

# Model*Sim*

**Xilinx**

**Tutorial**

**Version 5.5e**

**Published: 23/Aug/01**

The world's most popular HDL simulator

ModelSim is produced by Model Technology™ Incorporated. Unauthorized copying, duplication, or other reproduction is prohibited without the written consent of Model Technology.

The information in this manual is subject to change without notice and does not represent a commitment on the part of Model Technology. The program described in this manual is furnished under a license agreement and may not be used or copied except in accordance with the terms of the agreement. The online documentation provided with this product may be printed by the end-user. The number of copies that may be printed is limited to the number of licenses purchased.

*ModelSim* is a registered trademark of Model Technology Incorporated. Model Technology is a trademark of Mentor Graphics Corporation. PostScript is a registered trademark of Adobe Systems Incorporated. UNIX is a registered trademark of AT&T in the USA and other countries. FLEXlm is a trademark of Globetrotter Software, Inc. IBM, AT, and PC are registered trademarks, AIX and RISC System/6000 are trademarks of International Business Machines Corporation. Windows, Microsoft, and MS-DOS are registered trademarks of Microsoft Corporation. OSF/Motif is a trademark of the Open Software Foundation, Inc. in the USA and other countries. SPARC is a registered trademark and SPARCstation is a trademark of SPARC International, Inc. Sun Microsystems is a registered trademark, and Sun, SunOS and OpenWindows are trademarks of Sun Microsystems, Inc. All other trademarks and registered trademarks are the properties of their respective holders.

Copyright (c) 1990 - 2001, Model Technology Incorporated.

All rights reserved. Confidential. Online documentation may be printed by licensed customers of Model Technology Incorporated for internal business purposes only.

### **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 .....	T-5
Before you begin .....	T-7
Lesson 1 - Creating a Project .....	T-9
Lesson 2 - Basic VHDL simulation .....	T-15
Lesson 3 - Basic Verilog simulation .....	T-23
Lesson 4 - Debugging a VHDL design .....	T-37
Lesson 5 - Running a batch-mode simulation .....	T-47
Lesson 6 - Executing commands at startup .....	T-51
Lesson 7 - Using the Wave window .....	T-53
Index .....	T-63



# Introduction

---

## Chapter contents

[Software versions](#) . . . . . T-6

[Standards supported](#) . . . . . T-6

[Assumptions](#) . . . . . T-6

## Software versions

This documentation was written to support ModelSim 5.5d for Microsoft Windows 95/98/ME/NT/2000. If the ModelSim software you are using is a later release, check the README file that accompanied the software. Any supplemental information will be there.

Although this document covers both VHDL and Verilog simulation, you will find it a useful reference even if your design work is limited to a single HDL.

## Standards supported

ModelSim VHDL supports both the IEEE 1076-1987 and 1076-1993 VHDL, the 1164-1993 *Standard Multivalued Logic System for VHDL Interoperability*, and the 1076.2-1996 *Standard VHDL Mathematical Packages* standards. Any design developed with ModelSim will be compatible with any other VHDL system that is compliant with either IEEE Standard 1076-1987 or 1076-1993.

ModelSim Verilog is based on the IEEE Std 1364 *Standard Hardware Description Language Based on the Verilog Hardware Description Language*. The Open Verilog International *Verilog LRM version 2.0* is also applicable to a large extent. Both PLI (Programming Language Interface) and VCD (Value Change Dump) are supported for ModelSim PE and SE users.

In addition, all products support SDF 1.0 through 3.0, VITAL 2.2b, VITAL'95 - IEEE 1076.4-1995, and VITAL 2000.

## 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 ModelSim.

We also assume that you have a working knowledge of VHDL and 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.)

Additional details for VHDL and Verilog simulation can be found in the *ModelSim User's Manual* and *Command Reference*.

### Command, button, and menu equivalents

Many of the lesson steps are accomplished by a button or menu selection. When appropriate, VSIM command line (PROMPT:) or menu (MENU:) equivalents for these selections are shown in parentheses within the step. This example shows three options to the **run -all** command, a button, prompt command, and a menu selection.



(PROMPT: run -all)

(MENU: Run > Run -All)

### Drag and drop

Drag and drop allows you to copy and move signals among windows. If drag and drop applies to a lesson step, it is noted in a fashion similar to MENUS and PROMPTS with: DRAG&DROP.

### Command history

As you work on the lessons, keep an eye on the Main transcript window. The commands invoked by buttons and menu selections are echoed there. You can scroll through the command history with the up and down arrow keys, or the command history may be reviewed with several shortcuts at the ModelSim/VSIM prompt.

Shortcut	Description
click on prompt	left-click once on a previous ModelSim or VSIM prompt in the transcript to copy the command typed at that prompt to the active cursor
his or history	shows the last few commands (up to 50 are kept)

## Reusing commands from the Main transcript

ModelSim's Main transcript can be saved, and the resulting file used as a DO (macro) file to replay the transcribed commands. You can save the transcript at any time before or during simulation. You have the option of clearing the transcript (File > Clear Transcript) if you don't want to save the entire command history.

To save the contents of the transcript select **File > Save Transcript As** from the Main menu.

Replay the saved transcript with the **do** command:

```
do <do file name>
```

For example, if you saved a series of compiler commands as *mycompile.do* (the .do extension is optional), you could recompile with one command:

```
do mycompile.do
```

► **Note:** Neither the prompt nor the Return that ends a command line are shown in the examples.



# Lesson 1 - Creating a Project

---

The goals for this lesson are:

- Create a project

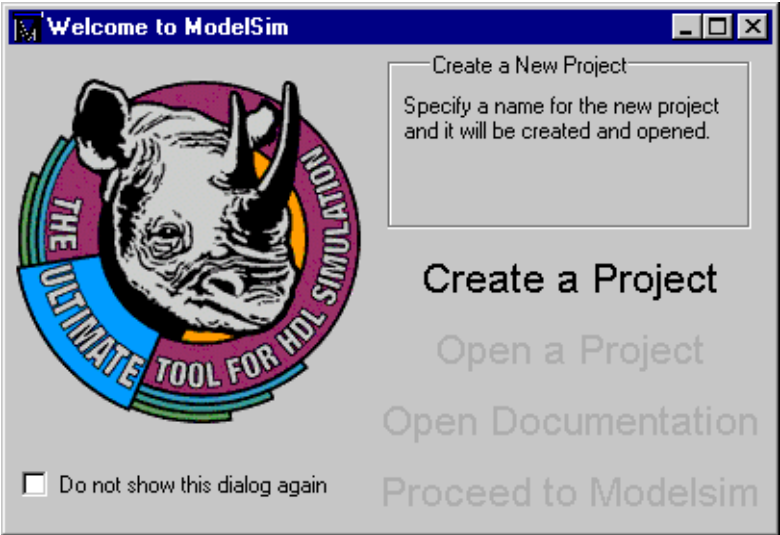
A project is a collection entity for an HDL design under specification or test. Projects ease interaction with the tool and are useful for organizing files and simulation settings. At a minimum, projects have a work library and a session state that is stored in a .mpf file. A project may also consist of:

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

For more information about using project files, see the *ModelSim User's Manual*.

- 1 Start ModelSim with one of the following:  
**for Windows** - your option - from a Windows shortcut icon, from the Start menu, or from a DOS prompt:  

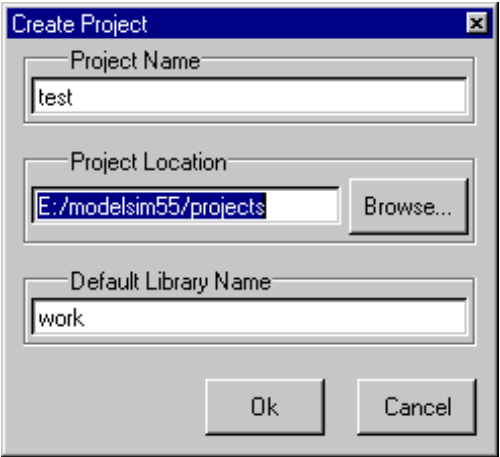
```
modelsim.exe
```
- 2 Upon opening ModelSim for the first time, you will see a Welcome to ModelSim dialog box. (If this screen is not available, you can enable it by selecting **Help > Enable Welcome** from the Main window. It will then display the next time you start ModelSim.)



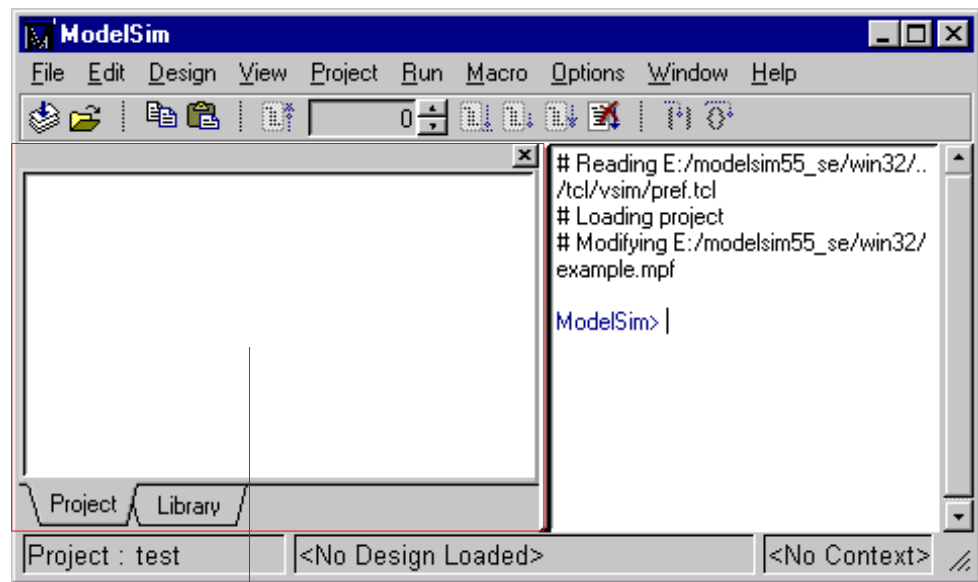
Select **Create a Project** from the Welcome to ModelSim dialog box. Or select **File > New > Project** from the ModelSim Main window.

Selecting **Create a Project** opens the Create Project dialog box.

- 3 In the Create Project dialog box, enter "test" as the Project Name and select a directory where the project file will be stored. Leave the Default Library Name set to "work."

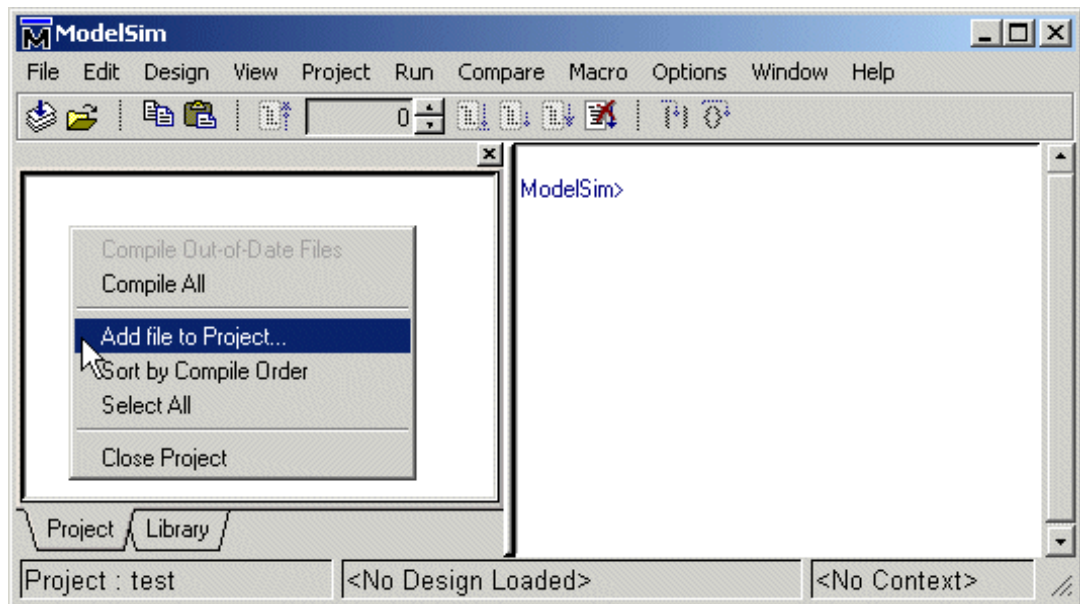


- 4 Upon selecting OK, you'll see the Main window with Project and Library tabs. Notice too that the Project name is shown in the status bar below the Workspace.

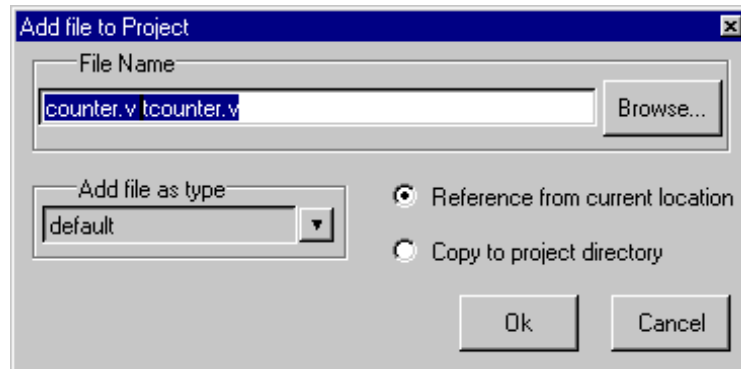


Workspace

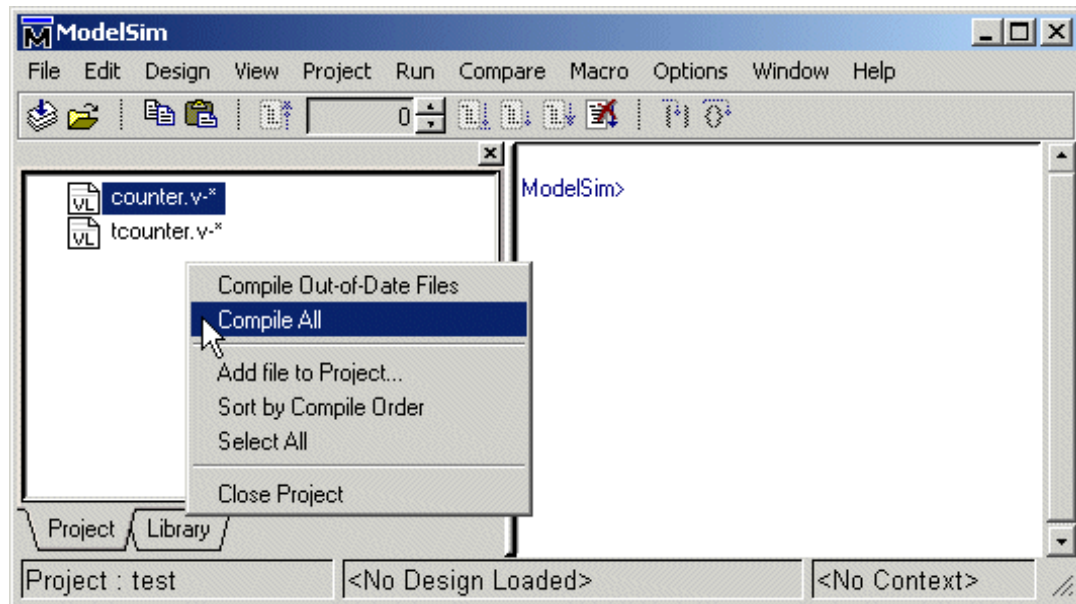
- 5 The next step is to add the files that contain your design units. Click your right mouse button



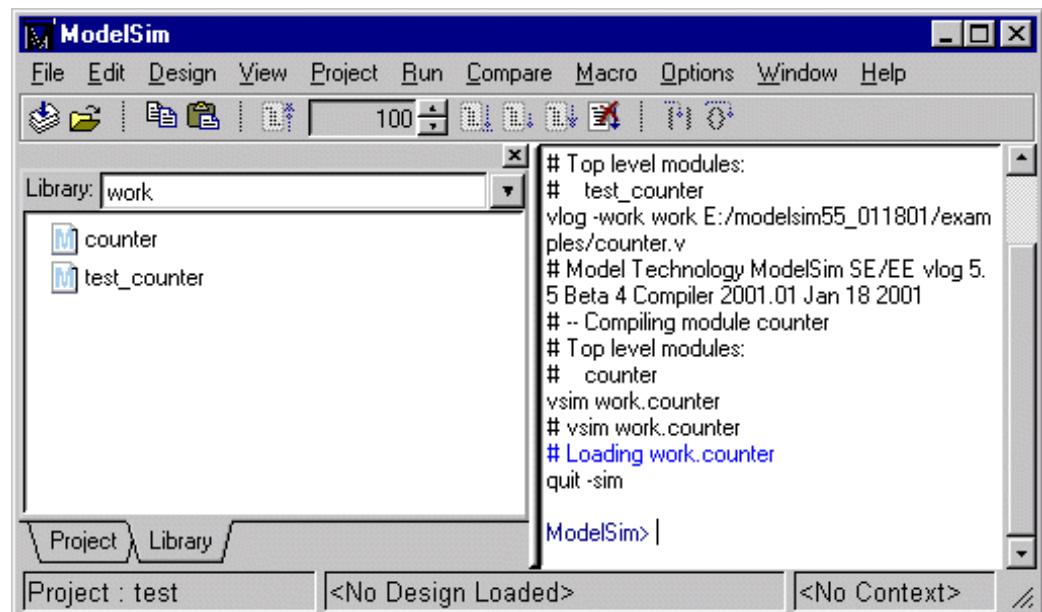
- 6 For this exercise, we'll add two Verilog files. Click the **Browse** button in the Add file to Project dialog box and open the examples directory in your ModelSim installation. Select *tcounter.v* and *counter.v*. Select **Reference from current location** and then click OK.



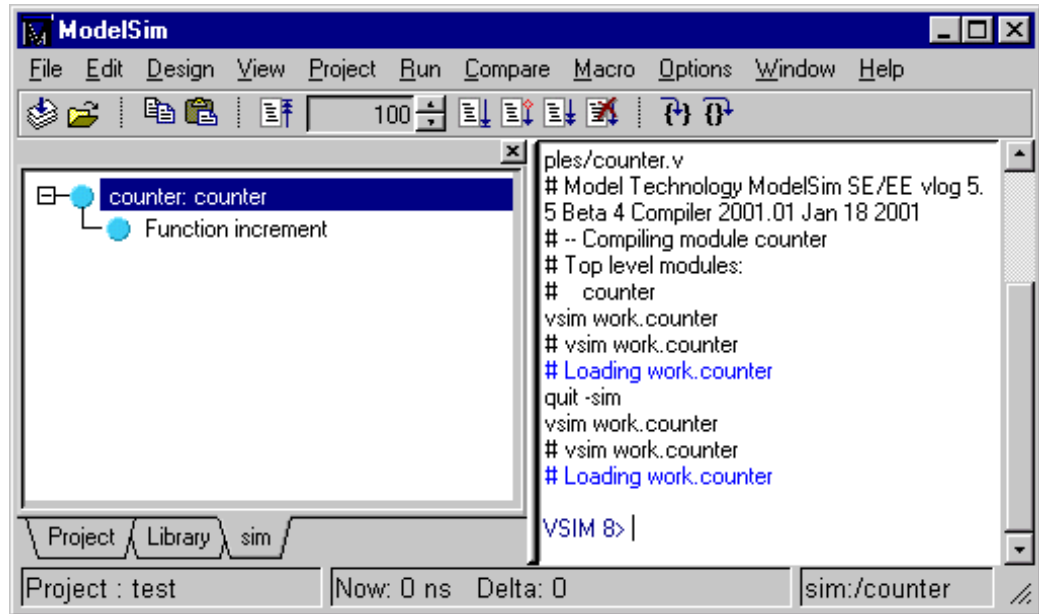
- 7 Right click in the Project page and select **Compile All**.



- 8 The two files are compiled. Click on the Library tab and you'll see the compiled design units listed. If you don't see the design units, make sure the Library: field shows "work."



- 9 The last step in this exercise is to load one of the design units. Double-click *counter* on the Designs page. You'll see a new page appear in the workspace that displays the structure of the *counter* design unit.



At this point, you would generally run the simulation and analyze or debug your design. We'll do just that in the upcoming lessons. For now, let's wrap up by ending the simulation and closing the project. Select **Design > End Simulation** and after confirming you want to quit simulating, select **File > Close > Project**.

## Lesson 2 - Basic VHDL simulation

---

The goals for this lesson are:

- Create a library
- Compile a VHDL file
- Start the simulator
- Learn about the basic *ModelSim* windows, mouse, and menu conventions
- Run *ModelSim* using the **run** command
- List some signals
- Use the waveform display
- Force the value of a signal
- Single-step through a simulation run
- Set a breakpoint

The project feature covered in Lesson 1 executes several actions automatically such as creating and mapping work libraries. In this lesson we will go through the whole process so you get a feel for how *ModelSim* really works.

## Preparing the simulation

- 1 Start by creating a new directory for this exercise (in case other users will be working with these lessons). Create the directory, then copy all of the VHDL (.vhd) files from `\<install_dir>\modeltech\examples` to the new directory.

Make sure the new directory is the current directory. Do this by invoking *ModelSim* from the new directory or by selecting the **File > Change Directory** command from the *ModelSim* Main window.

- 2 Start *ModelSim* with one of the following:

**for Windows** - your option - from a Windows shortcut icon, from the Start menu, or from a DOS prompt:

```
modelsim.exe
```

- **Note:** if you didn't add *ModelSim* to your search path during installation, you will have to include the full path when you type this command at a DOS prompt.

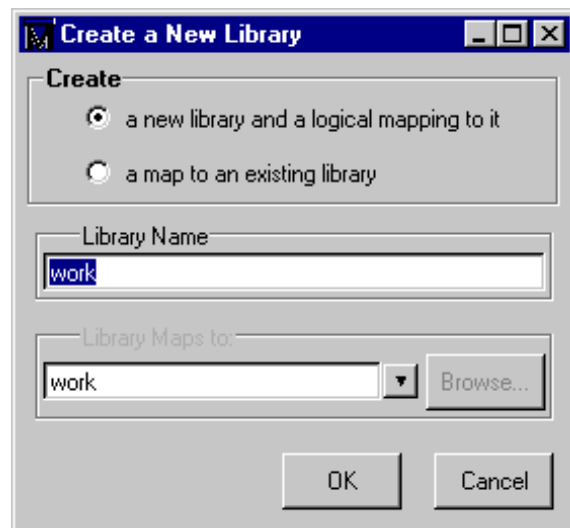
Select "Proceed to ModelSim" if the Welcome dialog appears.

- 3 Before you compile any HDL code, you'll need a design library to hold the compilation results. To create a new design library, make this menu selection in the Main window: **Design > Create a New Library**.

Make sure **Create: a new library and a logical mapping to it** is selected. Type "work" in the Library Name field and then select **OK**.

This creates a subdirectory named *work* - your design library - within the current directory. *ModelSim* saves a special file named *\_info* in the subdirectory.

(PROMPT: vlib work  
vmap work work)



- **Note:** Do not create a Library directory using Windows commands, because the *\_info* file will not be created. Always use the Design menu or the **vlib** command from either the *ModelSim* or DOS prompt.)

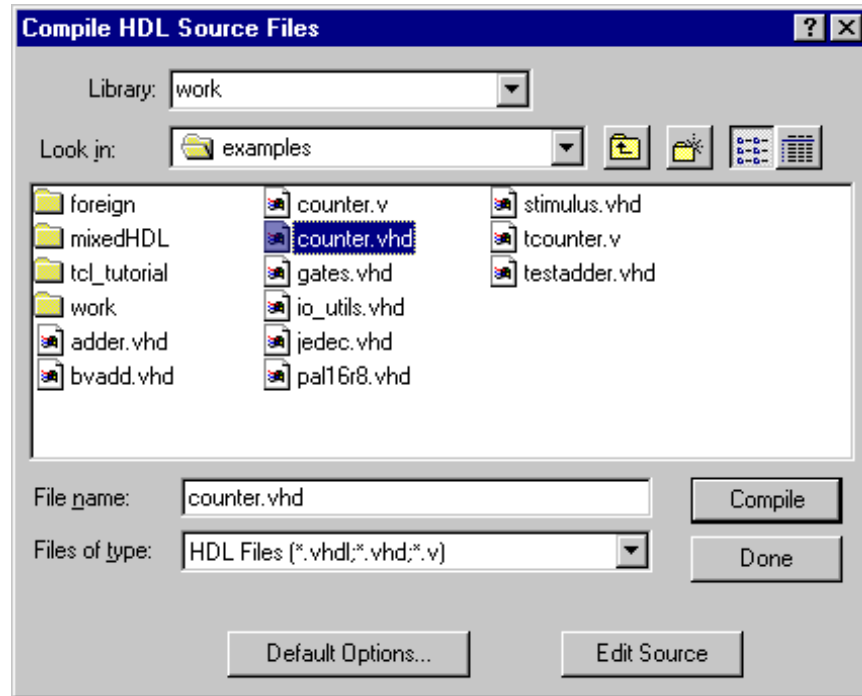


- 4 Compile the file *counter.vhd* into the new library by selecting the **Compile** button on the toolbar:



(PROMPT: vcom counter.vhd)

This opens the Compile HDL Source Files dialog box. (You won't see this dialog box if you invoke vcom from the command line.)



Complete the compilation by selecting *counter.vhd* from the file list and clicking **Compile**. Select **Done** when you are finished.

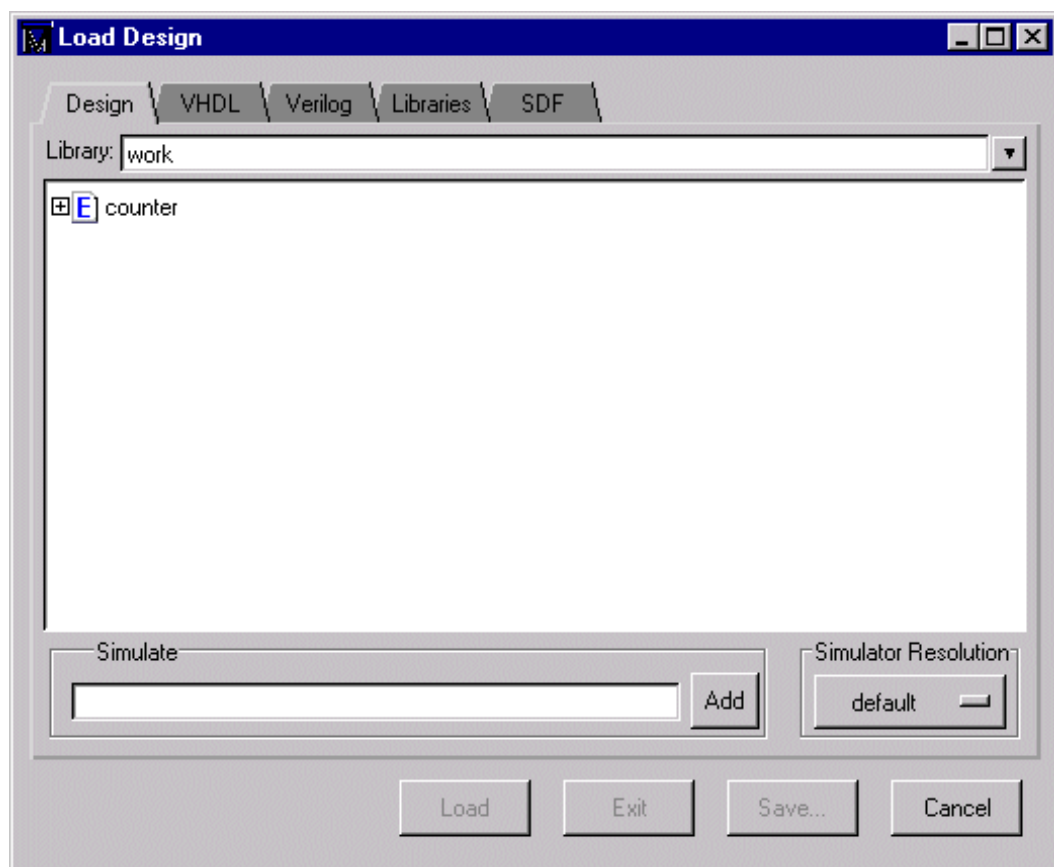
You can compile multiple files in one session from the file list. Individually select and Compile the files in the order required by your design.

- 5 Now let's load the design unit. Select the **Load Design** button from the toolbar:



(PROMPT: vsim counter)

The Load Design dialog box comes up, as shown below (you won't see this dialog box if you invoke **vsim** with *counter* from the command line).



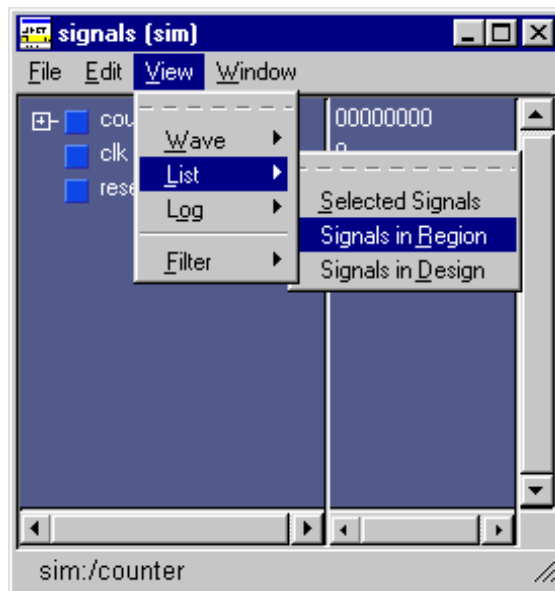
The Load Design dialog box lets you select the library and top-level design unit to simulate. You can also select the resolution limit for this simulation. By default, the following will appear for this simulation run:

- Simulator Resolution: default (the default is 1 ps)
- Library: work
- Design Unit: counter

If the Design Unit is an entity (like **counter** in this design), you can click on the plus-box prefix to view any associated architectures.



- 6 Select the entity **counter** and choose **Load** to accept these settings.
- 7 Next, select **View > All** from the Main window menu to open all ModelSim windows.  
(PROMPT: view \*)  
For descriptions of the windows, consult the *ModelSim User's Manual*.
- 8 From the Signals window menu, select **View > List > Signals in Region**. This command displays the top-level signals in the List window.  
(PROMPT: add list /counter/\*)



- 9 Next add top-level signals to the Wave window by selecting **View > Wave > Signals in Region from the Signals window menu**.  
(PROMPT: add wave /counter/\*)

## Running the simulation

We will start the simulation by applying stimulus to the clock input.

- 1 Click in the Main window and enter the following command at the VSIM prompt:

```
force clk 1 50, 0 100 -repeat 100
```

(MENU: Signals > Edit > Clock)

ModelSim interprets this **force** command as follows:

- force clk to the value 1 at 50 ns after the current time
- then to 0 at 100 ns after the current time
- repeat this cycle every 100 ns

- 2 Now you will exercise two different **Run** functions from the toolbar buttons on either the Main or Wave window. (The **Run** functions are identical in the Main and Wave windows.) Select the **Run** button first. When the run is complete, select **Run All**.



**Run.** This causes the simulation to run and then stop after 100 ns.

(PROMPT: run 100) (MENU: Run > Run 100ns)



**Run -All.** This causes the simulator to run forever. To stop the run, go on to the next step.

(PROMPT: run -all) (MENU: Run > Run -All)

- 3 Select the **Break** button on either the **Main** or **Wave** toolbar to interrupt the run. The simulator will stop running as soon as it gets to an acceptable stopping point.

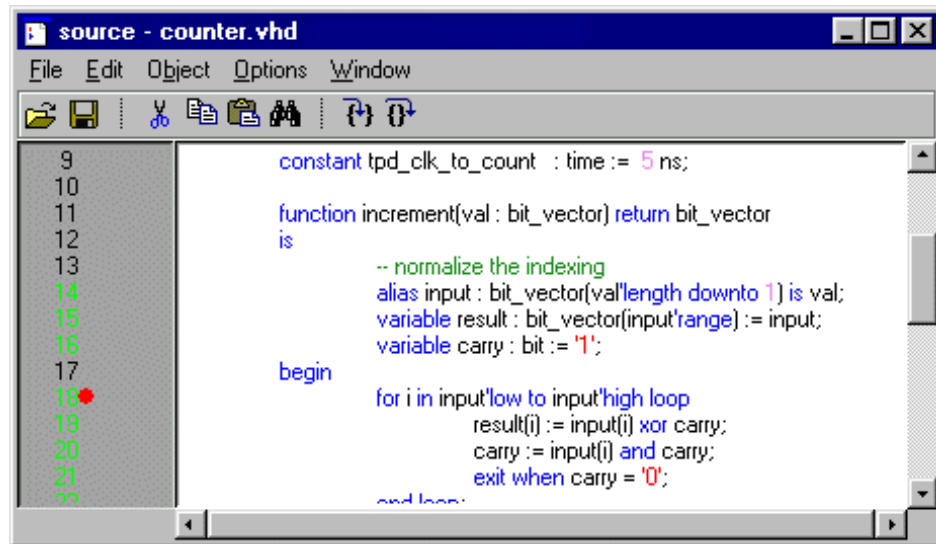


The arrow in the Source window points to the next HDL statement to be executed. (If the simulator is not evaluating a process at the time the Break occurs, no arrow will be displayed in the Source window.)

Next, you will set a breakpoint in the function on line 18.

- 4 Move the pointer to the Source window. Scroll the window vertically until line 18 is visible. Click on or near line number 18 to set the breakpoint. You should see a red dot next to the line number where the breakpoint is set. The breakpoint can be toggled between enabled and disabled by clicking it. When a breakpoint is disabled, the circle appears open. To delete the breakpoint, click the line number with your right mouse button and select Remove Breakpoint 18.

(PROMPT: bp counter.vhd 18)



► **Note:** Breakpoints can be set only on executable lines — denoted by green line numbers.

- 5 Select the **Continue Run** button to resume the run that you interrupted. ModelSim will hit the breakpoint, as shown by an arrow in the Source window and by a Break message in the Main window.



(PROMPT: run -continue) (MENU: Run > Continue)

- 6 Click the **Step** button to single-step through the simulation. Notice that the values change in the Variables window. You can keep clicking **Step** if you wish.



(PROMPT: run -step) (PROMPT: step)

- 7 When you're done, quit the simulator by entering the command:

```
quit -force
```

This command exits ModelSim without asking for confirmation.



## Lesson 3 - Basic Verilog simulation

---

The goals for this lesson are:

- Compile a Verilog design
- List signals in the design
- Examine the hierarchy of the design
- Simulate the design
- Change list attributes
- Set a breakpoint

The project feature covered in Lesson 1 executes several actions automatically such as creating and mapping work libraries. In this lesson we will go through the whole process so you get a feel for how ModelSim really works.

## Preparing the simulation

If you've completed any previous VHDL lesson, you'll notice that Verilog and VHDL simulation processes are almost identical.

- 1 Create and change to a new directory to make it the current directory.

You can make the directory current by invoking *ModelSim* from the new directory or by using the **File > Change Directory** command from the *ModelSim* Main window.

- 2 Copy the Verilog files (files with ".v" extension) from the `\<install_dir>\modeltech\examples` directory into the current directory.

Before you can compile a Verilog design, you need to create a design library in the new directory. If you are familiar only with interpreted Verilog simulators such as Cadence Verilog-XL, this will be a new idea for you. Since *ModelSim* is a compiled Verilog simulator, it requires a target design library for the compilation.

- 3 Invoke *ModelSim*:

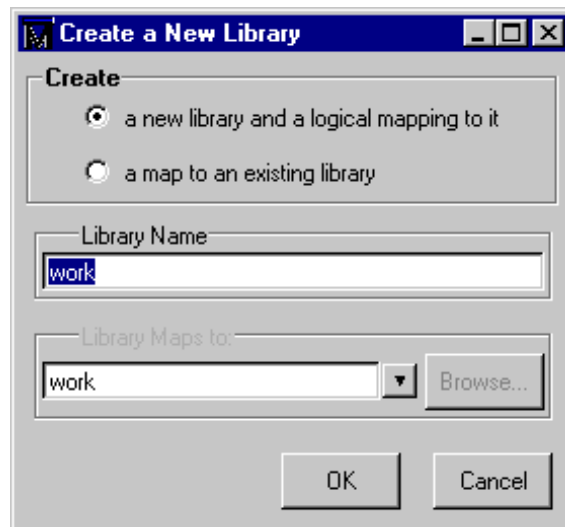
**for Windows** - your option - from a Windows shortcut icon, from the Start menu, or from a DOS prompt:

```
modelsim.exe
```

Select "Proceed to ModelSim" if the Welcome dialog appears.

- 4 Before you compile a source file, you'll need a design library to hold the compilation results. To create a new design library, select **Design > Create a New Library** in the Main window. (PROMPT: vlib work)

In the Create a New Library dialog box, select **Create: a new library and a logical mapping to it**. Type "work" in the **Library Name** field, and then select **OK**. This creates a subdirectory named *work* - your design library - within the current directory. This subdirectory contains a special file named *\_info*.



- **Note:** Do not use DOS commands to create a design library. Always use the Main Design menu or the **vlib** command.



Next, you'll compile the Verilog design.

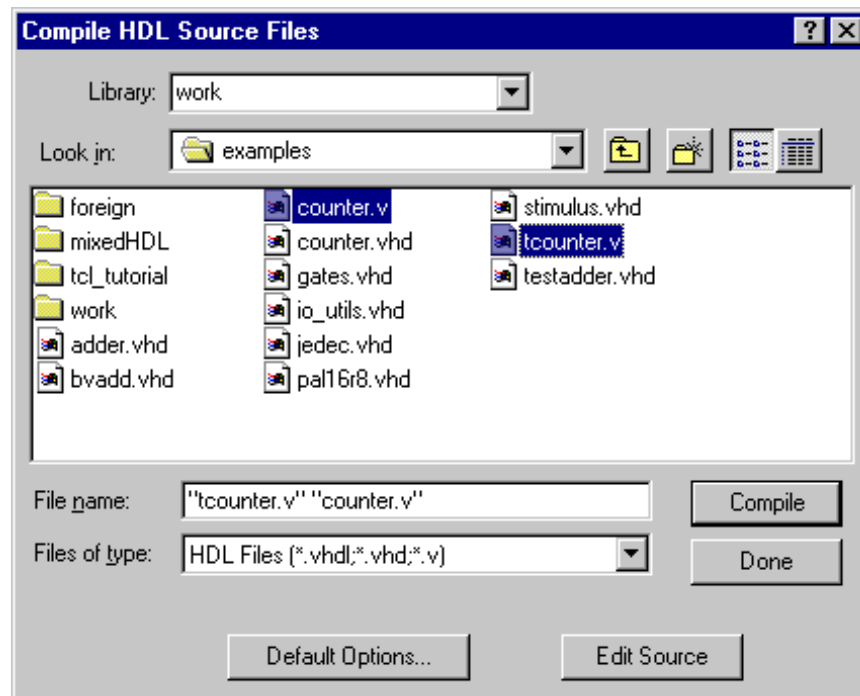
The example design consists of two Verilog source files, each containing a unique module. The file *counter.v* contains a module called **counter**, which implements a simple 8-bit binary up-counter. The other file, *tcounter.v*, is a testbench module (**test\_counter**) used to verify **counter**. Under simulation you will see that these two files are configured hierarchically with a single instance (instance name **dut**) of module **counter** instantiated by the testbench. You'll get a chance to look at the structure of this code later. For now, you need to compile both files into the **work** design library.

- 5 Compile the *counter.v*, and *tcounter.v* files into the **work** library by selecting the **Compile** button on the toolbar:



(PROMPT: vlog counter.v tcounter.v)

This opens the Compile HDL Source Files dialog box.



Complete the compilation by selecting both files. **Control+click** (left mouse button) on *counter.v*, then *tcounter.v* from the file list and choose **Compile**, then **Done**.

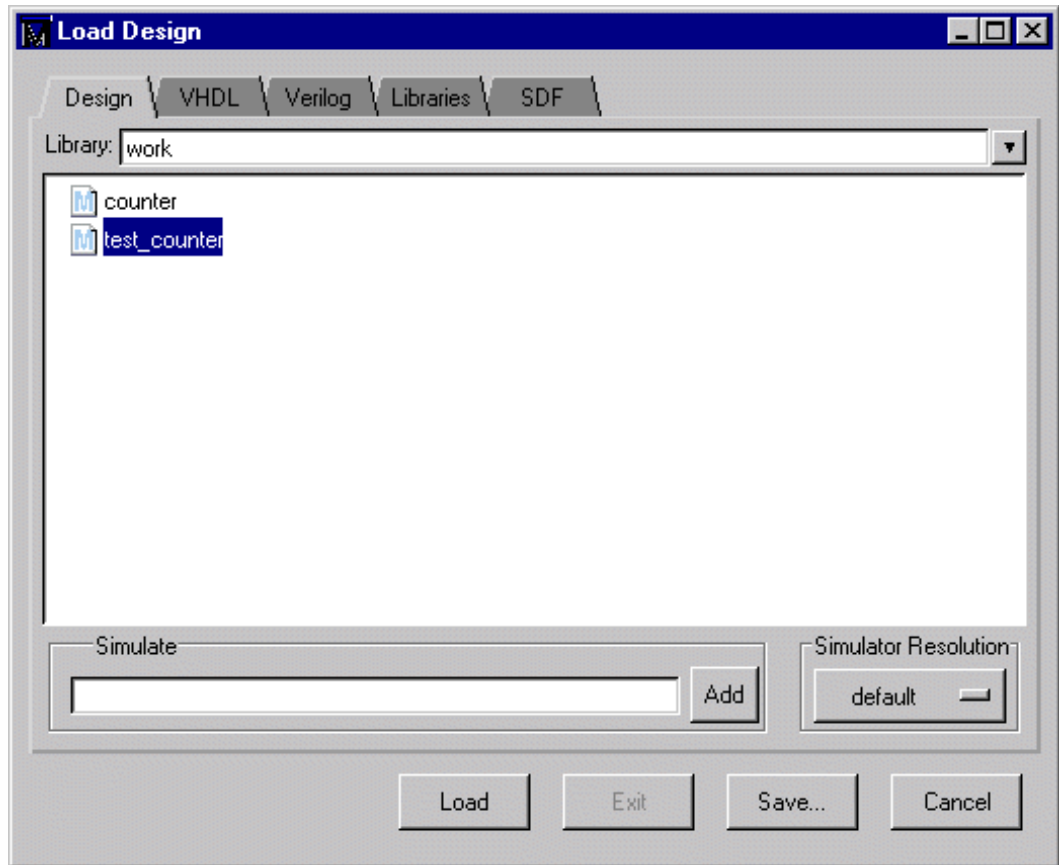
- **Note:** The order in which you compile the two Verilog modules is not important (other than the source-code dependencies created by compiler directives). This may again seem strange to Verilog-XL users who understand the possible problems of interface checking between design units, or compiler directive inheritance. ModelSim defers such checks until the design is loaded. So it doesn't matter here if you choose to compile *counter.v* before or after *tcounter.v*.

- 6 Start the simulator by selecting the **Load Design** button from the toolbar:



(PROMPT: vsim test\_counter)

The Load Design dialog box comes up, as shown below.



The Load Design dialog box allows you to select a design unit to simulate from the specified library. You can also select the resolution limit for the simulation. The default library is **work** and the default resolution is 1 ps.

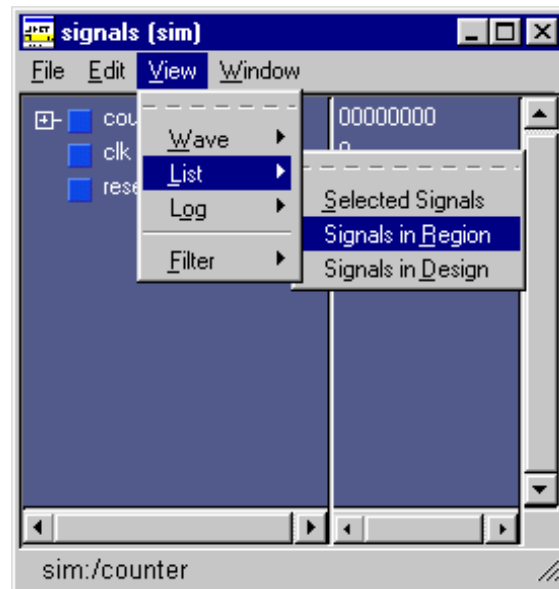
- 7 Select **test\_counter** and click **Load** to accept these settings.
- 8 Bring up the Signals, List and Wave windows by entering the following command at the VSIM prompt within the Main window:

```
view signals list wave
```

(Main MENU: View > <window name>)

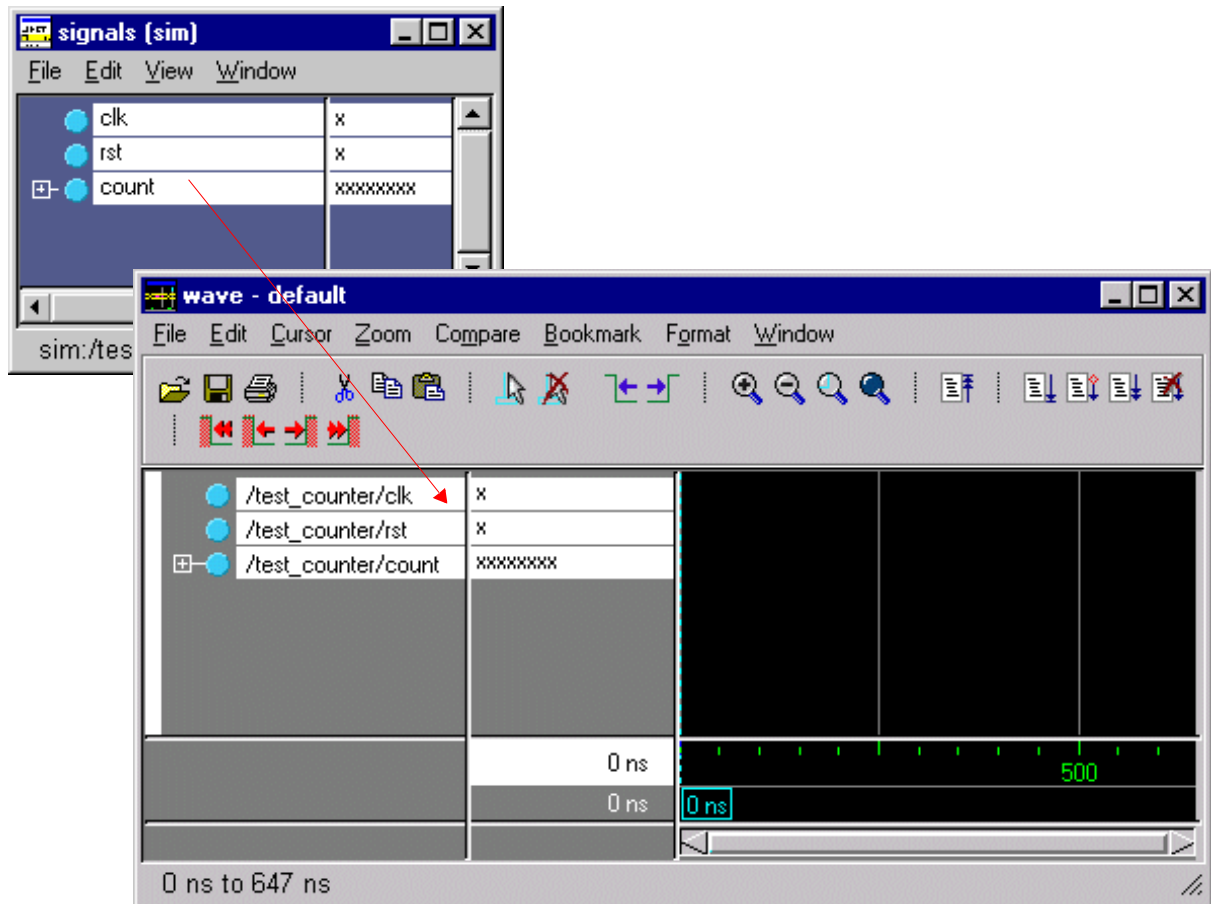
- 9 To list the top-level signals, move the pointer to the Signals window and select **View** > **List** > **Signals in Region**.

(PROMPT: add list /test\_counter/\*)



- 10 Now let's add signals to the Wave window with ModelSim's drag and drop feature.

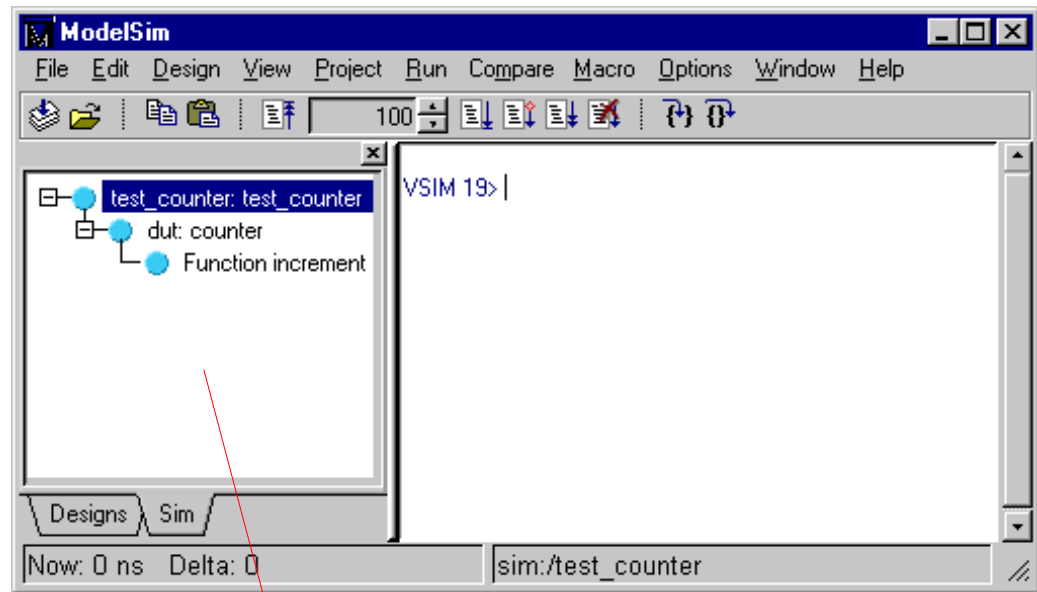
In the Signals window, select **Edit > Select All** to select the three signals. Drag the signals to either the pathname or the values pane of the Wave window.



HDL items can also be copied from one window to another (or within the Wave and List windows) with the **Edit > Copy** and **Edit > Paste** menu selections. You can also delete selected items with the **Edit > Delete** selection.

- 11 Next open the Source window. Select **View > Source** from the Main window.  
(PROMPT: view source)

- 12 You may have noticed when you loaded the design in Step 6 that a new pane appeared in the workspace area of the Main window.



Structure pane

The Structure pane shows the hierarchical structure of the design. By default, only the top level of the hierarchy is expanded. You can navigate within the hierarchy by clicking on any line with a "+" (expand) or "-" (contract) symbol. The same navigation technique works anywhere you find these symbols within ModelSim.

By clicking the "+" next to **dut: counter** you can see all three hierarchical levels: **test\_counter**, **counter** and a function called **increment**. (If **test\_counter** is not displayed you simulated **counter** instead of **test\_counter**.)

- 13 Click on **Function increment** and notice how other ModelSim windows are automatically updated as appropriate.

Specifically, the Source window displays the Verilog code at the hierarchical level you selected in the Structure window. The source-file name is also displayed in the Source window title bar.

Using the Structure pane in this way is analogous to scoping commands in interpreted Verilogs.

For now, make sure the **test\_counter** module is showing in the Source window by clicking on the top line in the Structure pane.

## Running the simulation

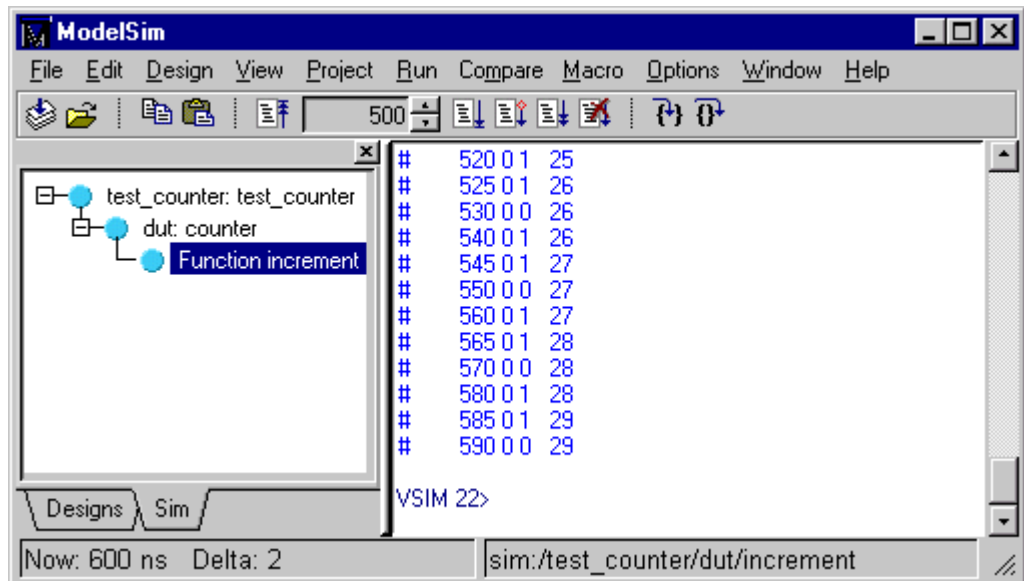
Now you will exercise different Run functions from the toolbar.

- 1 Select the **Run** button on the Main window toolbar. This causes the simulation to run and then stop after 100 ns (the default simulation length).



(PROMPT: run) (MENU: Run > Run 100 ns)

- 2 Next change the run length to 500 on the **Run Length** selector and select the **Run** button again.



Now the simulation has run for a total of 600ns (the default 100ns plus the 500 you just asked for). The status bar at the bottom of the Main window displays this information.

- 3 The last command you executed (**run 500**) caused the simulation to advance for 500ns. You can also advance simulation to a specific time. Type:

```
run @ 3000
```

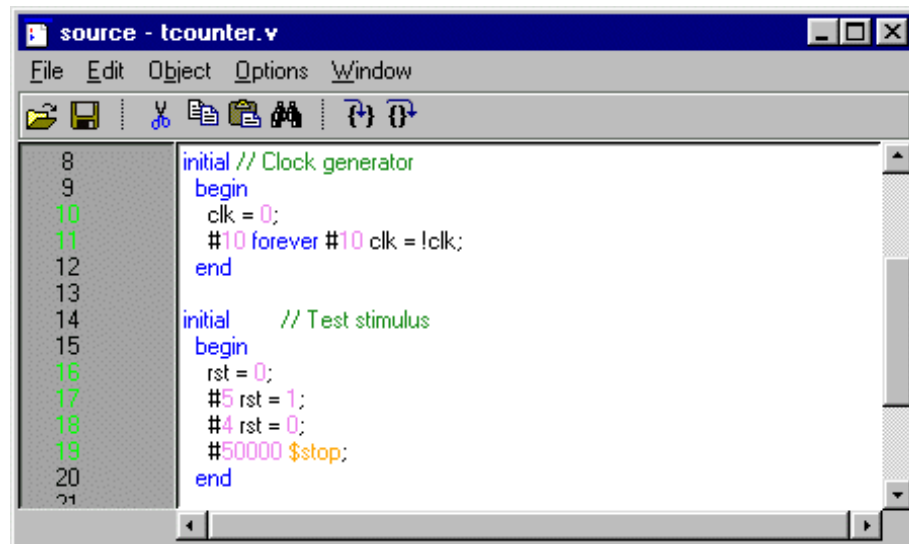
This advances the simulation to time 3000ns. Note that the simulation actually ran for an additional 2400ns (3000 - 600).

- 4 Now select the **Run All** button from the Main window toolbar. This causes the simulator to run forever.



(PROMPT: run -all) (Main MENU: Run > Run -All)

- 5 Select the **Break** button to interrupt the run.

A screenshot of a software window titled 'source - tcounter.v'. The window has a menu bar with 'File', 'Edit', 'Object', 'Options', and 'Window'. Below the menu bar is a toolbar with icons for file operations and simulation control. The main area of the window displays Verilog code with line numbers 8 through 21 on the left. The code is as follows:

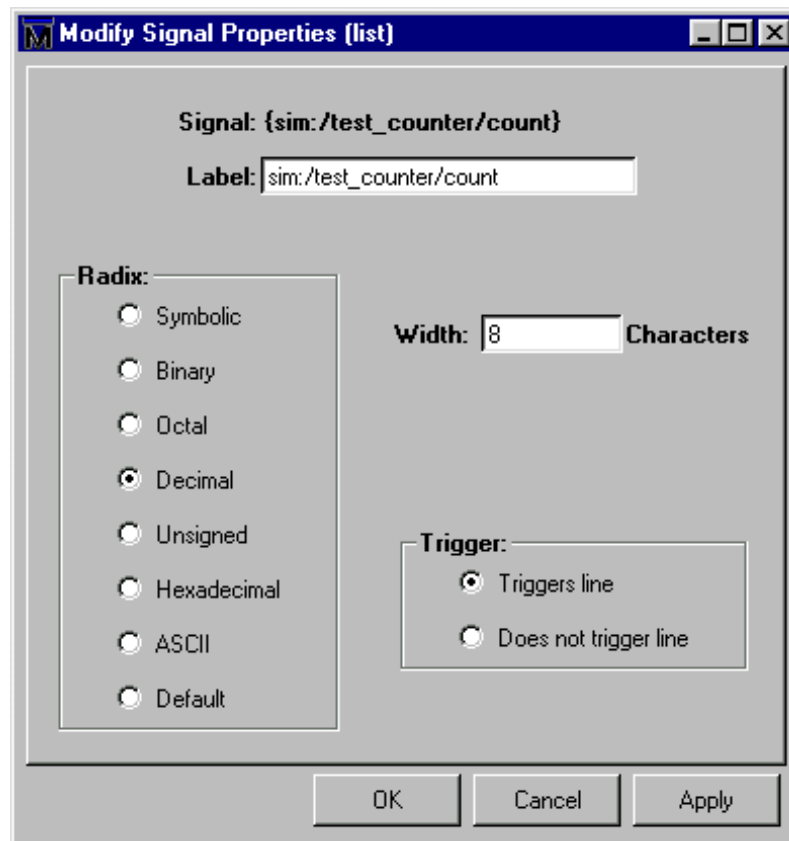
```
8      initial // Clock generator
9      begin
10         clk = 0;
11         #10 forever #10 clk = !clk;
12     end
13
14     initial // Test stimulus
15     begin
16         rst = 0;
17         #5 rst = 1;
18         #4 rst = 0;
19         #50000 $stop;
20     end
21
```

Your Source window won't look exactly like the illustration above because your simulation very likely stopped at a different point.

## Debugging the simulation

Next we'll take a brief look at some interactive debug features of the ModelSim environment. To start with, let's see what we can do about the way the List window presents its data.

- 1 In the List window select **/test\_counter/count**. From the List window menu bar select **Prop > Signal Props**. The Modify Signal Properties (list) dialog box is opened.



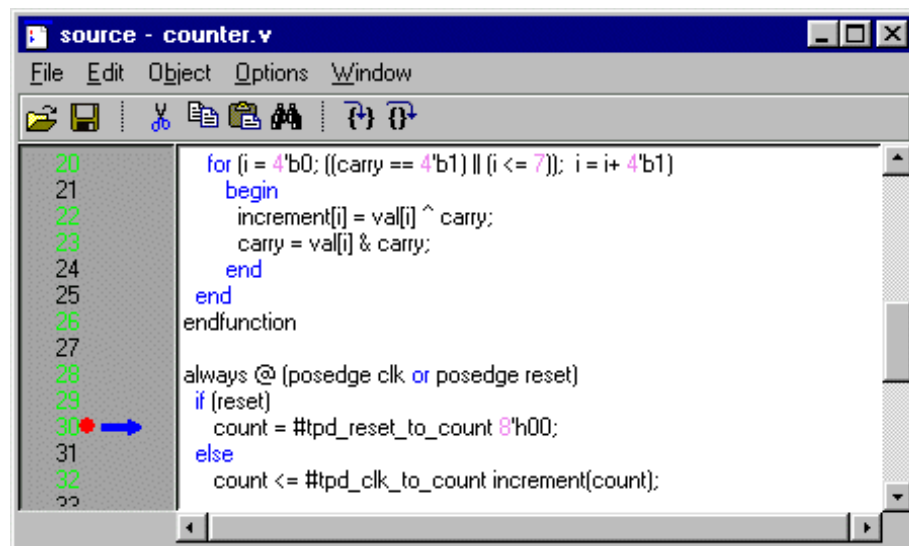
Select a display radix of **Decimal** for the signal **count**. Click **OK**. This causes the List window output to change; the count signal is now listed in decimal rather than the default binary.



- 2 Now let's set a breakpoint at line 30 in the *counter.v* file (which contains a call to the Verilog function increment). To do this, select **dut: counter** in the Structure pane of the Workspace. Move the cursor to the Source window and scroll the window to display line 30. Click on or near line number 30 to set a breakpoint. You should see a red dot next to the line number where the breakpoint is set.

The breakpoint can be toggled between enabled and disabled by clicking it. When a breakpoint is disabled, the circle appears open. To delete the breakpoint, click the line number with your right mouse button and select Remove Breakpoint.

- **Note:** Breakpoints can be set only on executable lines — denoted by green line numbers.

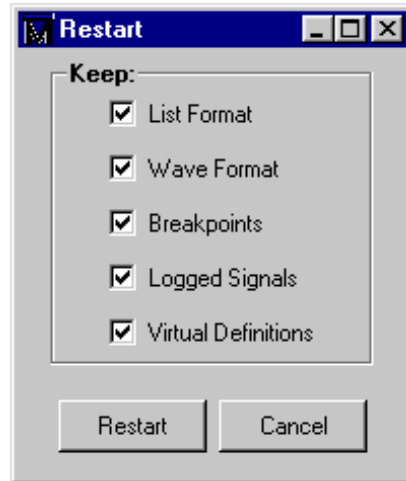


- 3 Select the **Restart** button to reload the design elements and reset the simulation time to zero.



(Main MENU: File > Restart) (PROMPT: restart)

Make sure all items in the Restart dialog box are selected, then click **Restart**.



- **Note:** The Verilog code in this example has a "stop" statement on line 19. If you resume the execution of the simulation without restarting first, you will stop at that line.

- 4 Select the **Run -all** button from the Main window toolbar to resume execution of the simulation.



(PROMPT: run -all) (Main MENU: Run > Run -All)

When the simulation hits the breakpoint, it stops running, highlights the line with an arrow in the Source window, and issues a Break message in the Main window.

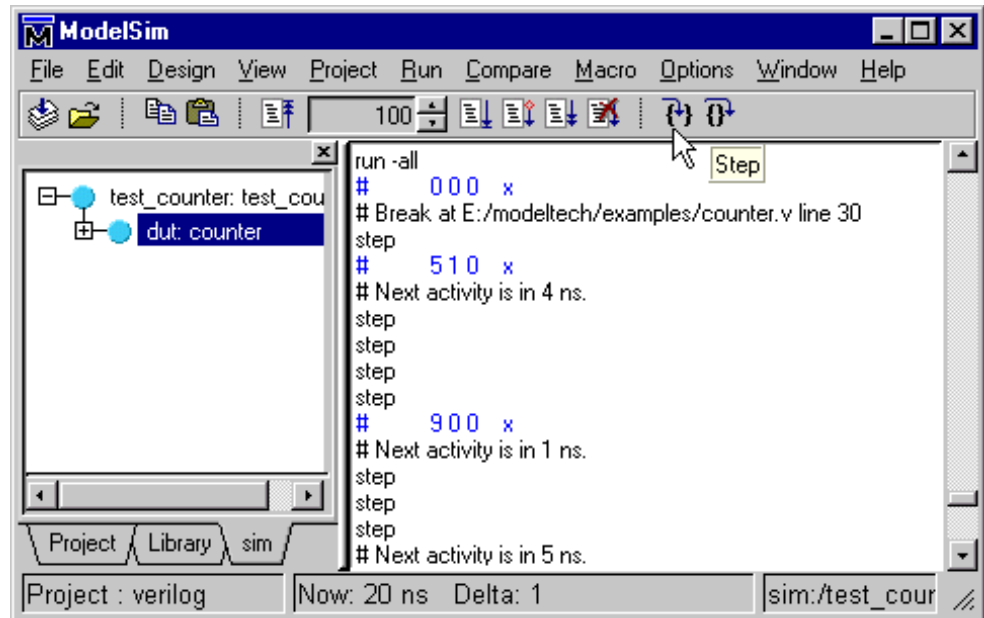
- 5 Typically when a breakpoint is reached you will be interested in one or more signal values. You have several options for checking values.

You can look at the values shown in the Signals window; you can move your mouse pointer over the *count* variable in the Source window and press the right mouse button; or you can use the **examine** command:

```
examine count
```

As a result of your command the count is output to the Main window.

- 6 Let's move through the Verilog source functions with ModelSim's Step command. Click **Step** on the toolbar.



The Step button on the toolbar single-steps the debugger.

- 7 Experiment by yourself for awhile. Set and clear breakpoints and use the Step and Step Over commands until you feel comfortable with their operation. When you're done, quit the simulator by entering the command:

```
quit -force
```



## Lesson 4 - Debugging a VHDL design

---

The goals for this lesson are:

- Show an example of a VHDL testbench - a VHDL architecture that instantiates the VHDL design units to be tested, provides simulation stimuli, and checks the results
- Map a logical library name to an actual library
- Change the default run length
- Recognize assertion messages in the command window
- Change the assertion break level
- Restart the simulation run using the **restart** command
- Examine composite types displayed in the Variables window
- Change the value of a variable
- Use a strobe to trigger lines in the List window
- Change the radix of signals displayed in the List window

## Preparing the simulation

- 1 Create a new directory for this exercise and copy the following VHDL (.vhd) files from `<install_dir>\modeltech\examples` to the new directory.
  - gates.vhd
  - adder.vhd
  - testadder.vhd
- 2 Make sure the new directory is the current directory. Do this by invoking *ModelSim* from the new directory or by using the **File > Change Directory** command from the *ModelSim* Main window.
- 3 Start *ModelSim* with one of the following:
 

**for Windows** - your option - from a Windows shortcut icon, from the Start menu, or from a DOS prompt:

```
modelsim.exe
```

Select "Proceed to ModelSim" if the Welcome dialog appears.
- 4 Enter the following command at the *ModelSim* prompt to create a new library:
 

```
vlib library_2
```
- 5 Compile the source files into the new library by entering this command at the *ModelSim* prompt:
 

```
vcom -work library_2 gates.vhd adder.vhd testadder.vhd
```
- 6 Now let's map the new library to the work library. To create a mapping you can edit the [Library] section of the *modelsim.ini* file, or you can create a logical library name with the **vmap** command:
 

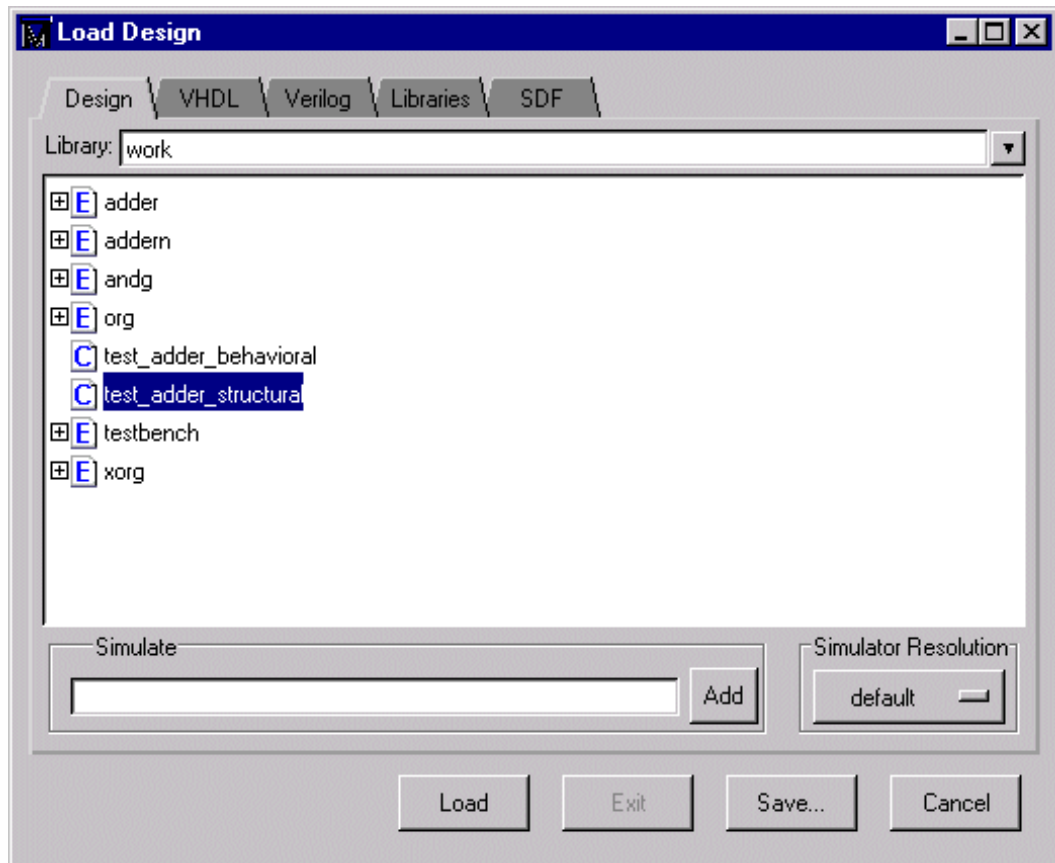
```
vmap work library_2
```

*ModelSim* modifies the *modelsim.ini* file for you.
- 7 Start the simulator by selecting **Design > Load Design** from the Main window, or by clicking the Load Design icon. The Load Design dialog box is displayed, as shown below.

8 Perform the following steps in the Load Design dialog box:

- Make sure that the simulator resolution is **default**.
- Look in the Design Unit scroll box and select the configuration named **test\_adder\_structural**.
- Click **Load** to accept the settings.

(PROMPT: vsim -t ns work.test\_adder\_structural)



- 9 To open all of the ModelSim windows, enter the following command in the Main window at the VSIM prompt:

```
view *
```

(Main MENU: View > All)

- 10 Drag and drop the top-level signals to the List window in the following manner: make sure the hierarchy is not expanded (no minus boxes), select all signals in the Signals window with **Edit > Select All**, then drag the selected signals to the List window.

(Signals MENU: View > List > Signals in Region) (PROMPT: add list \*)

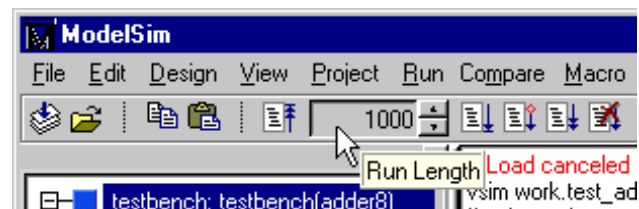
- 11 To add top-level signals to the Wave window, enter the command:

```
add wave *
```

(Signals MENU: View > Wave > Signals in Region) (DRAG&DROP)

- 12 Now change the default simulation run length to 1000 (ns) with the run length selector on the Main toolbar. Click on the field to edit the number to 1000 (notice how the arrows allow you to change the run length in increments).

(Main MENU: Options > Simulation > Defaults)





## Running and debugging the simulation

- 1 Now you will run the simulator. Select the **Run** button on the Main window toolbar.



(PROMPT: run)

A message in the Main window will notify you that there was an assertion error.

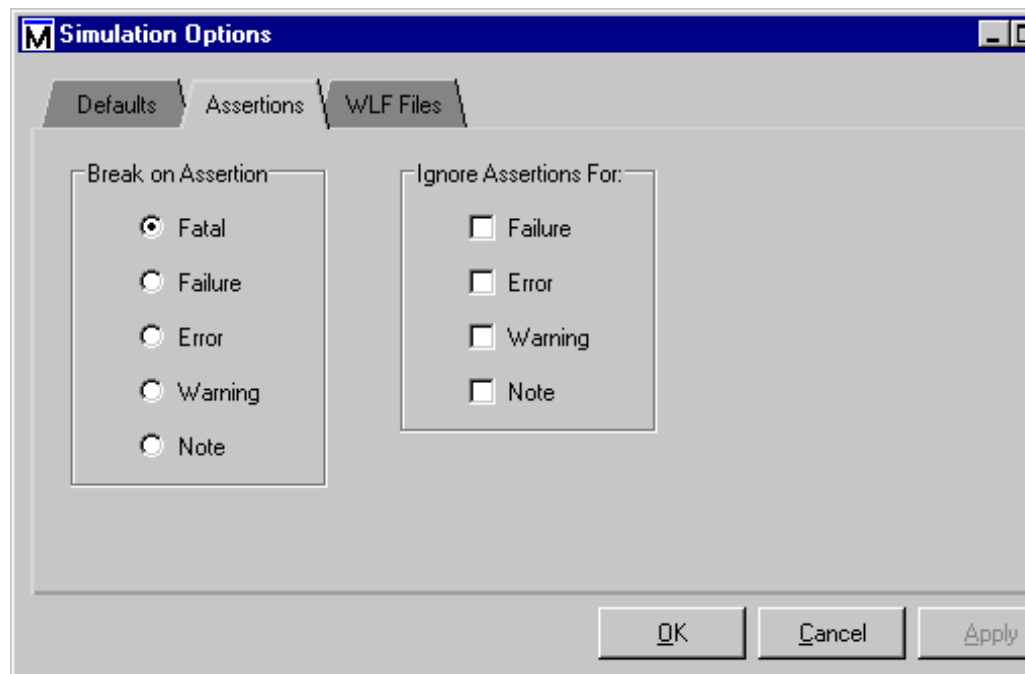
```
run
# ** Error: Sum is 00000111. Expected 00001000
#   Time: 600 ns Iteration: 0 Instance: /testbench
# ** Note: There were ERRORS in the test.
#   Time: 1 us Iteration: 0 Instance: /testbench

VSIM 10>
```

Now: 1 us Delta: 1 sim:/testbench

Let's find out what's wrong. Perform the following steps to track down the assertion message.

- 2 First, change the simulation assertion options. Select **Options > Simulation** from the Main window menu.

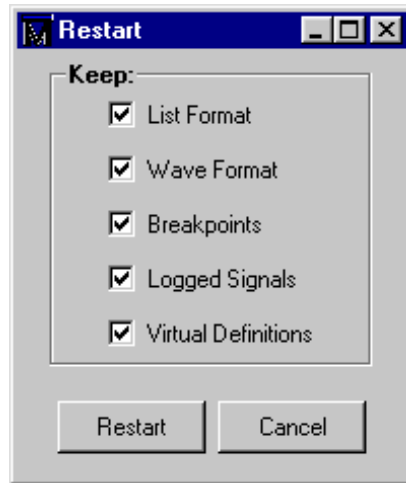


- 3 Select the **Assertions** page. Change the selection for **Break on Assertion** to **Error** and click **OK**. This will cause the simulator to stop at the HDL assertion statement.
- 4 To restart the simulation select the **Restart** button on the Main toolbar.



(Main MENU: File > Restart) (PROMPT: restart)

Make sure all items in the Restart dialog box are selected, then click **Restart**.

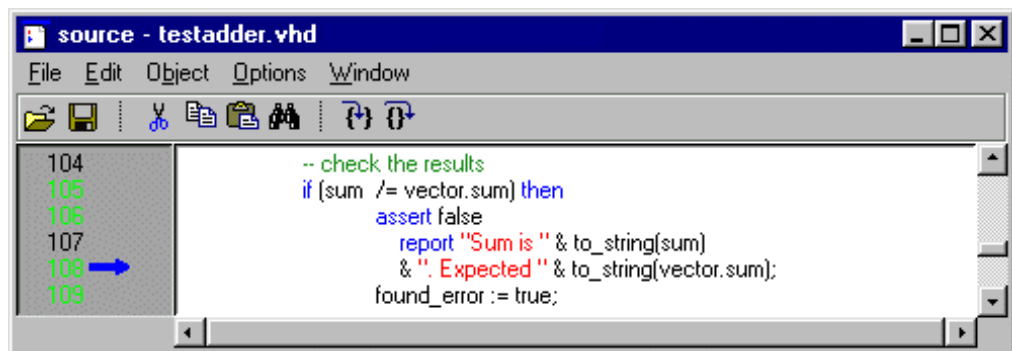


- 5 From the Main window toolbar select the **Run** button.

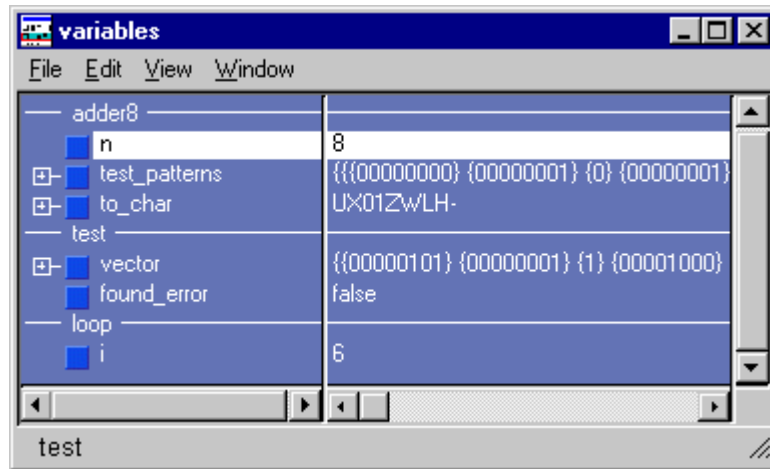


(Main MENU: Run > Run 1000 ns) (PROMPT: run)

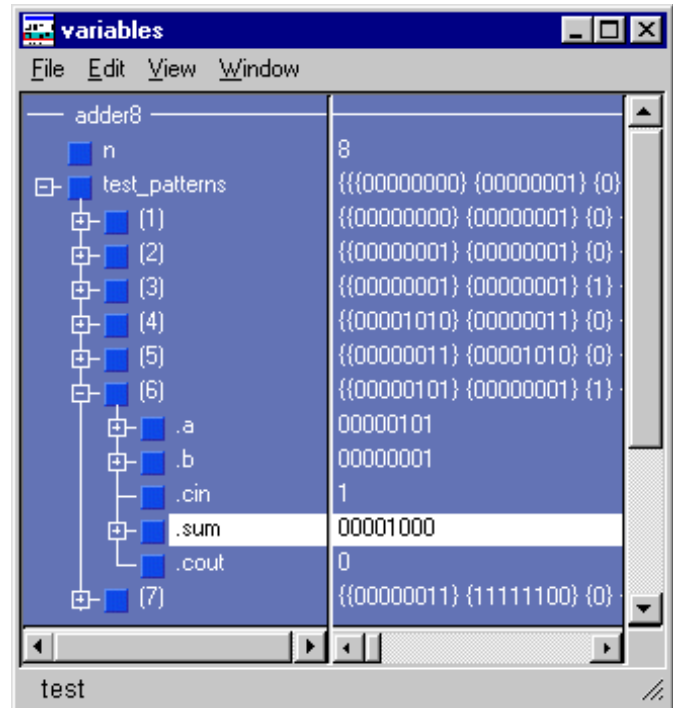
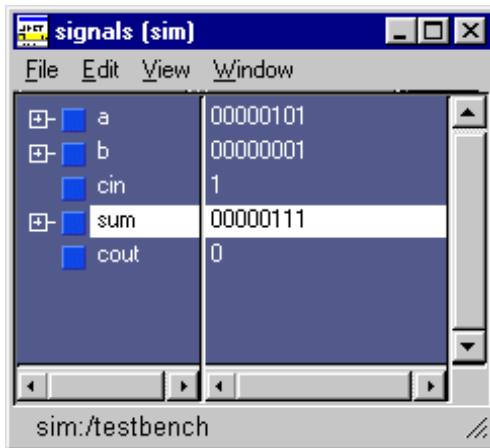
Notice that the arrow in the Source window is pointing to the assertion statement.



- 6 If you look at the Variables window now, you can see that  $i = 6$ . This indicates that the simulation stopped in the sixth iteration of the test pattern's loop.



- 7 Expand the variable named **test\_patterns** by clicking the [+]. (You may need to resize the window for a better view.)
- 8 Also expand the sixth record in the array **test\_patterns(6)**, by clicking the [+]. The Variables window should be similar to the one below.



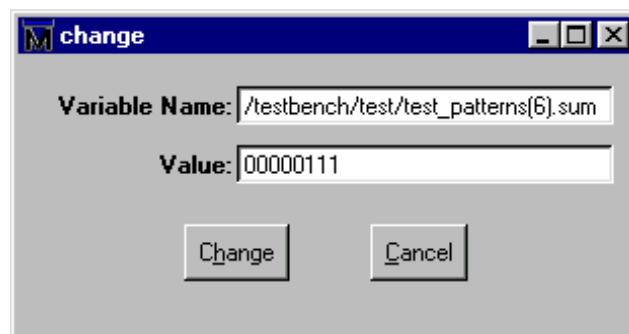
The assertion shows that the Signal **sum** does not equal the **sum** field in the Variables window. Note that the sum of the inputs **a**, **b**, and **cin** should be equal to the output **sum**. But there is an error in the test vectors. To correct this error, you need to restart the simulation and modify the initial value of the test vectors.

- 9 In the Main window, type:

```
restart -f
```

The **-f** option causes ModelSim to restart without popping up the confirmation dialog.

- 10 Update the Variables window by selecting the **testbench** process in the **test** Process window.
- 11 In the Variables window, expand **test\_patterns** and **test\_pattern(6)** again. Then highlight the **.sum** record by clicking on the variable name (not the box before the name) and then use the **Edit > Change** menu selection.



- 12 Select the last four bits (**1000**) in the value field by dragging the pointer across them. Then replace them with **0111**, and click **Change**. (Note that this is a temporary edit, you must use your text editor to permanently change the source code.)
- 13 Select the **Run** button from the Main window toolbar.



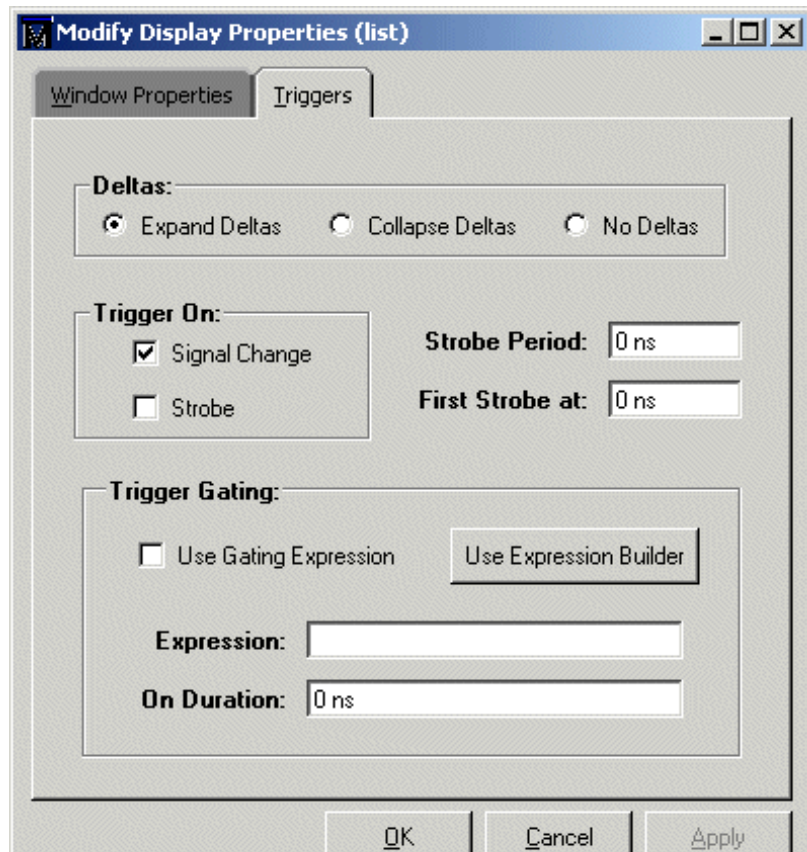
(Main MENU: Run > Run 1 us) (PROMPT: run)

At this point, the simulation will run without errors.

```
run
# ** Note: Test completed with no errors.
# Time: 1 us Iteration: 0 Instance: /testbench
VSIM 14>
Now: 1 us Delta: 1 Env: /testbench
```

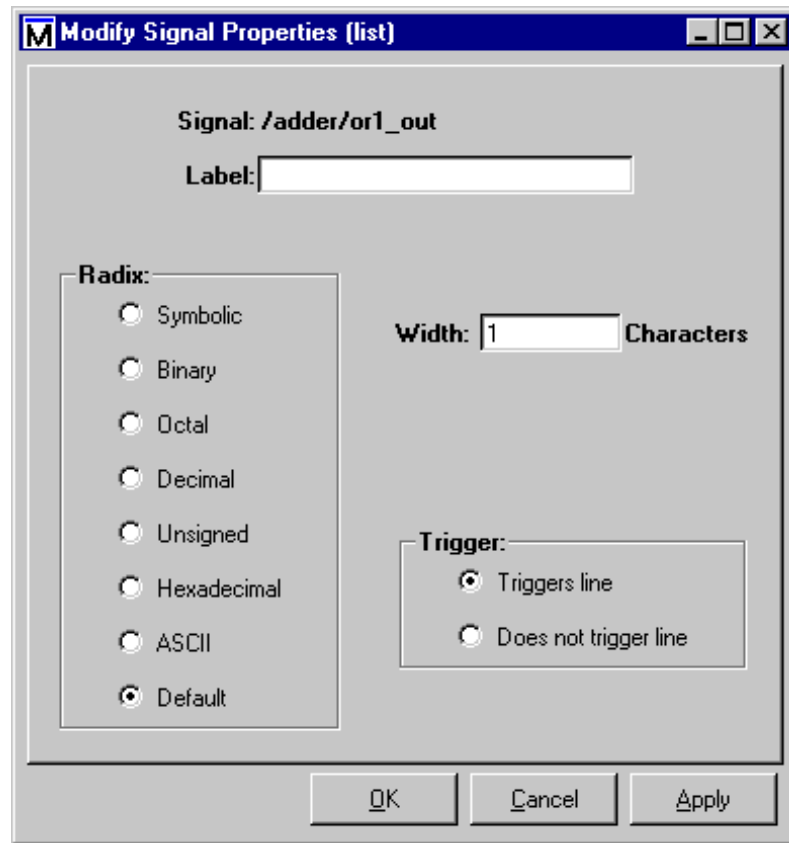
## Changing new-line triggering

By default, a new line is displayed in the List window for each transition of a listed signal. The following steps will change the triggering so the values are listed every 100 ns.



- 1 In the List window, select **Prop > Display Props**.
- 2 Perform these steps on the **Triggers** page:
  - Deselect **Trigger On: Signals** to disable triggering on signals.
  - Select **Trigger On: Strobe** to enable the strobe.
  - Enter **100** in the **Strobe Period** field.
  - Enter **70** in the **First Strobe at** field.
  - Click **OK** to accept the settings.

- 3 Your last action will be to change the radix to decimal for signals a, b, and sum. Select **Prop > Signal Props**. This opens the Modify Signal Properties (list) dialog box.



- 4 In the List window select the signal you want to change, then make the property changes in the dialog box. Make the following property changes:
- Select signal **a**, then click **Decimal**, then click **Apply**.
  - Select signal **b**, then click **Decimal**, then **Apply**.
  - Select signal **sum**, then click **Decimal**, then **OK**.

This brings you to the end of this lesson, but feel free to experiment further with the menu system. When you are ready to end the simulation session, quit *ModelSim* by entering the following command at the VSIM prompt:

```
quit -force
```

## Lesson 5 - Running a batch-mode simulation

---

The goals for this lesson are:

- Run a batch-mode VHDL simulation
- Execute a macro (DO) file
- View a saved simulation

Batch-mode allows you to execute several commands that are written in a text file. You create a text file with the list of commands you wish to run, and then specify that file when you start *ModelSim*. This is particularly useful when you need to run a simulation or a set of commands repeatedly.

▲ **Important:** Batch-mode simulations must be run from a DOS prompt. In Windows, you get a DOS prompt by selecting **Start > Programs > Command Prompt**. Unless directed otherwise, enter all commands in this lesson at a DOS prompt.

- 1 To set up for this lesson you'll need to create a new directory and make it the current directory. Copy this file into your new directory:

```
\<install_dir>\modeltech\examples\counter.vhd
```

- 2 Create a new design library (Remember, enter these commands at a DOS prompt):

```
vlib work
```

- 3 Map the library:

```
vmap work work
```

- 4 Then compile the source file:

```
vcom counter.vhd
```

- 5 You will use a macro file that provides stimulus for the counter. For your convenience, a macro file has been provided with ModelSim. You need to copy this macro file from the installation directory to the current directory:

```
<install_dir>\modeltech\examples\stim.do
```

- 6 Create a batch file using an editor; name it *yourfile*. With the editor, put the following on separate lines in the file:

```
add list -decimal *
do stim.do
write list counter.lst
```

and save to the current directory.

- 7 To run the batch-mode simulation, enter the following at the command prompt:

```
vsim -do yourfile -wlf saved.wlf counter
```

This is what you just did in Step 7:

- invoked the VSIM simulator on a design unit called "counter"
- instructed the simulator to save the simulation results in a log file named *saved.wlf* by using the **-wlf** switch
- used the contents of *yourfile* to specify that values are to be listed in decimal, to execute a stimulus file called *stim.do*, and to write the results to a file named *counter.lst*, the default for a design named counter

- 8 Since you saved the simulation results in *saved.wlf*, you can view the simulation results by starting up VSIM with its **-view** switch:

```
vsim -view saved.wlf
```



- 9 Open these windows with the **View** menu in the Main window, or the equivalent command at the ModelSim prompt:

```
view signals list wave
```

- **Note:** If you open the Process or Variables windows they will be empty. You are looking at a saved simulation, not examining one interactively; the logfile saved in *saved.wlf* was used to reconstruct the current windows.

- 10 Now that you have the windows open, put the signals in them:

```
add wave *  
add list *
```

- 11 Use the available windows to experiment with the saved simulation results and quit when you are ready:

```
quit -f
```

For additional information on the batch and command line modes, please refer to the *ModelSim User's Manual*.



## Lesson 6 - Executing commands at startup

---

The goals for this lesson are:

- Specify the design unit to be simulated on the command line
- Edit the *modelsim.ini* file
- Execute commands at startup with a DO file

▲ **Important:** Start this lesson from the DOS prompt in the same directory in which you completed *Chapter Lesson 5 - Running a batch-mode simulation*.

- 1 For this lesson, you will use a macro (DO) file that provides startup information. For convenience, a startup file has been provided with the ModelSim program. You need to copy this DO file from the installation directory to your current directory:

```
\<install_dir>\modeltech\examples\startup.do
```

- 2 Next, you will edit the modelsim.ini file in the \modeltech directory (or the modelsim.ini file in your current directory if one exists) to specify a command that is to be executed after the design is loaded. To do this, open

```
<install_dir>\modeltech\modelsim.ini
```

using a text editor and uncomment the following line (by deleting the leading ;) in the [vsim] section of the file:

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

Then save *modelsim.ini*.

▶ **Note:** The *modelsim.ini* file must be write-enabled for this change to take place. Using MS Explorer, right-click on \<install\_dir>\modeltech\modelsim.ini, then click Properties. In the dialog box, uncheck the Read-only box and click OK. (You can also copy the file to your current directory.)

- 3 Take a look at the DO file. It uses the predefined variable \$entity to do different things at startup for different designs.
- 4 Start the simulator and specify the top-level design unit to be simulated by entering the following command at the DOS prompt:

```
vsim counter
```

Notice that the simulator loads the design unit without displaying the Load Design dialog box. This is handy if you are simulating the same design unit over and over. Also notice that all the windows are open. This is because the **view \*** command is included in the startup macro.

- 5 If you plan to continue with the following practice sessions, keep ModelSim running. If you would like to quit the simulator, enter the following command at the VSIM prompt:

```
quit -f
```

- 6 You won't need the *startup.do* file for any other examples, so use your text editor to comment out the "Startup" line in *modelsim.ini*.

## Lesson 7 - Using the Wave window

---

The goals for this lesson are:

- Practice using the Wave window time cursors.
- Practice zooming the waveform display.
- Practice using Wave window keyboard shortcuts.
- Practice combining items into a virtual object.
- Practice creating and viewing datasets.

Any of the previous lesson simulations may be used with this practice, or use your own simulation if you wish.

## Using time cursors in the Wave window

When the Wave window is first drawn, there is one cursor located at time zero. Clicking anywhere in the waveform display brings that cursor to the mouse location.

These Wave window buttons give you quick access to cursor placement and zooming.

add cursor  
delete (selected) cursor  
find previous transition  
find next transition  
zoom in 2x  
zoom out 2x  
zoom area  
zoom full

The screenshot shows the ModelSim Wave window titled "wave - default". The menu bar includes File, Edit, Cursor, Zoom, Compare, Bookmark, Format, and Window. The toolbar contains various icons for file operations, editing, and zooming. A red circle highlights the "Cursor" and "Zoom" sections of the toolbar. The waveform display shows several signals: /top/clock, /top/prw, /top/pstrb, /top/prdy, /top/paddr, /top/pdata, /top/srw, /top/sstrb, and /top/srdy. The /top/pdata signal is highlighted with a blue box. The waveform is zoomed in, showing a time interval from 0 ns to 966 ns. A time cursor is positioned at 425 ns, and its value is displayed in a bold box. Other time measurements are shown: 52 ns, 373 ns, and 500 ns. The bottom status bar shows the time range "0 ns to 966 ns".



click a value here to scroll the window to that value

interval measurement

selected cursor is bold

Click and drag with the center mouse button to zoom in on an area of the display.

You can add up to 20 cursors to the waveform pane by selecting **Cursor > Add Cursor** (or the Add Cursor button shown below). The selected cursor is drawn as a bold solid line; all other cursors are drawn with thin dashed lines. Remove cursors by selecting them and choosing using the **Cursor > Delete Cursor** menu selection (or the Delete Cursor button shown below).

 <p><b>Add Cursor</b> add a cursor to the center of the waveform window</p>	 <p><b>Delete Cursor</b> delete the selected cursor from the window</p>
--	--

## Finding a cursor

The cursor value (on the **Goto** list) corresponds to the simulation time of that cursor. Choose a specific cursor view with **Cursor > Goto** menu selection.

## Making cursor measurements



Each cursor is displayed with a time box showing the precise simulation time at the bottom. When you have more than one cursor, each time box appears in a separate track at the bottom of the display. ModelSim also adds a delta measurement showing the time difference between two adjacent cursor positions.

If you click in the waveform display, the cursor closest to the mouse position is selected and then moved to the mouse position. Another way to position multiple cursors is to use the mouse in the time box tracks at the bottom of the display. Clicking anywhere in a track selects that cursor and brings it to the mouse position.

The cursors are designed to snap to the closest wave edge to the left on the waveform that the mouse pointer is positioned over. To modify the snap distance, select **Edit > Display Properties** (Wave window).

You can position a cursor without snapping by dragging in the area below the waveforms.

You can also move cursors to the next transition of a signal with these toolbar buttons:

 <p><b>Find Previous Transition</b> locate the previous signal value change for the selected signal</p>	 <p><b>Find Next Transition</b> locate the next signal value change for the selected signal</p>
--	--

## Zooming - changing the waveform display range

Zooming lets you change the simulation range in the waveform display. You can zoom with either the **Zoom** menu, toolbar buttons, mouse, keyboard, or commands.

### Using the Zoom menu

You can use the Wave window menu bar, or call up the **Zoom** menu by clicking the right mouse button (of a three-button mouse) in the waveform pane.

► **Note:** The right mouse button of a two-button mouse will not open the **Zoom** menu. It will, however, allow you to create a zoom area by dragging left to right while holding down the button.





The Zoom menu options include:

- **Zoom Full**  
Redraws the display to show the entire simulation from time 0 to the current simulation time.
- **Zoom In**  
Zooms in by a factor of two, increasing the resolution and decreasing the visible range horizontally.
- **Zoom Out**  
Zooms out by a factor of two, decreasing the resolution and increasing the visible range horizontally.
- **Zoom Last**  
Restores the display to where it was before the last zoom operation.
- **Zoom Area with Mouse Button 1**  
Use mouse button 1 to create a zoom area. Position the mouse cursor to the left side of the desired zoom interval, press mouse button 1 and drag to the right. Release when the box has expanded to the right side of the desired zoom interval.
- **Zoom Range**  
Brings up a dialog box that allows you to enter the beginning and ending times for a range of time units to be displayed.



## Zooming with the toolbar buttons

These zoom buttons are available on the toolbar:

	<b>Zoom in 2x</b> zoom in by a factor of two from the current view		<b>Zoom area</b> use the cursor to outline a zoom area
	<b>Zoom out 2x</b> zoom out by a factor of two from current view		<b>Zoom Full</b> zoom out to view the full range of the simulation from time 0 to the current time

## Zooming with the mouse

To zoom with the mouse, position the mouse cursor to the left side of the desired zoom interval, press the middle mouse button (three-button mouse), or right button (two-button mouse), and while continuing to press, drag to the right and then release at the right side of the desired zoom interval.

## Keyboard shortcuts for zooming

Using the following keys when the mouse cursor is within the Wave window will cause the indicated actions:

Key	Action
i I or +	zoom in
o O or -	zoom out
f or F	zoom full
l or L	zoom last
r or R	zoom range
<arrow up>	scroll waveform display up
<arrow down>	scroll waveform display down
<arrow left>	scroll waveform display left
<arrow right>	scroll waveform display right
<page up>	scroll waveform display up by page
<page down>	scroll waveform display down by page
<tab>	searches forward (right) to the next transition on the selected signal

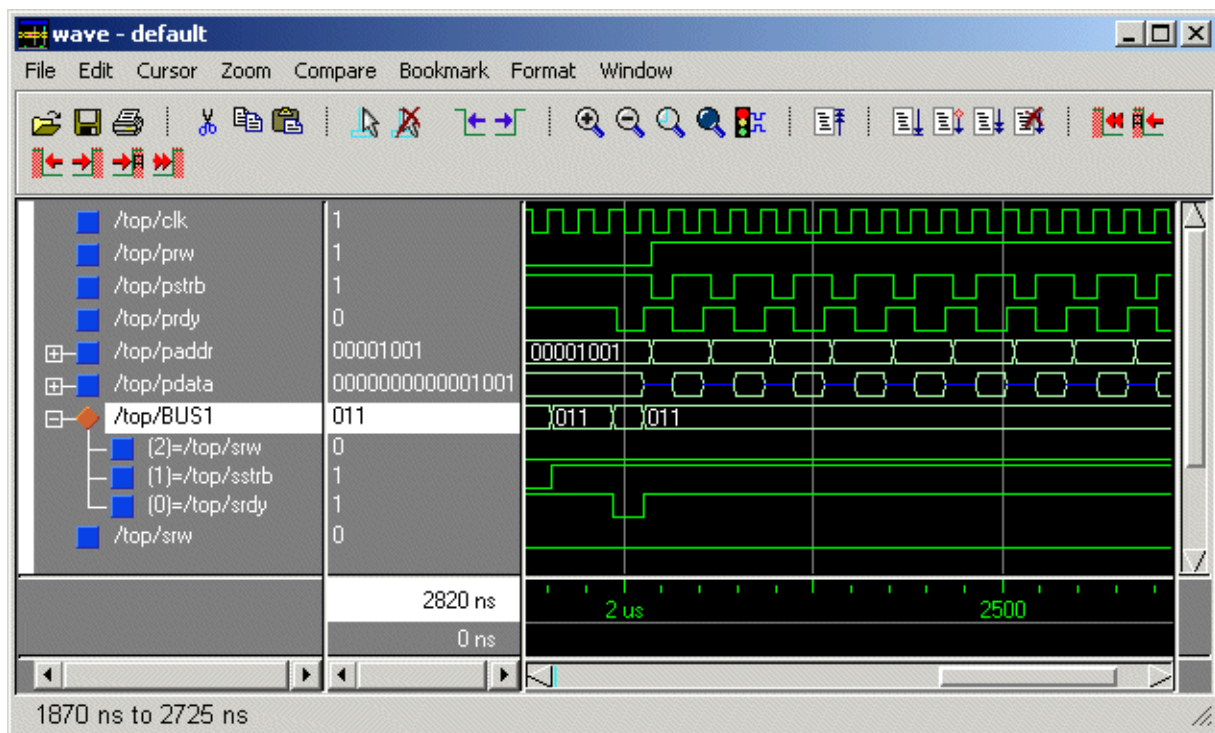
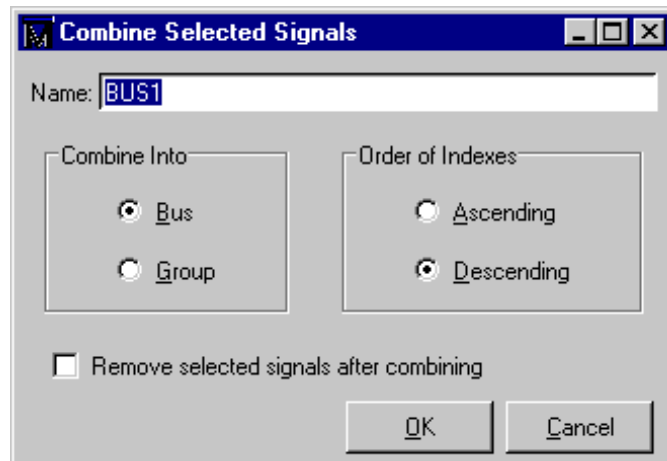
Key	Action
<shift-tab>	searches backward (left) to the previous transition on the selected signal
<Control-f>	opens the find dialog box; searches within the specified field in the pathname pane for text strings

## Combining items in the Wave window

The Wave window allows you to combine signals into buses or groups. Use the **Edit > Combine** menu selections to call up the Combine Selected Signals Dialog box.

A bus is a collection of signals concatenated in a specific order to create a new virtual signal with a specific value.

In the illustration below, three data signals have been combined to form a new bus called BUS1. Notice, the new bus has a value that is made up of the values of its component signals arranged in a specific order. Virtual objects are indicated by an orange diamond.



## Creating and viewing datasets

Datasets allow you to view previous simulations or to compare simulations. To view a dataset, you must first save a ModelSim simulation to a WLF file (using the **vsim -wlf** command). Once you have saved a WLF file, you can open it as a view-mode dataset.

In this lesson you will compare two simple Verilog designs: a structural description and an RTL description of a 4-bit, binary counter. To begin, you will simulate the structural description and save it to a WLF file. Then you will simulate the RTL version. Finally, you will open the WLF file as a dataset and compare the two simulations in the Wave window.

### Simulating the structural version

- 1 Start by creating a new working directory, making it the current directory, and copying the files from `\modeltech\examples\datasets` into it.

- 2 Use the **vlib** command to create a **work** library in the current directory.

```
vlib work
```

(MENU: Design > Create a New Library)

- 3 Use the **vmap** command to map the work library to a physical directory. A *modelsim.ini* file will be written into the **work** directory.

```
vmap work work
```

- 4 Compile the structural version of the counter.

```
vlog cntr_struct.v
```



(MENU: Design > Compile)

- 5 Load the design and save the simulation to a WLF file named *struct.wlf*.

```
vsim -wlf struct.wlf work.cntr_struct
```

- 6 Now you will run a DO file that applies stimulus to the design, runs the simulation, and adds waves to the Wave window.

```
do stimulus.do
```

(MENU: Macro > Execute Macro)

The waves that appear in the Wave window are saved automatically into the *struct.wlf* file.

- 7 Quit the simulation.

```
quit -sim
```

(MENU: Design > End Simulation)

## Simulating the RTL version

- 1 Compile the RTL version of the counter.

```
vlog cnt_rtl.v
```

- 2 Simulate the design.

```
vsim work.cnt_rtl
```



(MENU: Design > Load Design)

- 3 Run a DO file to apply stimulus to the design.

```
do stimulus.do
```

## Comparing the two designs

To compare the two simulations, we will create a second pane in the Wave window, open the *struct.wlf* file, and add the signals from the dataset to the new pane.

- 1 Add a second pane to the Wave window.

MENU: Wave > File > New Window Pane

Notice that a thick, white vertical bar at the left edge of the window indicates that the new pane is active. You probably want to increase the height of this new pane by pointing at the top border of the pane and clicking and dragging with the two-headed arrow.

- 2 Open *struct.wlf* (if you don't specify a dataset name, it will be named "struct" by default).

```
dataset open struct.wlf
```

(MENU: Wave > File > Open Dataset)

- 3 Add signals for the "struct" dataset.

```
add wave *
```

Notice that the pathname prefix for the signals you just added is the dataset name "struct". The pathname prefix for the active simulation is "sim".

The results for each simulation should be the same. You can continue experimenting with the two simulations or quit the simulator.

```
quit -f
```

(MENU: Main > File > Quit)



# Index

---

## A

Assertion errors [41](#)

## B

Batch-mode simulation [47](#)

Breakpoints [20](#)

continuing simulation after [21](#)

## C

Command history [7](#)

Compile

compile order of Verilog modules [25](#)

Verilog [24](#)

## D

Debugging a VHDL design [37](#)

Design library

create new [24](#)

creating [16](#)

do command [8](#)

DO files

executing a DO file in batch-mode [48](#)

using a DO file at startup [52](#)

using the transcript as a DO file [8](#)

drag and drop [7](#)

## E

Errors

breaking on assertion [42](#)

finding in VHDL designs [41](#)

viewing in Source window [42](#)

examine command [34](#)

## F

Finding

a cursor in the Wave window [55](#)

force command [20](#)

## H

Hierarchy

of a Verilog design [29](#)

## I

IEEE std 1076 [6](#)

IEEE std 1364 [6](#)

## K

Keyboard shortcuts, Wave window [57](#)

## L

Libraries

creation and mapping [38](#)

logical mapping [24](#)

List window

change display radix [32](#)

placing top level Verilog signals in [27](#)

Load design [18](#)

## M

Macros [8](#)

## Q

quit VSIM command [21](#), [35](#)

## R

restart [34](#), [42](#)

Reusing commands [8](#)

Run length selector

change run length [40](#)

run VSIM command [20](#)

## S

Searching

for HDL item names and transitions in the Wave window [56](#)

Shortcuts

command history [7](#)

Wave window [57](#)

Signal transitions

searching for [56](#)

## Signals

- add to List window [40](#)
- add to Wave window [40](#)
- applying stimulus to [20](#)
- display values with examine command [34](#)
- listing in region [19](#)
- placing top-level Verilog signals in the List and Wave window [27](#)
- specifying radix of [46](#)
- triggering listings for [45](#)

## Simulation

- batch-mode [47](#)
- executing commands at startup [51](#)
- Load Design dialog box [26](#)
- saving results in log file [48](#)
- single-stepping [21](#)
- starting [38](#)
- Verilog [23](#)
- view switch [48](#)
- wlf switch [48](#)

Standards supported [6](#)System initialization file [52](#)

## T

## Transcript

- save [8](#)

## Triggering

- changing in List window [45](#)
- modify [45](#)

## V

## Verilog

- compile [24](#)
- interface checking between design units [25](#)

Verilog simulation [23](#)

## W

## Wave window

- placing top level Verilog signals in [27](#)

## Windows

- viewing all [40](#)
- Wave window
  - changing display range (zoom) [56](#)
  - cursor measurements [55](#)
  - using time cursors [54](#)
  - zooming [56](#)

Work library mapping [38](#)

## Z

## Zoom

- from Wave toolbar buttons [57](#)
- from Zoom menu [56](#)
- with the mouse [57](#)

Zooming in the Wave window [56](#)