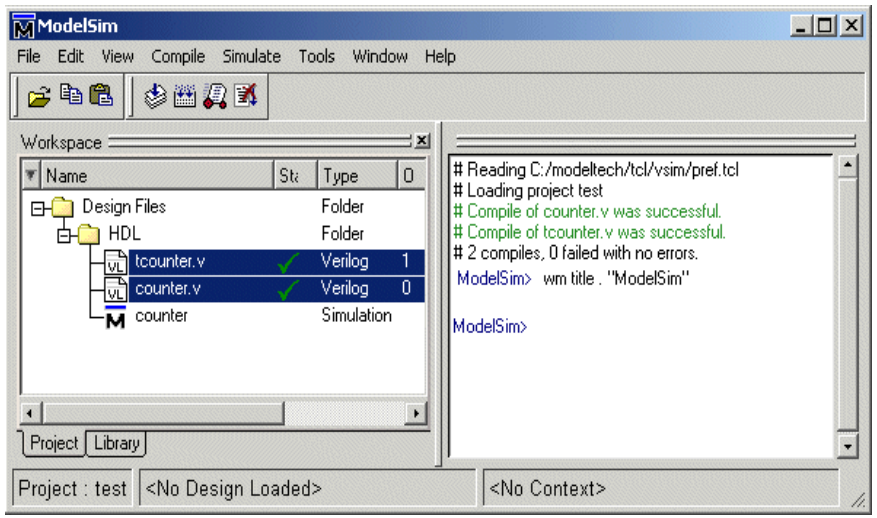


Figure 25: A Simulation Configuration in the Project tab

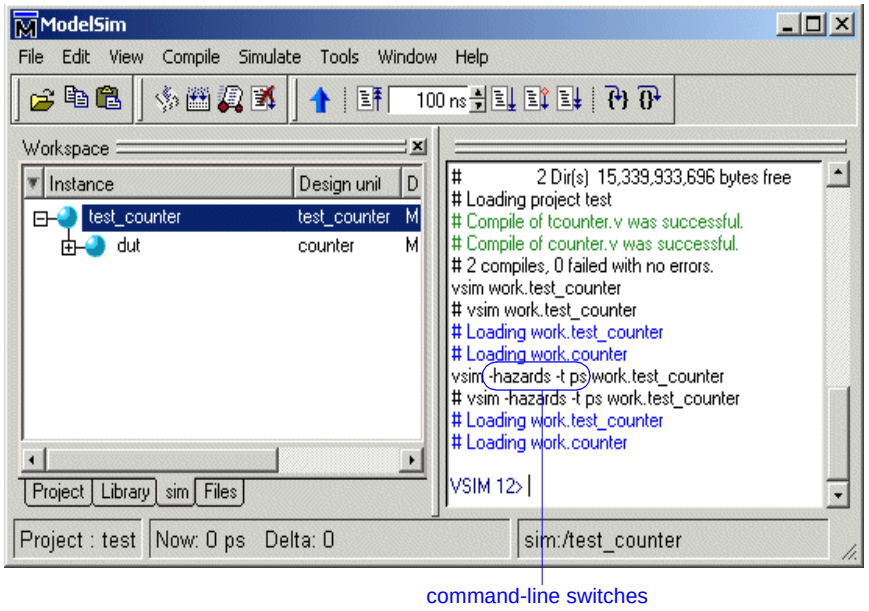


2 Load the Simulation Configuration.

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 26). These are the command-line equivalents of the options you specified in the Simulate dialog.

Figure 26: Transcript shows options used for Simulation Configuration



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 **File > Close > Project**. Click OK.

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

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-43)

Creating the resource library

1 Create a directory for the resource library.

Create the directory that will hold the resource library. Copy *counter.v* from *<install_dir>/modeltech/examples* to the new directory.

2 Create a directory for the testbench.

Create a new the directory 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 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.

4 Create the resource library.

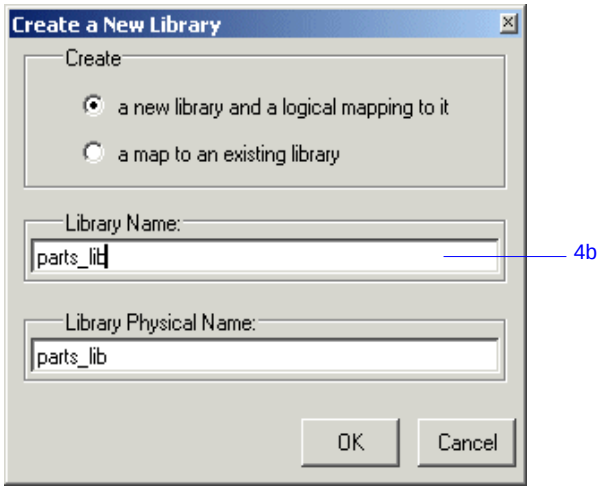
a Select **File > New > Library**.

b Type **parts_lib** in the Library Name field (Figure 27).

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 27: 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 28).

c Double-click *counter.v*.

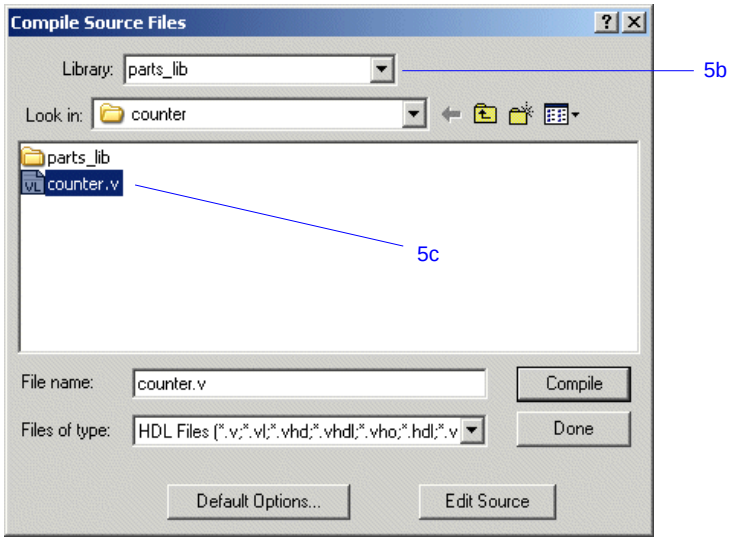
d Click Done.

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

6 Change to the directory you created in step 2.

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

Figure 28: 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 29). 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 30). 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 29: Verilog simulation error reported in the Main window

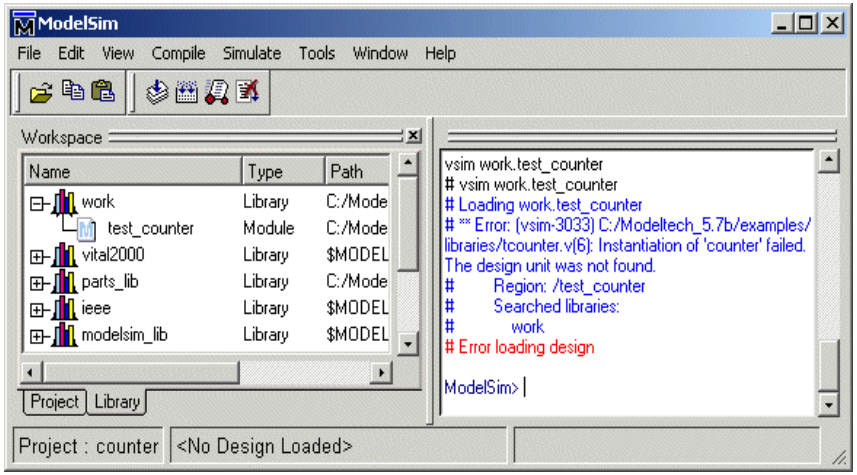
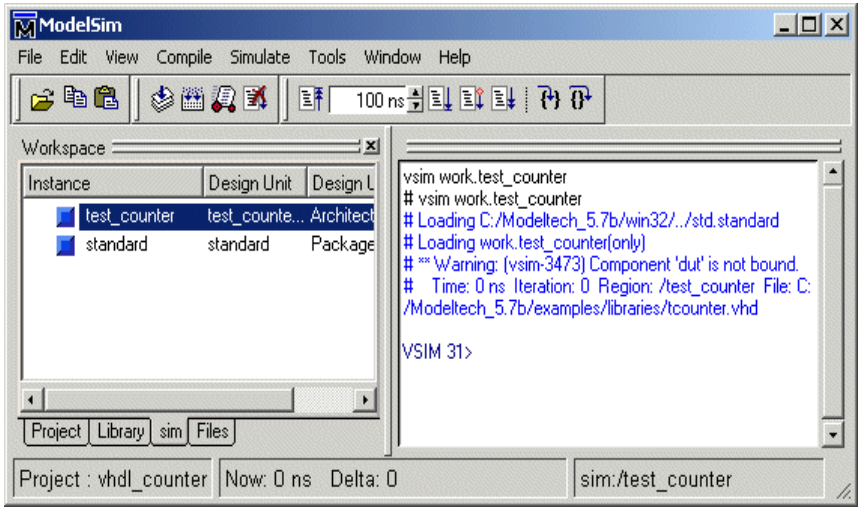


Figure 30: VHDL simulation warning reported in Main window



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.

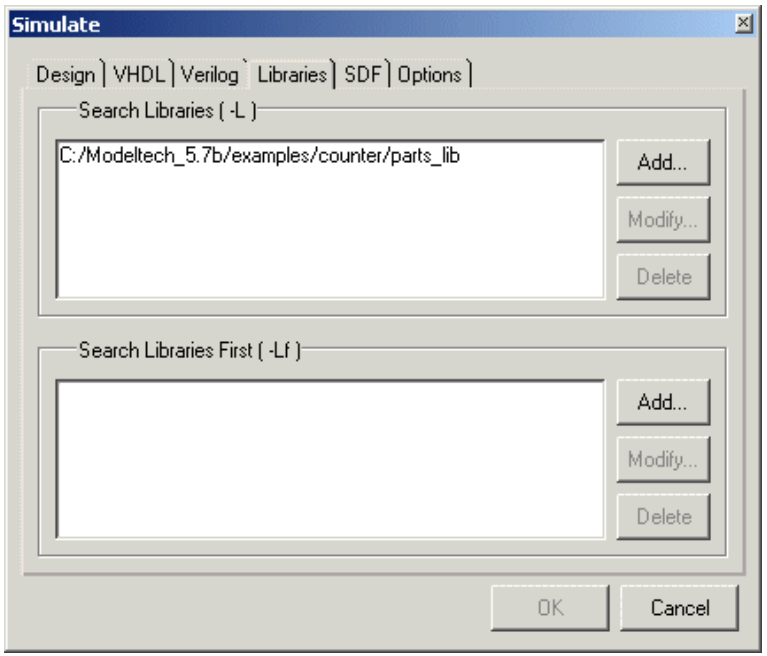
e Click Open.

The dialog should have *parts_lib* listed in the Search Libraries field (Figure 31).

f Click OK.

The design loads without errors.

Figure 31: 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 and browse to *parts_lib* in the first directory you created earlier in the lesson.

The dialog should look similar to the one shown in Figure 32.

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

a Right-click *tcounter.vhd* in the Library tab and select **Edit**.

This opens the file in the Source window.

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

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

c Select **File > Save**.

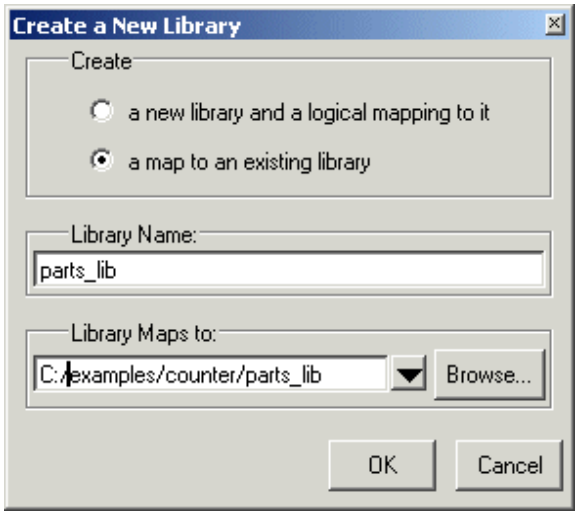
3 Recompile and simulate.

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

b In the Library tab of the Main window, click the '+' icon next to the *work* library and double-click *test_counter*.

The design loads without errors.

Figure 32: Mapping to the parts_lib library



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 **File > Close > Project**. Click OK.

Lesson 5 - Viewing simulations in the Wave window

Topics

The following topics are covered in this lesson:

Introduction

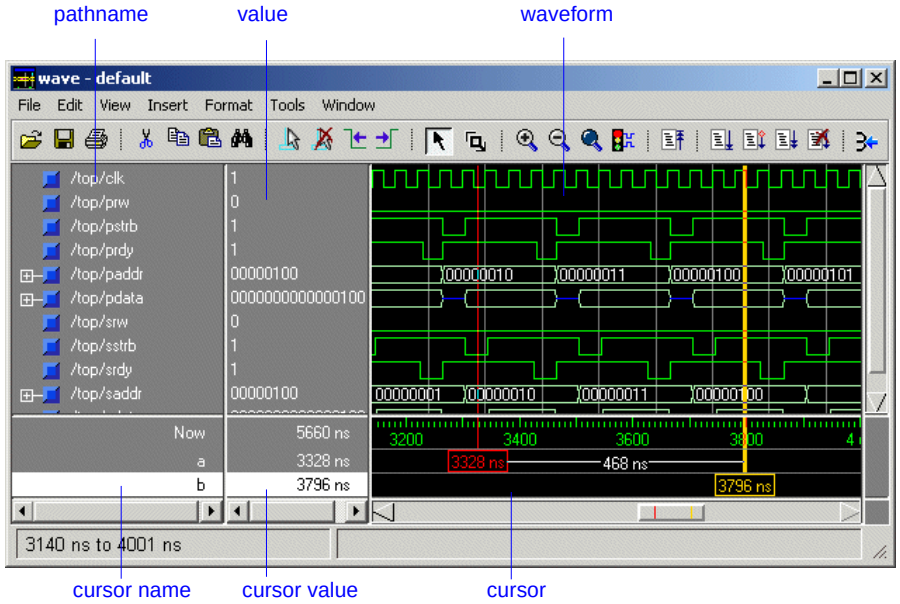
The Wave window allows you to view the results of your simulation as HDL waveforms and their values.

The Wave window is divided into a number of window panes (Figure 33). All window panes in the Wave window can be resized by clicking and dragging the bar between any two panes.

Related reading

ModelSim User's Manual – "Wave window" (UM-236), *Chapter 7 - WLF files (datasets) and virtuals* (UM-141)

Figure 33: The Wave window and its many panes



Loading a design

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

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

   a 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 items to the Wave window

ModelSim offers several methods for adding items to the Wave window. In this exercise, you'll try out different methods.

1 Add items from the Signals window.

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

   b Select **View > Signals** to open the Signals window.

   c In the Signals window, select **Add > Wave > Signals in Design**.

   ModelSim adds three signals to the Wave window.

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

   This deletes all items in the window.

2 Add items using drag-and-drop.

   You can drag an item to the Wave window from many other windows (e.g., Main, Signals, and Variables).

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

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

   b Drag a signal from the Signals window to the Wave window.

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

3 Add items using a command.

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

   ModelSim adds all items from the current region.

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

Zooming the waveform display

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

1 Zoom the display using various techniques.

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

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

You should see blue vertical lines and numbers defining an area to zoom in (Figure 34).

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 35).

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

Figure 34: Zooming in with the mouse pointer

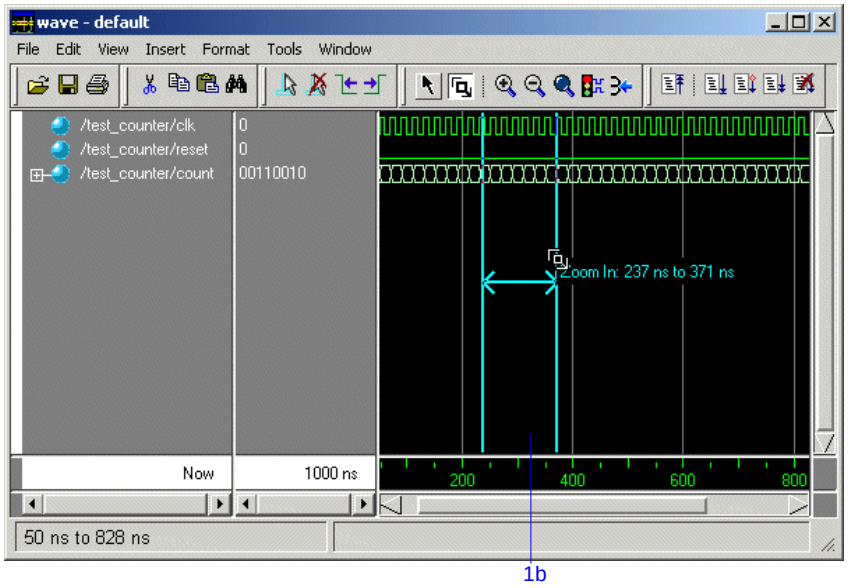
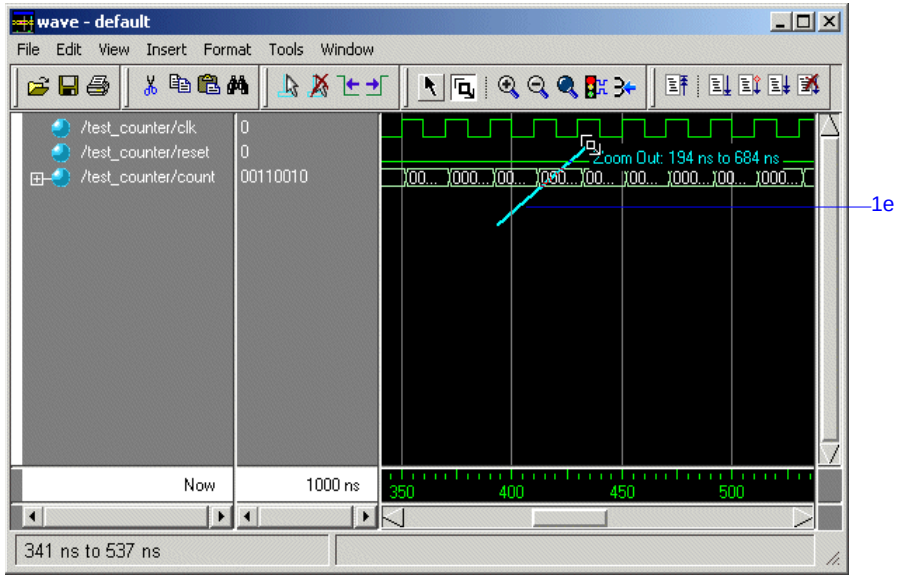


Figure 35: Zooming out with the mouse pointer



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 interval; 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 36).

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.

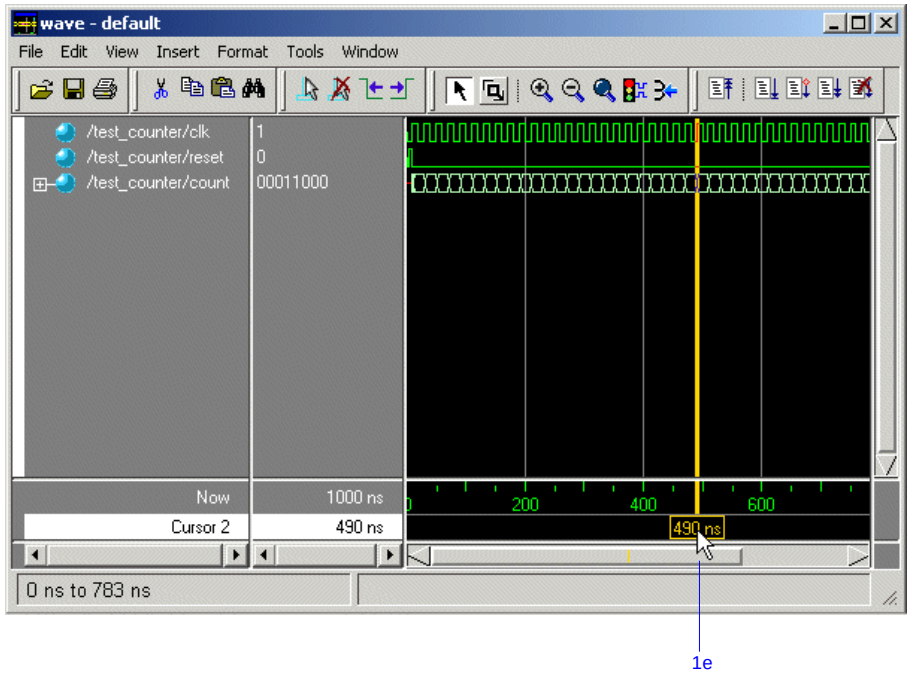
d In the waveform pane, drag the cursor to the right of a transition with the mouse positioned over a waveform.

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 36).

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

Figure 36: 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 37).

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 38).

Figure 37: Renaming a cursor

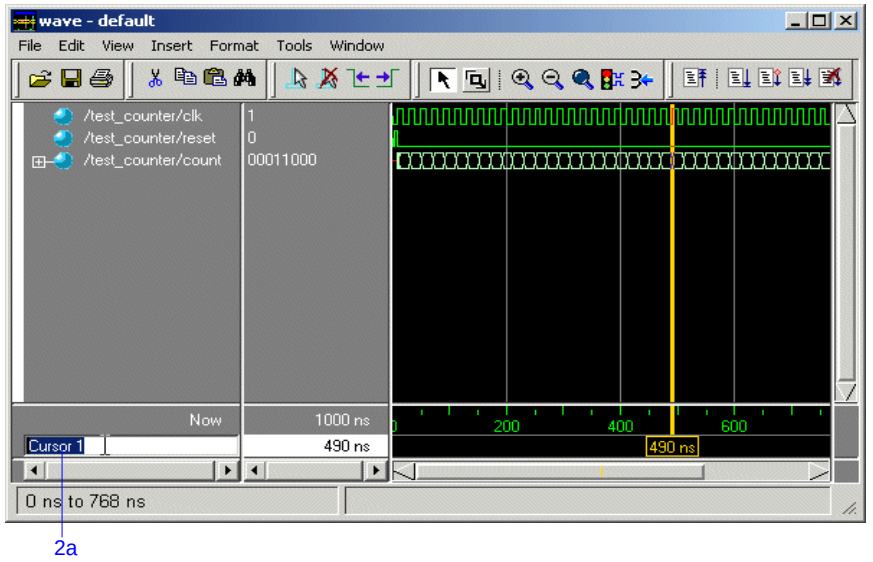
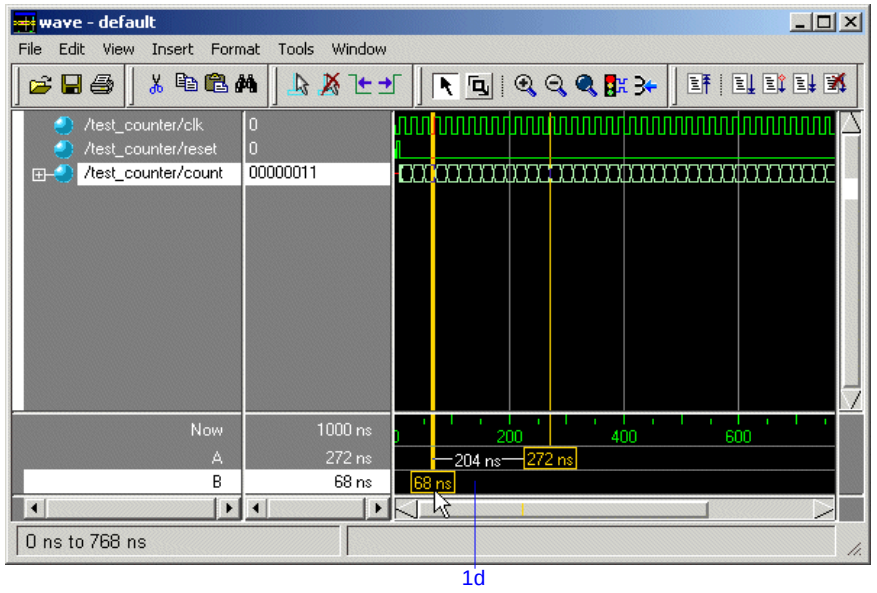


Figure 38: Interval measurement between two cursors



2 Lock cursor *B*.

a With cursor *B* selected, select **Edit > Edit Cursor**.

b Check **Lock cursor to specified time** and click OK (Figure 39).

The cursor color changes to red and you can no longer drag the cursor (Figure 40).

3 Delete cursor *B*.

a With cursor *B* selected, select **Edit > Delete Cursor**.

Figure 39: The cursor properties dialog

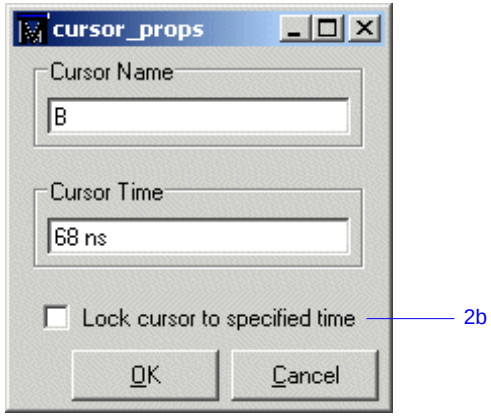
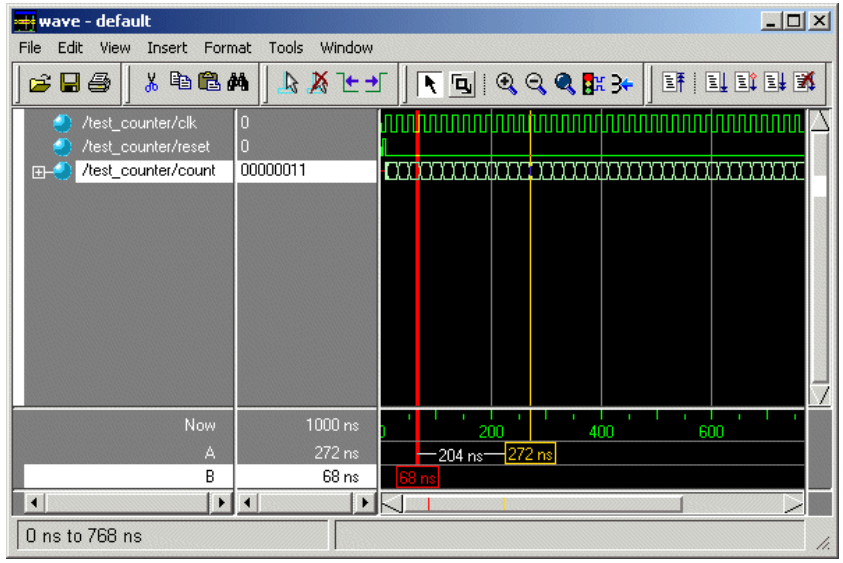


Figure 40: A locked cursor in the Wave window



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

Lesson wrap-up

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

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

Lesson 6 - Viewing and initializing memories

Topics

The following topics are covered in this lesson:

Introduction

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

Note: This lesson uses the Verilog files *dp_syn_ram.v*, *ram_tb.v*, and *sp_syn_ram.v* in the examples. If you are a VHDL user, use *dp_syn_ram.vhd*, *ram_tb.vhd*, and *sp_syn_ram.vhd* instead.

Related reading

ModelSim User's Manual – "Memory window" (UM-201)

ModelSim Command Reference – **mem display** (CR-99), **mem load** (CR-102), **mem save** (CR-105) , **radix** (CR-126) 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 Use the ModelSim icon in Windows.

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

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

3 Create the working library and compile the design.

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

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

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

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

4 Load the design.

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

b Double-click the *ram_tb* design unit (Figure 41). The design appears as shown in Figure 42.

Figure 41: Loading the memory testbench

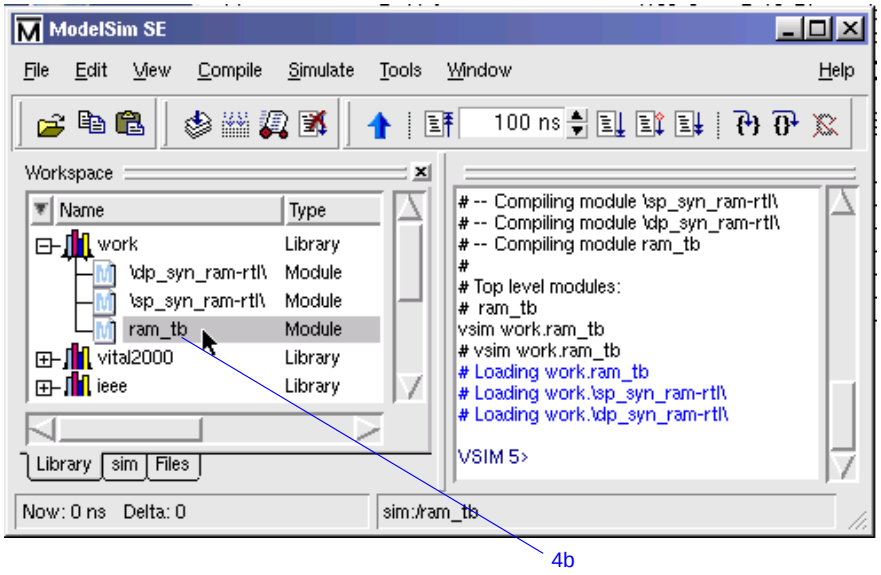
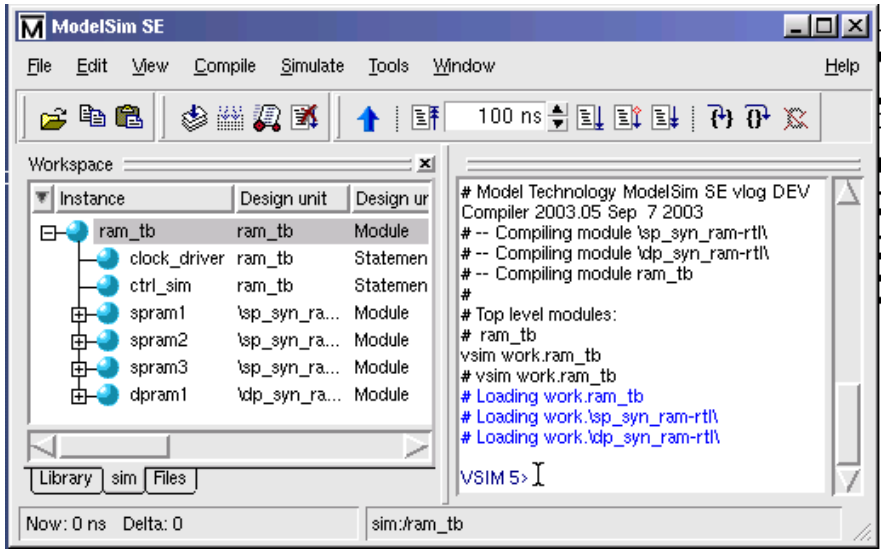


Figure 42: Loaded memory testbench



Viewing a memory

Memories can be viewed via the ModelSim GUI.

1 Open a Memory instance.

a Select **View > Memory** from the Main window menu bar to open the Memory window.

The Memory List pane on the left side of the window lists the memories in the current design context (*ram_tb*).

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 Select the */ram_tb/spram1* instance in the Memory List to view its contents in the Address Data pane.

The data are all **X** (**0** in VHDL) since you have not yet simulated the design (Figure 43).

d In the Main window "sim" tab (as shown in Figure 42 above), select instance *spram2*.

The Memory List pane in the Memory window updates automatically to display *spram2* (Figure 44). However, the Address Data pane still shows the contents of instance *spram1*, because you have not yet opened the *spram2* memory instance.

By default the Memory List updates dynamically to display memories from the current context (i.e., the current design unit). You can make the Memory List static by fixing it to a particular context. To do this you would select **File > Environment > Fix to current context**. For the purposes of this tutorial, you will leave the view dynamic.

Figure 43: Viewing the memory instance

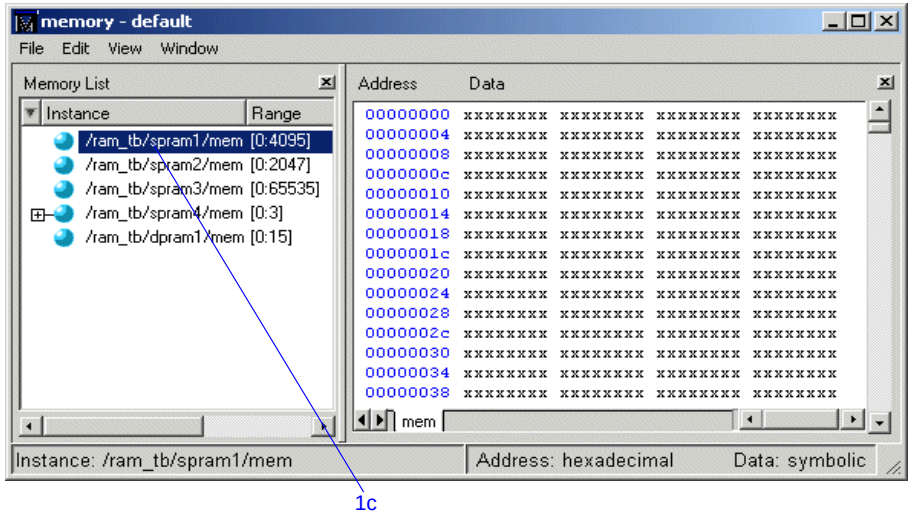
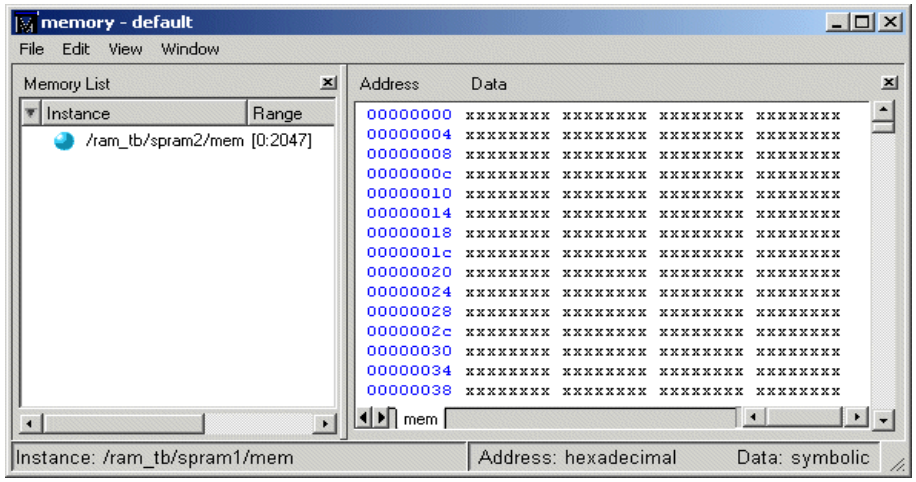


Figure 44: Memory List shows /ram_tb/spram2



2 Simulate the design.

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

The Address Data pane updates to show values from instance *ram_tb/spram1* (Figure 45).

VHDL:
In the transcript window, you will see an assertion failure that is functioning to stop the simulation. The simulation itself has not failed.

You can open additional memory instances by selecting an instance in either the Main window, Signals window, or Structure window, and dragging and dropping it into the Memory window.

3 Open a second memory instance from the Main window.

a In the Main window, drag and drop *spram2* into the Address Data pane of the Memory window.

The contents of *spram2* is displayed, and a new tab is added to the bottom of the Address Data pane (Figure 46). The Memory List now displays only the */ram_tb/spram2* instance.

b Reset the Memory List view to *ram_tb*. Click on the *ram_tb* module in the Main window "sim" tab.

Figure 45: Memory display updates with simulation

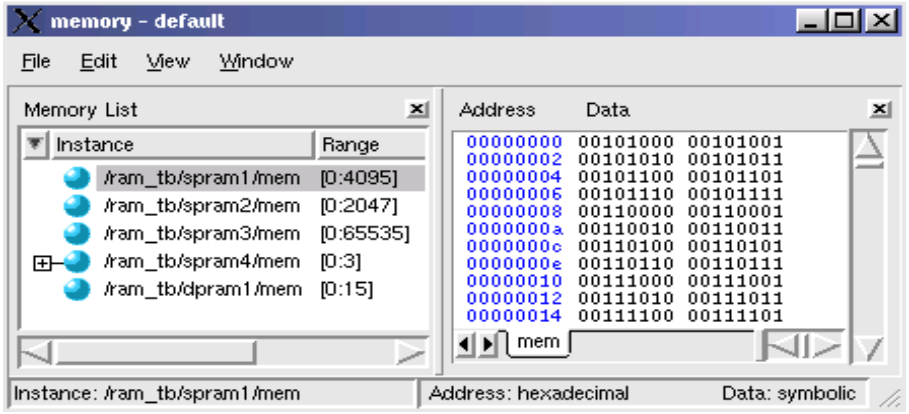
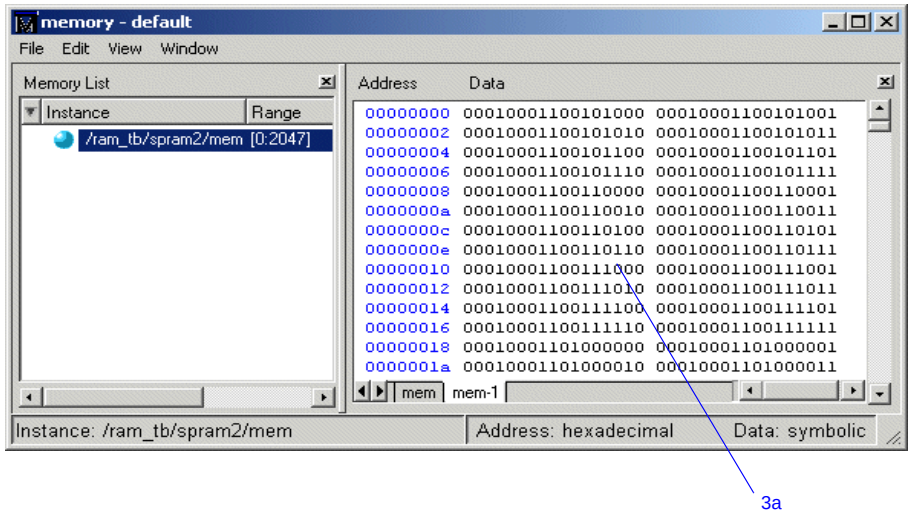


Figure 46: View of spram2 data dragged from Main window



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

a Select the *mem* tab at the bottom of the Address Data pane to view the *ram_tb/spram1* instance.

b Select **View > Display options** to bring up the dialog box (Figure 47).

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

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

e Click OK.

You can see the results of the settings in Figure 48.

Figure 47: Display Options dialog box

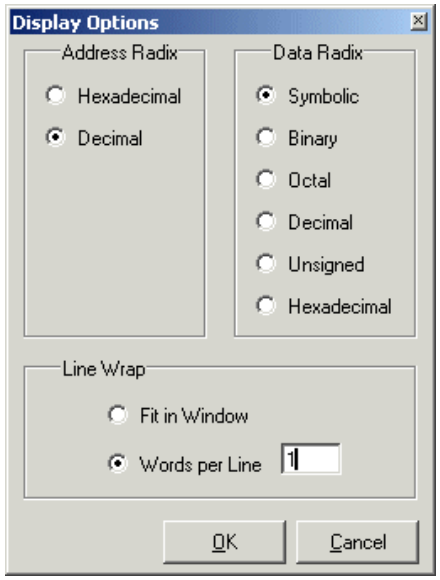
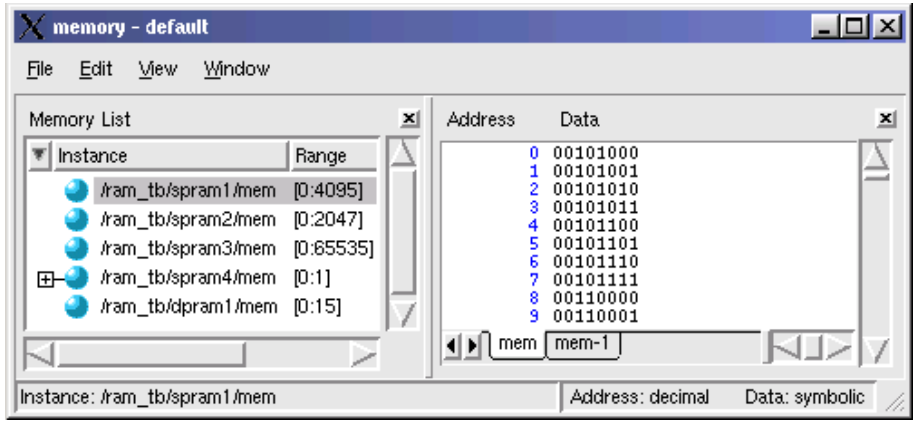


Figure 48: Memory window: new address radix and line length



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 If necessary, click on the */ram_tb/spram1* in the Memory list pane.

b Select **Edit > Goto** from the Memory menu.

The Goto dialog box opens in the data pane (Figure 49).

c Type 12 in the dialog box.

d Click OK.

The requested address appears in the top line of the Address Data pane in the Memory window (Figure 49).

2 Edit the address location directly.

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

a Double click any address in the Address Data pane of the Memory window (Figure 50)

b Enter any desired address.

c Press <Enter> on your keyboard.

The Address Data pane scrolls to that address.

Figure 49: The Goto dialog box

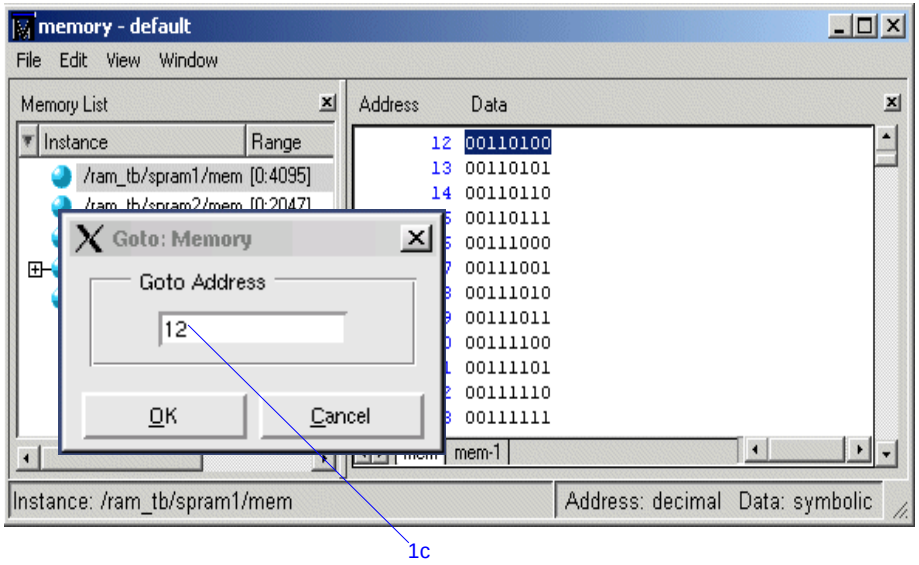
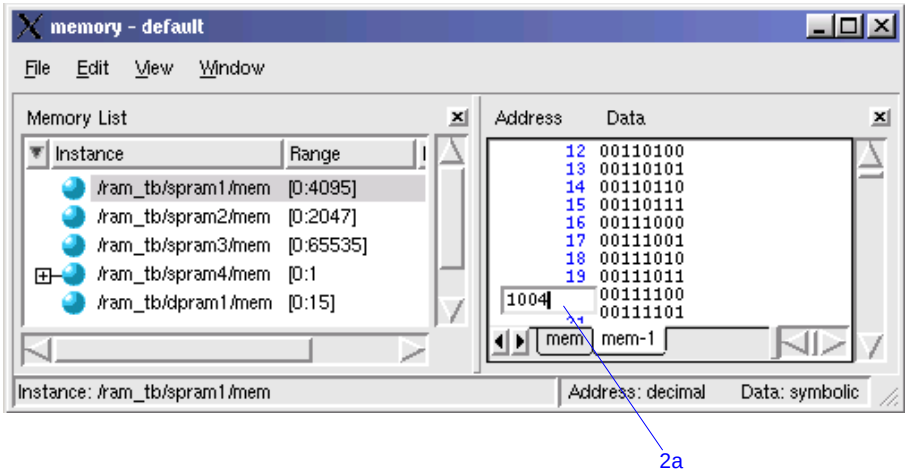


Figure 50: Edit the address directly



3 Let's search for a particular data entry now.

a Select **Edit > Data Search** from the Memory window menu bar.

The Data Search in Memory dialog box opens in the data pane (Figure 51).

b Type **11111010** in the **Search for:** field and click **Search Next**.

The Address Data pane scrolls to the first occurrence of that address (Figure 52). Click Search Next a few more times to search through the list.

c Click Close to close the dialog box.

Figure 51: Find: searching for data value

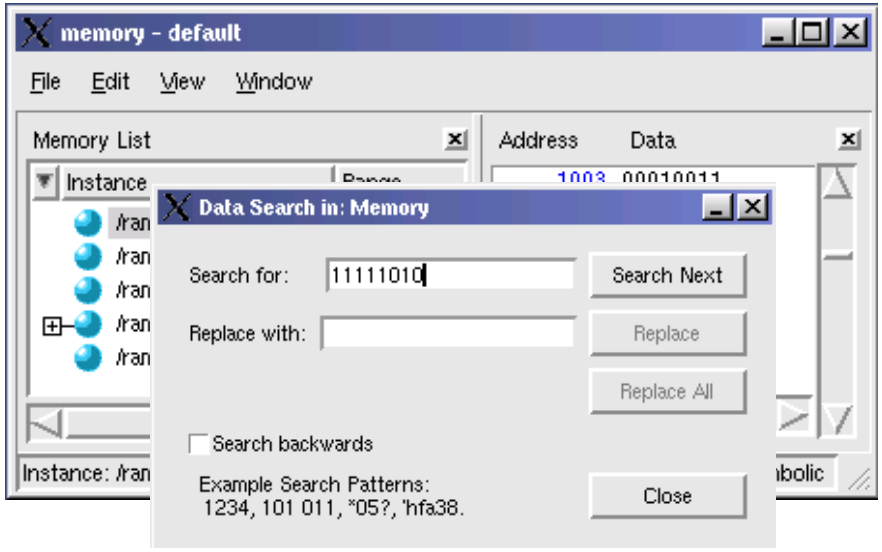
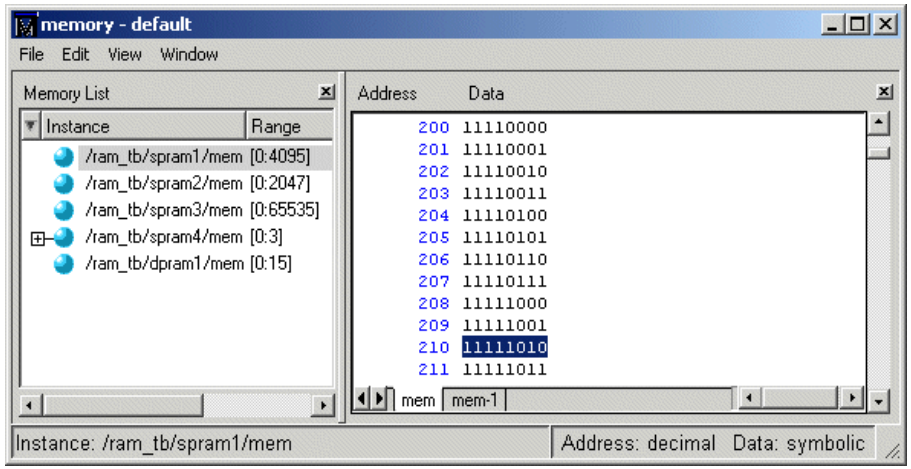


Figure 52: Data value found



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* instance to a file.

   a Select **File > Save** in the Memory window to bring up the Save Memory dialog box (Figure 53).

   Note that **MTI** is the default File Format.

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

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

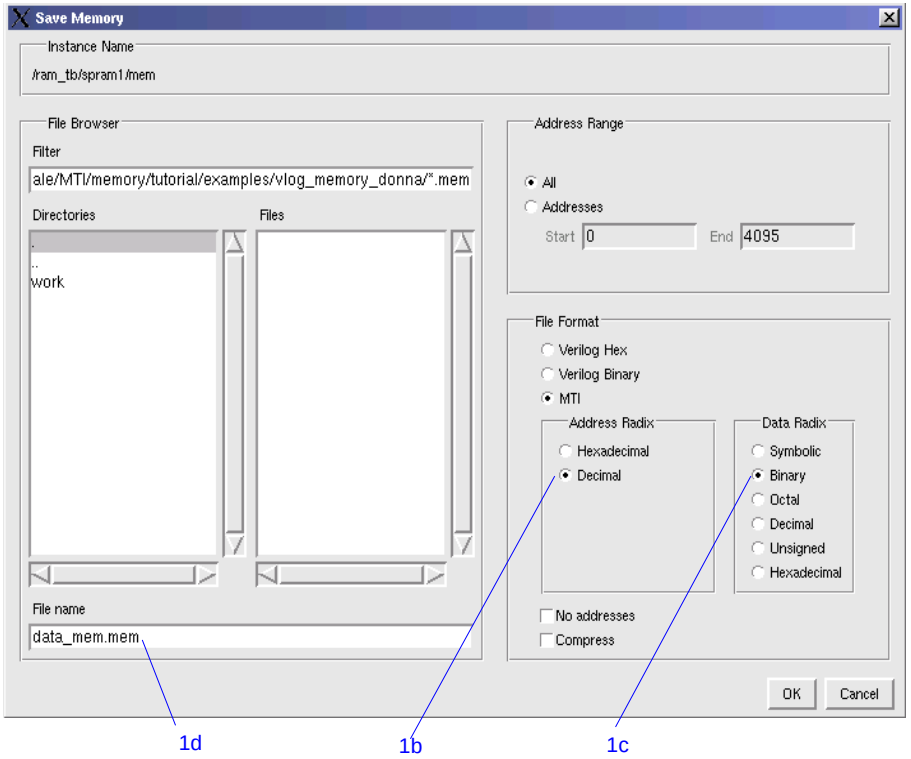
   d Type **data_mem.mem** into the Filename field.

   e 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.

Figure 53: Save Memory dialog box



2 Save a relocatable memory pattern file.

a Click on the *ram_tb/spram2* instance in the Memory List pane.

b Select **View > Display Options** to set the Address Radix of this file to Decimal. Click OK

c Select **File > Save** to bring up the Save Memory dialog box (Figure 54).

d Specify start of address as **0** and end address as **250.**

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

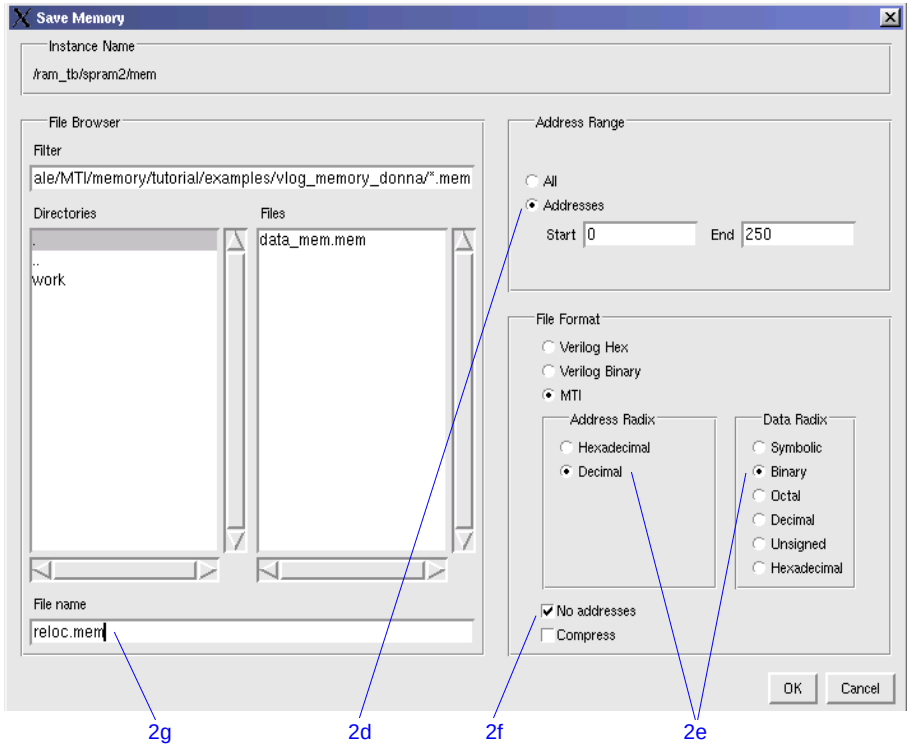
f Click **No addresses** to create a memory pattern that you can use to relocate somewhere else in the memory, or in another memory.

g Enter the file name as **reloc.mem.**

h Click OK

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

Figure 54: Saving a relocatable memory file



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*.

   a Click on the *ram_tb/spram3* instance in the Memory window.

   Scan the contents so you can identify changes once the initialization is complete.

   b Select **View > Display Options** to bring up the Display Options dialog.

   c Change the Address Radix to Decimal and click OK.

2 Initialize *spram3* from a file.

   a Select **File > Load** to bring up the Load Memory dialog box (Figure 55).

   The default Load Type is File Only.

   b Select *data_mem.mem* file in the Files list.

   c Click OK.

   The addresses in instance *ram_tb/spram3* are updated with the data from *data_mem.mem* (Figure 56).

Figure 55: Load Memory dialog box

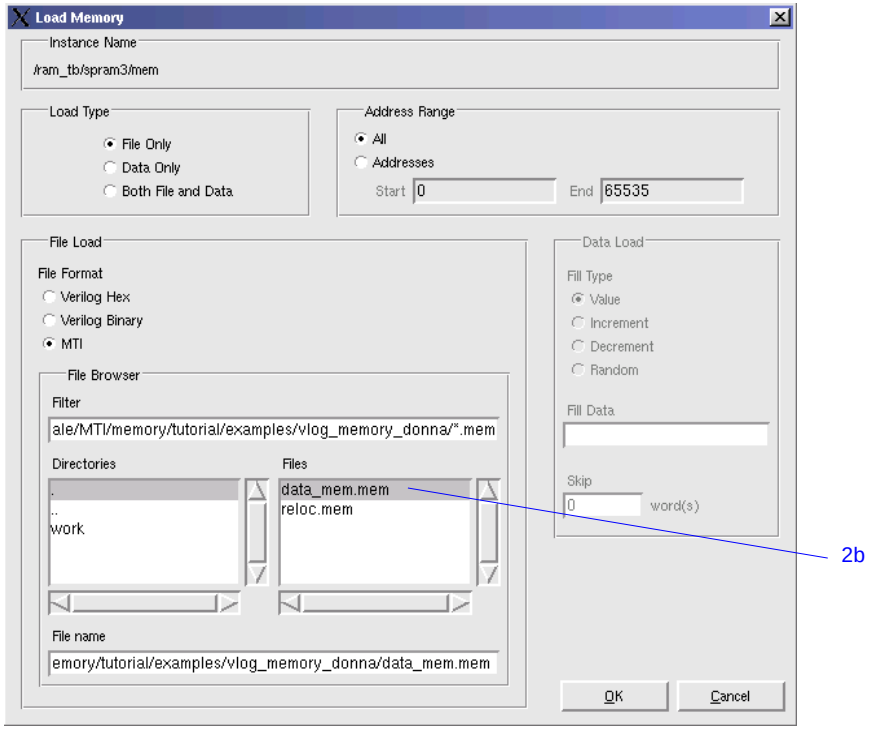
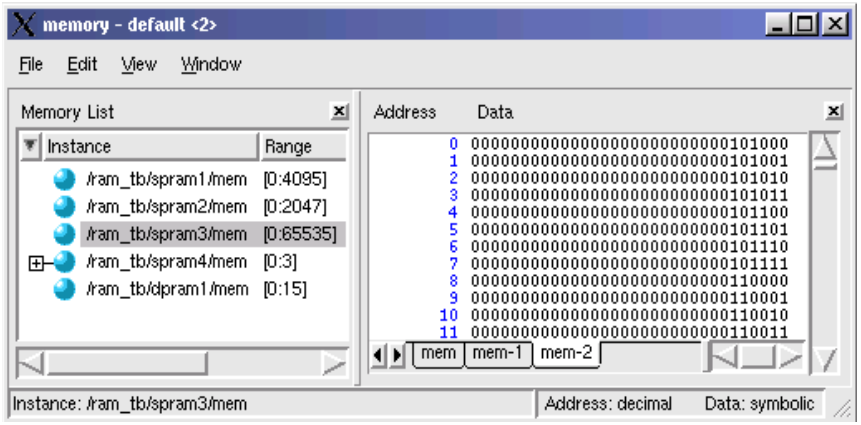


Figure 56: Initialized memory from file and fill pattern



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* instance with a relocatable memory pattern (*reloc.mem*) and a fill pattern.

a Select **File > Load** to bring up the Load Memory dialog box (Figure 57).

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

c For File Load, select *reloc.mem* from the Files list.

d 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.

e For Fill Type, select **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 58).

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

4 Right click in the Address Data pane and select **Close All**.

Figure 57: Loading a relocatable memory file

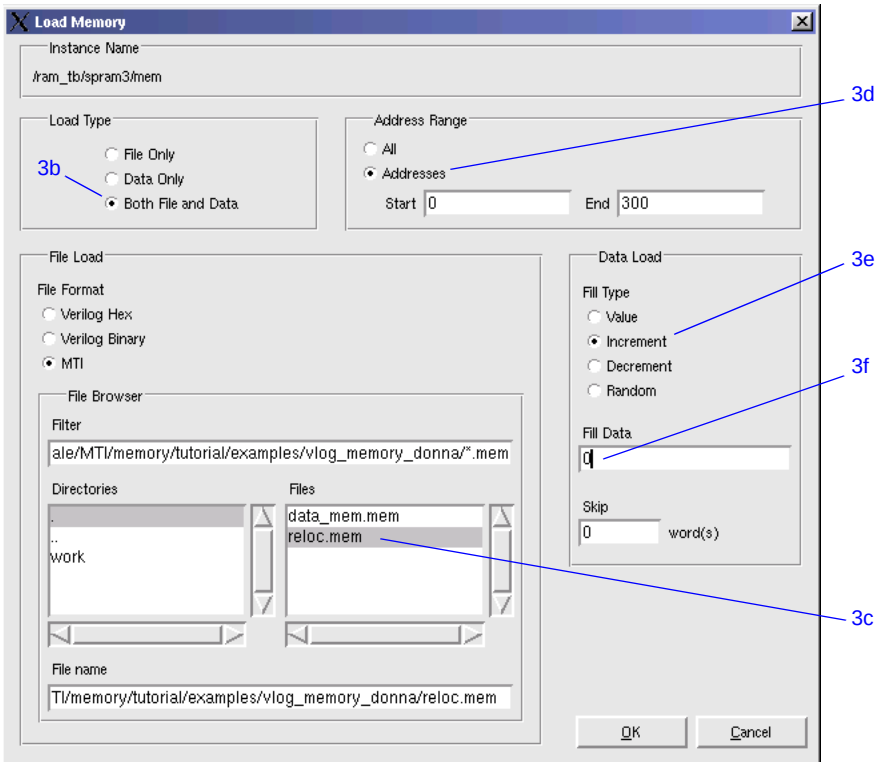
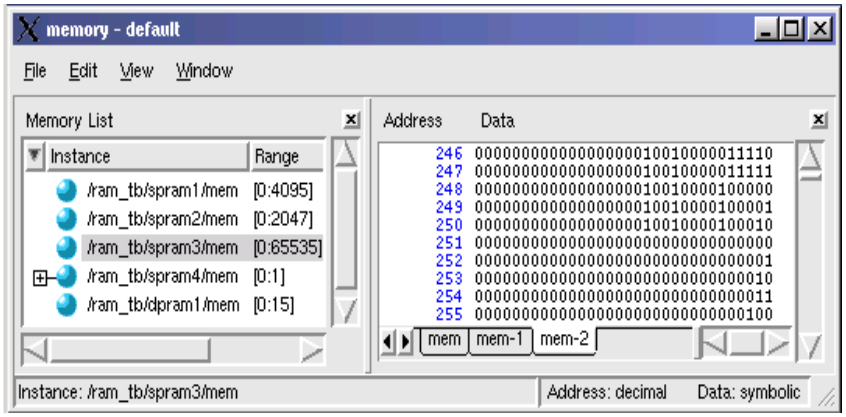


Figure 58: Overwritten values in memory file



Interactive debugging commands

The memory window can 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 Click on the *ram_tb/dpram1* instance in the Memory window.

   b Select **View > Display Options** to bring up the dialog box.

   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 Select **Edit > Change** from the Memory window menu bar (Figure 60).

   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 61).

Figure 59: Original memory contents

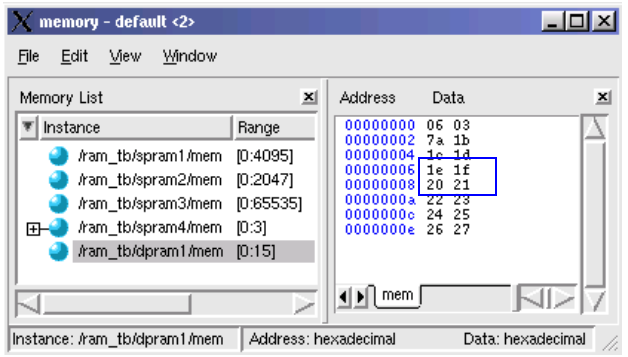


Figure 60: Changing memory contents for a range of addresses

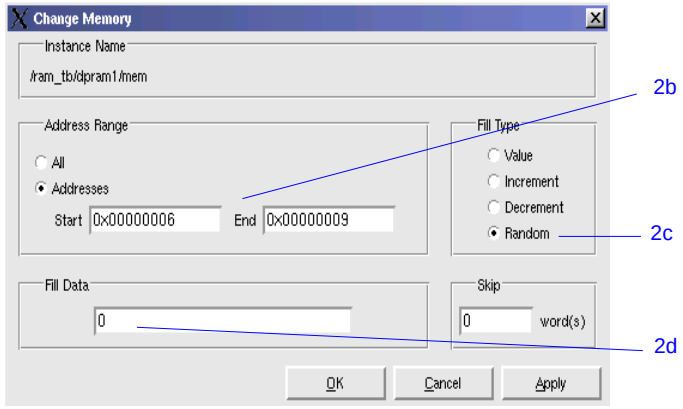
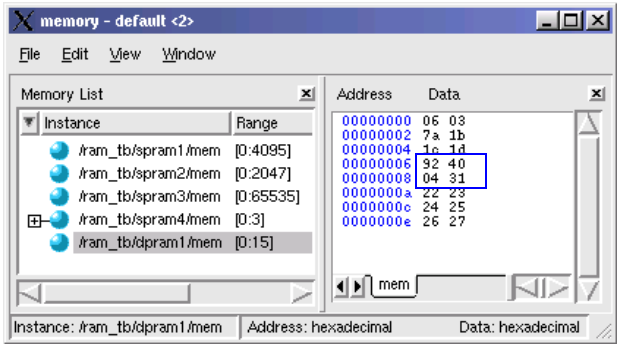


Figure 61: Random contents of a range of addresses



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 62.

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

This brings up the Change dialog box (Figure 63). 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 64).

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.

c Double-click on another value to quickly save the previously edited value and begin editing a new value.

d 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 62: Changing contents by highlighting

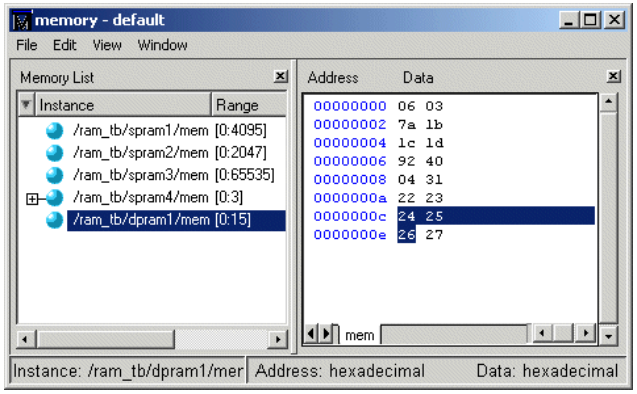


Figure 63: Entering data to change

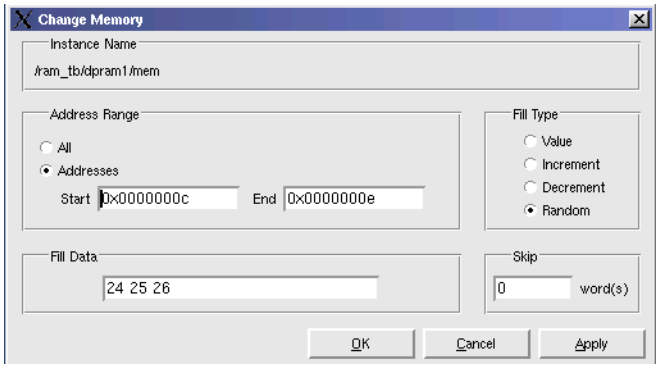
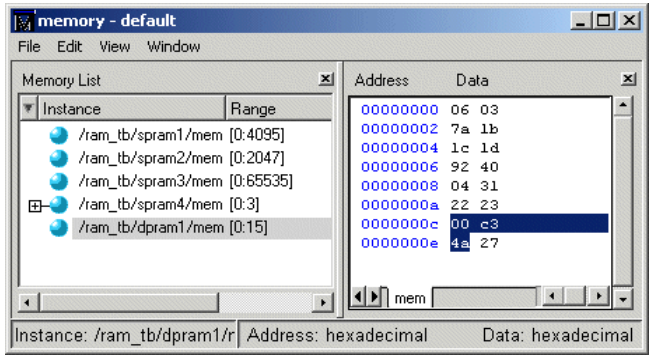


Figure 64: Changed contents for specified addresses



Lesson Wrap-up

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

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

Lesson 7 - 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 – Chapter 12 - Tcl and macros (DO files) (UM-141)

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 *Chapter Lesson 2 - Basic simulation*.

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

   d Select **File > Transcript > Clear Transcript** to clear the transcript.

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

   a Enter the following commands, one at a time, at the VSIM> prompt in the Main 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 transcript.

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

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

4 View the DO file.

a Type **edit sim.do** at the VSIM prompt.

The DO file opens in either the Source window (Figure 65) or the editor you specified with the EDITOR environment variable. Make sure the file includes only those commands shown in step 2 above. Delete any extraneous commands and save the file.

5 Load the simulation anew and use the DO file.

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

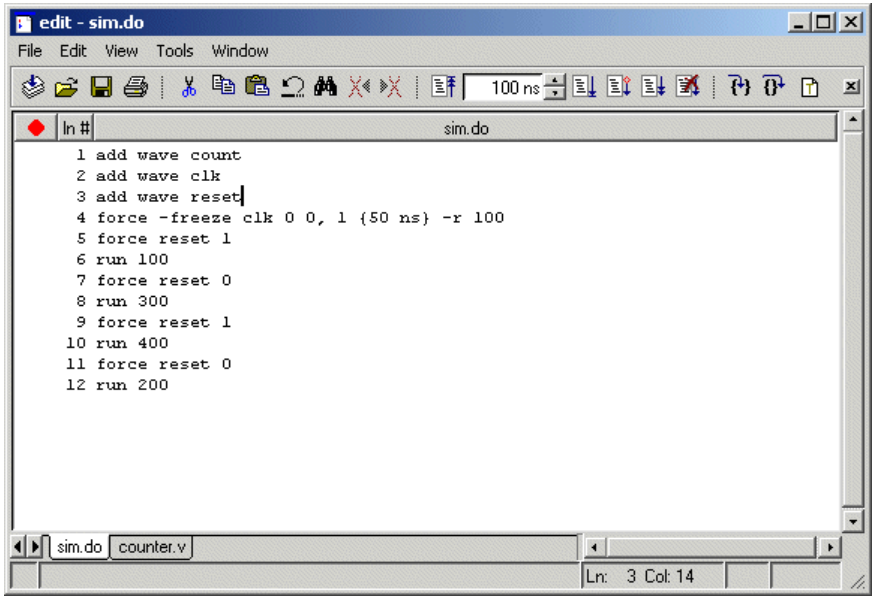
b Type **vsim test_counter** at the VSIM> prompt.

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

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

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

Figure 65: The DO file opened in the Source window



Running ModelSim with a startup DO file

You can configure ModelSim to execute a particular DO file every time you start the tool. You modify the local ModelSim initialization file (*modelsim.ini*) to point at the file, and from that point forward, the DO file is executed whenever you invoke ModelSim from that directory.

In this exercise, you'll create a startup DO file that opens the Signals, Source, and Wave windows and changes their window titles to the name of your design. You'll be working from the system command prompt in the directory you created in *Chapter Lesson 2 - Basic simulation*.

1 Create the DO file.

   a Open a text editor.

   b Type the following lines into a new file:

```
view wave -title "$entity"
view si -title "$entity"
view so -title "$entity"
```

   The pre-defined variable *$entity* will read in the name of the top-level design unit you simulate.

   c Save the file with the name *startup.do* and place it in the directory you created in *Chapter Lesson 2 - Basic simulation*.

2 Edit the *modelsim.ini* file.

   a With a text editor, open the *modelsim.ini* file in the current directory. This should be the same directory to which you saved the *startup.do* file.

   b Find the following line in the file:

```
; Startup = do startup.do
```

   c Remove the semicolon (;) to un-comment the line and then save and close the file.

3 Execute ModelSim from the system prompt.

   a Open a DOS prompt and change to the directory you've been using in the last two steps.

   b Type **vsim -title "counter" test_counter** at the command prompt.

   ModelSim opens, loads the *test_counter* design unit, and then displays the Signals, Source, and Wave windows with the title "test_counter". Notice that we had to use the **-title** argument to **vsim** in order to change the title of the Main window.

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

5 Edit the *modelsim.ini* to remove the startup DO file.

   a With a text editor, open the *modelsim.ini* file in the current directory.

   b Find the following line in the file:

   ```
   Startup = do startup.do
   ```

   c Add a semicolon (;) and space to the beginning of the line and then save and close the file.

Running ModelSim in command-line mode

We use the term "command-line mode" to refer to simulations that are run from a DOS 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 prompt in the new directory you created in step 1.

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

b For Verilog, type **vlog counter.v** at the DOS 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 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 prompt.

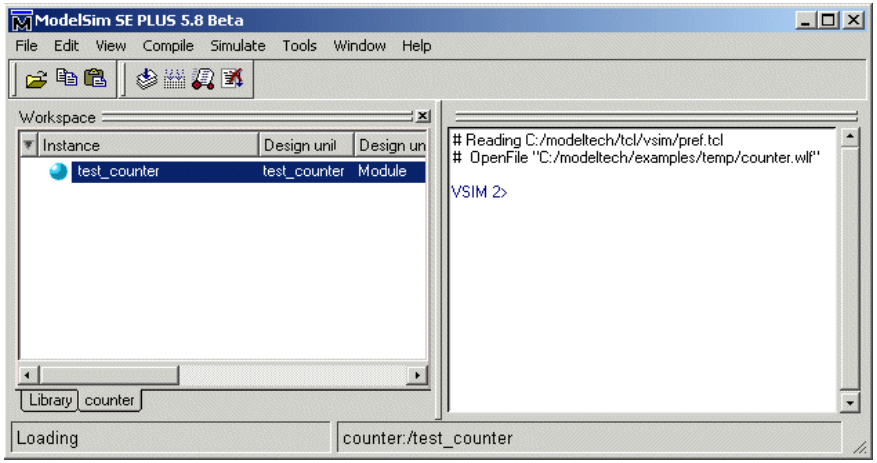
The GUI opens and a dataset tab named "counter" is displayed in the Workspace (Figure 66).

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 66: 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 *Chapter Lesson 2 - Basic simulation*.

2 Load the *test_counter* design unit.

a Start ModelSim.

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

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

3 Execute the DO file and run the design.

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

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

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

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

Lesson Wrap-up

This concludes this lesson.

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

Index

P

Q

R

S

T

V

W

Z