Getting Started with Model*Sim* EE

VHDL, Verilog, and Mixed-HDL Simulation
for Workstations

Model Technology
A MENTOR GRAPHICS COMPANY

Software Version: 5.1
Published: April 1998

Model Technology Incorporated
8905 SW Nimbus Avenue, Suite 150
Beaverton OR 97008-7100 USA

phone: 503-641-1340
fax: 503-526-5410
email: support@model.com, sales@model.com
home page: http://www.model.com

$50 US

Software License Agreement

This is a legal agreement between you, the end user, and Model Technology Incorporated (MTI). By opening the sealed package, or by signing this form, you are agreeing to be bound by the terms of this agreement. If you do not agree to the terms of this agreement, promptly return the unopened package and all accompanying items to the place you obtained them for a full refund.

Model Technology Software License

1. LICENSE. MTI grants to you the **nontransferable, nonexclusive** right to use one copy of the enclosed software program (the "SOFTWARE") for each license or hardware security key you have purchased. The SOFTWARE must be used on the computer hardware server equipment that you identified in writing by make, model, and workstation or host identification number and the equipment served, in machine-readable form only , as allowed by the authorization code provided to you by MTI or its agents. All authorized systems must be used within the country for which the systems were sold. Workstation licenses must be located at a single site, i.e. within a one-kilometer radius (and identified in writing to MTI). The restriction to identified products does not apply to PC products licensed by a hardware security key, and such PC products may be relocated within the country for which sold.

2. COPYRIGHT. The SOFTWARE is owned by MTI (or its licensors) and is protected by United States copyright laws and international treaty provisions. Therefore you must treat the SOFTWARE like any other copyrighted material, except that you may either (a) make one copy of the SOFTWARE solely for backup or archival purposes, or (b) transfer the SOFTWARE to a single hard disk provided you keep the original solely for backup or archival purposes. You may not copy the written materials accompanying the SOFTWARE.

3. USE OF SOFTWARE. The SOFTWARE is licensed to you for internal use only. You shall not conduct benchmarks or other evaluations of the SOFTWARE without the advance written consent of an authorized representative of MTI. You shall not sub-license, assign or otherwise transfer the license granted or the rights under it without the prior written consent of MTI or its applicable licensor. You shall keep the SOFTWARE in a restricted and secured area and shall grant access only to authorized persons. You shall not make software available in any form to any person other than your employees whose job performance requires access and who are specified in writing to MTI. MTI may enter your business premises during normal business hours to inspect the SOFTWARE, subject to your normal security.

4. PERMISSION TO COPY LICENSED SOFTWARE. You may copy the SOFTWARE only as reasonably necessary to support an authorized use. Except as permitted by Section 2, you may not make copies, in whole or in part, of the SOFTWARE or other material provided by MTI without the prior written consent of MTI. For such permitted copies, you will include all notices and legends embedded in the SOFTWARE and affixed to its medium and container as received from MTI. All copies of the SOFTWARE, whether provided by MTI or made by you, shall remain the property of MTI or its licensors.

You will maintain a record of the number and location of all copies of the SOFTWARE made, including copes that have been merged with other software, and will make those records available to MTI or its applicable licensor upon request.

5. TRADE SECRET. The source code of the SOFTWARE is trade secret or confidential information of MTI or its licensors. You shall take appropriate action to protect the confidentiality of the SOFTWARE and to ensure that any user permitted access to the SOFTWARE does not provide it to others. You shall take appropriate action to protect the confidentiality of the source code of the SOFTWARE. You shall not reverse-assemble, reverse-compile or otherwise reverse-engineer the SOFTWARE in whole or in part. The provisions of this section shall survive the termination of this Agreement.

6. TITLE. Title to the SOFTWARE licensed to you or copies thereof are retained by MTI or third parties from whom MTI has obtained a licensing right.

7. OTHER RESTRICTIONS. You may not rent or lease the SOFTWARE. You shall not mortgage, pledge or encumber the SOFTWARE in any way. You shall ensure that all support service is performed by MTI or its designated agents. You shall notify MTI of any loss of the SOFTWARE.

8. TERMINATION. MTI may terminate this Agreement, or any license granted under it, in the event of breach or default by you. In the event of such termination, all applicable SOFTWARE shall be returned to MTI or destroyed.

9. EXPORT. You agree not to allow the MTI SOFTWARE to be sent or used in any other country except in compliance with this license and applicable U.S. laws and regulations. If you need advice on export laws and regulations, you should contact the U.S. Department of Commerce, Export Division, Washington, DC 20230, USA for clarification.

Table of Contents

1 - Introduction

Chapter contents

The purpose of this guide is to help you install and become familiar with Model*Sim* EE, Model Technology's HDL-simulation software for workstations. Tutorials are included that apply to one or more of the following Model*Sim* simulators:

- **Model*Sim* EE/PLUS**
  VHDL, Verilog, and mixed VDHL/Verilog simulation for workstations
- **Model*Sim* EE/VHDL**
  VHDL simulation for workstations
- **Model*Sim* EE/VLOG**
  Verilog simulation for workstations

Software versions

This documentation was written to support Model*Sim* EE/PLUS 5.1 for workstations. If the software you are using is a later release, check the README file that accompanied the software. Any supplemental information will be there.

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

Model*Sim*’s graphic interface

While your workstation interface provides the window-management frame, Model*Sim* controls all internal-window features including menus, buttons, and scroll bars. The resulting interface remains consistent within these workstation environments:

- SPARCstation with OpenWindows or OSF/Motif
- IBM RISC System/6000 with OSF/Motif
- Hewlett-Packard HP 9000 Series 700 with HP VUE or OSF/Motif

For an in-depth look at Model*Sim*’s graphic interface see, "ModelSim EE Graphic Interface" (p27).

Standards supported

Model*Sim* VHDL supports both the IEEE 1076-1987 and 1076-1993 VHDL standards. Any design developed with Model*Sim* will be compatible with any other VHDL system that is compliant with either IEEE Standard 1076-1987 or 1076-1993.

Model*Sim* Verilog is based on the IEEE Std 1364-1995 (*IEEE 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. The PLI is supported on PCs and workstations, while VCD support is available for workstation users only.

In addition, all products support SDF 1.0, 2.0, and 2.1, VITAL 2.2b, and VITAL ’95.

Assumptions

We assume that you are familiar with the use of your operating system. You should be familiar with the window management functions of the interface on your workstation: either OpenWindows or OSF/Motif on SPARCstations and OSF/Motif on IBM and Hewlett-Packard workstations.

In addition, we assume that you have a working knowledge of VHDL and Verilog. Although Model*Sim* is an excellent tool to use while learning HDL concepts and practices, this guide is not written to support that goal. If you need more information about HDLs, visit the Model Technology home page at http://www.model.com. You can also contact your local Model Technology distributor for information about training classes.

Sections in this guide

In addition to this introduction, you will find the following major sections in this guide:

Text conventions

Text conventions used in this manual include:

italic text	provides emphasis and sets off file and path names
bold text	indicates commands, command options, and menu choices, as well as package and library logical names
`monospaced type`	monospace type is used for program and command examples
The right angle (>)	is used to connect menu choices when traversing menus as in: **File > Save**

HDL and HDL item defined

"HDL" refers to either VHDL or Verilog when a specific language reference is not needed. Depending on the context, "HDL item" can refer to any of the following:

- **VHDL**
  block statement, component instantiation, constant, generate statement, generic, package, signal, or variable
- **Verilog**
  function, module instantiation, named fork, named begin, net, task, or register variable

Syntax conventions

The syntax elements of Model*Sim* commands are signified as follows:

< >	angled brackets surrounding a syntax item indicate a user-defined argument; do not enter the brackets in commands
[ ]	square brackets indicate an optional item; if the brackets surround several words, all must be entered as a group; the brackets are not entered
`...`	an ellipsis indicates items that may appear more than once; the ellipsis itself does not appear in commands.
|	the vertical bar indicates a choice between items on either side of it. Do not include the bar in the command
#	comments are preceded by the number sign (#)

Where to find our documentation

Model Technology's documentation is available in the following formats and locations:

Document	Format	How to get it
Getting Started (installation & tutorial)	paper	shipped with the product; additional copies at $50 each (for customers with current maintenance)
Getting Started EE	PDF online	find "gs_ee<version>.pdf" in the "<install_dir>/modeltech/docs" directory; current version available from the Support page of our web site: http://www.model.com
EE Reference Manual	paper	shipped with the product; additional copies at $50 each (for customers with current maintenance)
EE Reference Manual	PDF online	find "ee_man<version>.pdf" in "<install_dir>/modeltech/docs"; current version available for ftp from the Support page of our web site: http://www.model.com (password required)
Tcl man pages	HTML	find "contents.html" in "<install_dir>/modeltech/docs/html/", or use the Main window menu selection: Help > Tcl Man Pages
various tech notes	ASCII	find the "technotes" directory in "<install_dir>/modeltech/docs/technotes"

Download a free PDF reader

Model Technology's online documentation (in PDF format) requires a free Adobe Acrobat reader available through http://www.adobe.com.

Comments

Comments and questions about this manual and Model*Sim* software are welcome. Call, write, or fax or email:

Model Technology Incorporated
8905 SW Nimbus Avenue, Suite 150
Beaverton OR 97008-7100 USA

phone: 503-641-1340
fax: 503-526-5410

email: manuals@model.com
home page: http://www.model.com

2 - Model*Sim* EE Installation

Chapter contents

This chapter covers installation of Model*Sim* EE for workstations.

Please refer to the installation instructions packaged with your product for the latest information; *they may supercede the installation instructions here*. Also please be sure to read the *README* file included with Model*Sim* after you finish your installation.

The Model*Sim README* file includes additional information for QuickHDL users.

System requirements for Model*Sim* EE

	SPARCstation	IBM RISC/6000	HP 700
Operating system & interface	SunOS 4.1 & OpenWindows 3.0 Solaris 2.4 & OSF/ Motif 1.2 or OpenWindows 3.0	AIX 3.2.5 & OSF/Motif 1.2	HP-UX 9.01 & HP VUE 3.0 or OSF/Motif 1.2
Memory	32 Mb minimum		
Storage	hard disk with at least 35 Mb spare capacity		
Installation media	1/4” tape, QIC 24 format, CD-ROM		4 mm DAT, CD-ROM

Installation procedure

Before you begin a new Model*Sim* EE installation, make sure you have a Model*Sim* Authorization Certificate with the *license.dat* file; you need the file to run Model*Sim* . If you don’t have the certificate or the license file, you can get one by email (license@model.com) or fax from Model Technology.

Be sure to include your workstation’s identification in your fax or email:

Syntax	Platform
hostid	SPARC
/etc/lanscan	HP 700
uname -m	RISC/6000

The commands used to install Model*Sim* are case-sensitive, so they must be entered exactly as shown in the following steps. (Note that neither the prompt nor the <Return> at the end of a line is shown in the examples.)

Step 1.

Model*Sim* can be installed in any directory. These instructions assume that you are installing it in the */usr* directory. You can change to the desired installation directory by entering one of the following commands:

```
cd /usr
```

or

```
cd <pathname>
```

Step 2.

For CD-ROM installation, extract the file for your platform (plus *base.tar* and *docs.tar)* with this syntax (we assume that the *<CD-ROM>* is mounted):

```
    tar xf /<CD-ROM>/WORKSTATION/install/<CD-ROM filename>
```

For tape installation, extract the files using the tape install command for your platform.

Platform	<CD-ROM filename>	Tape install commands
All platforms	base.tar (extract for all platforms)	
Sun OS-4/Solaris 1.x	sun4.tar	tar xf /dev/rst0
Sun OS-5/Solaris 2.x	sunos5.tar	tar xf /dev/rmt/0
HP	hp700.tar	tar xf /dev/rmt/0m
IBM	rs6000.tar	tar xf /dev/<cartridge_drive>
All platforms	docs.tar (manuals in pdf format)	

This creates a top-level directory named *modeltech*.

Step 3.

Add the pathname of the Model*Sim* executable directory to your search path:

```
/usr/modeltech/sun4
/usr/modeltech/sunos5
/usr/modeltech/hp700
/usr/modeltech/rs6000
```

You can also use the **vco** command within your shell script to return the platform directory.

- for csh:
  ```
  set path = ( $path /<install_dir>/‘vco‘ )
  ```
- for sh (Bourne or Korn)
  ```
  PATH=$PATH:/<install_dir>/‘vco‘ export PATH
  ```

Step 4.
Enter or edit the *license.dat* file in the *modeltech* directory. Typically the file looks like this:

```
SERVER hostname nnnnnnnn 1650
DAEMON modeltech ./modeltech ./options
FEATURE vsim modeltech 1997.090 dd-mmm-yyyy 2 6C92577EC335F4C9568D ck=61
FEATURE vcom modeltech 1997.090 dd-mmm-yyyy 2 6C92577EC335F4C9568D ck=61
```

Note:
To enable starting the license daemon from any directory, change the DAEMON line to use full path. For example:

```
DAEMON modeltech /usr/modeltech/hp700/modeltech/usr/modeltech/hp700/options
```

Step 5. For HP installation using HP-UX version 9.x only: either the device /dev/lan0 must be writable by the userID that starts the server, or the lmgrd daemon and the lmhostid utility must be made “setuid root”. The commands to do this are:

```
chmod a+w /dev/lan0
```

or

```
chown root lmgrd lmhostid
chmod u+s lmgrd lmhostid
```

Step 6.
Start the license manager daemon by entering the following commands:

```
cd /usr/modeltech/<platform>
START_SERVER
```

where <platform> can be sun4, sunos5, hp700, rs6000.

If your system runs other applications that use Globetrotter Software’s FLEXlm, a complete user’s manual for FLEXlm is available at Globetrotter Software’s home page:

http://www.globetrotter.com/manual.htm

Step 7.

You may delete the executables you don’t want; for example:

```
cd /usr/modeltech/
rm -rf hp700
```

Installed directories and files

A typical installation would have the files and structure illustrated below. For details on platform subdirectories (hp700, rs6000, sun4, and sun5) see "Platform specific directories" (p25).

Directory	Files & subdirectories	Description
<install directory> /modeltech	convert.tcl, modelsim.ini, README, README.HTML, rebuild_libs.csh, rebuild_libs.sh, vsim.ps	translates 4.6 do files into 5.1 do files, modelsim initialization file, modelsim readme files in text and HTML formats, recreates all libraries from the source files (for csh and sh shells), Postscript header file
./arithmetic	std_logic_arith	Mentor Graphics specific arithmetic packages
./bin	qhcvt, dumplog64, hm_entity, qhdel, qhdir, qhlib, qhmake, qhmap, qhsim, qvhcom, qvlcom, sm_entity, vcom, vdel, vdir, vlib, vlog, vmake, vmap, vsim	platform independent executables and QuickHDL mappings to Model*Sim* executables
./docs	technotes, verilog, HTML, ee_get<version>.pdf, ee_man<version>.pdf, sdk_um.pdf,	general Model*Sim* technotes, Verilog specific technotes, HTML versions of the Tcl/Tk man pages, "Getting Started with Model*Sim*" tutorial in PDF format, EE/PLUS reeference manual in PDF format, MGC standard developer's kit manual in PDF format
./examples	various	example simulation models, testbenches, macros, and utilities

Directory	Files & subdirectories	Description
./examples/foreign	various	example file directory for foreign interface
./examples/mixedhdl	various	example file directory for mixed VHDL/Verilog design
./examples/tcl_getstart	various	example files for Tcl/Tk tutorial
./ieee	math_real, math_complex, numeric_bit, numeric_extra, numeric_signed, numeric_std, numeric_unsigned, std_logic_1164, std_logic_1164_extensions, std_logic_arith, std_logic_misc, std_logic_signed, std_logic_textio, std_logic_unsigned, vital_primitives, vital_timing	library for accelerated IEEE and Synopsys arithmetic packages
./ieeepure	std_logic_1164, vital_primitives, vital_timing	standard VHDL IEEE library
./include	acc_user.h, acc_vhdl.h, mti.h, tcl.h, veriuser.h	include files for use with Verilog PLI, VHDL foreign interface, and TCL
./mgc_portable	qsim_logic, qsim_relations, qsim_tc	Mentor Graphics QuickSim compatible logic set
./std_developerskit	std_iopak, std_mempak, std_regpak, std_simflags, std_timing	libraries for MGC standard developer's kit
./synopsys	arithmetic, attributes, types	accelerated arithmetic packages
./vhdl_src/arithmetic	std_arit.vhd	source for Mentor Graphics arithmetic libraries
./vhdl_src/ieee	mti_numeric_bit.vhd, mti_numeric_std.vhd, stdlogic.vhd	sources for rebuilding basic IEEE std_logic_1164 library and accelerated IEEE arithmetic packages

Directory	Files & subdirectories	Description
./vhdl_src/mentor	numeric_extra.vhd, numeric_signed.vhd, numeric_unsigned.vhd, qsim_logic.vhd, qsim_relations.vhd, qsim_tc.vhd, std_logic_1164_ext.vhd	source for Mentor Graphics QuickHDL add-ons
./vhdl_src/std	standard.vhd, textio.vhd	sources for VHDL STD library and package TEXTIO
./vhdl_src/std_devloperskit	examples, iopakb.vhd, iopakp.vhd, mempakb.vhd, mempakp.vhd, regpakb.vhd, regpakp.vhd, simflagb.vhd, simflagp.vhd, synthreg.vhd, timingb.vhd, timingp.vhd	sources and examples for MGC standard developer's kit
./vhdl_src/synopsys	mti_std_logic_arith.vhd, mti_std_logic_misc.vhd, mti_std_logic_signed.vhd, mti_std_logic_unsigned.vhd, std_logic_textio.vhd, syn_ari.vhd, syn_attributes.vhd, syn_type.vhd	sources for rebuilding accelerated arithmetic packages
./vhdl_src/verilog	vltypes.vhd	source for rebuilding Verilog library
./vhdl_src/vital2.2b	prmtvs_b.vhd, prmtvs_p.vhd, timing_b.vhd, timing_p.vhd	sources for rebuilding VITAL version 2.2b library
./vhdl_src/vital95	prmtvs_b.vhd, prmtvs_p.vhd, timing_b.vhd, timing_p.vhd	sources for rebuilding VITAL version 95 library

Platform specific directories

The four platform-specific directories within the *modeltech* directory (*hp700*, *rs6000*, *sun4*, and *sun5*) include the same files, except as noted below.

Directory	Files & subdirectories	Description
<platform>	START_SERVER	license manager script
	libsm.sl, libswiftpli.sl	SmartModel support libraries
	lmcksum, lmdown, lmgrd, lmhostid, lmremove, lmreread, lmstat, lmswitchr, modeltech	license manager executables
	options	license manager executable
	sm_entity	Model*Sim* tool for use with LMG models
	tssi2mti	Model*Sim* tool for use with SEF files
	vcom	Model*Sim* VHDL compiler
	vdel, vdir, vmake, vmap	Model*Sim* library management tools
	vlog	Model*Sim* Verilog compiler
	vish	Tcl/TK user interface
	vlm	simulation license manager
	vgencomp	Model*Sim* tool for use with Verilog modules
	vsim	Model*Sim* simulator
./rs6000	mti_exports	list of foreign interface routines exported from mti
./sun4	vsim.swift	Model*Sim* simulator for use with LMG models

3 - Model*Sim* EE Graphic Interface

Chapter contents

This chapter describes the graphic interface available while operating VSIM, the Model*Sim* simulator. Model*Sim*'s graphic interface is designed to provide consistency throughout all workstation environments. Your workstation interface provides the window-management frame; Model*Sim* controls all internal-window features including menus, buttons, and scroll bars.

Because its graphic interface is based on Tcl/Tk, Model*Sim* allows you to create your own simulation environment. Easily accessible preference variables, and configuration commands give you control over the use and placement of windows, menus, menu options and buttons.

Command-line simulation

Although this chapter deals primarily with the use of Model*Sim* via its graphic interface, VSIM also provides a set of commands that can be used to run your simulations from the command line. These commands are described in the *ModelSim Reference Manual*.

Window features

Model*Sim*'s graphic interface provides many features that add to its usability; features common to many of the windows are described below.

Feature	Feature applies to these windows
Drag and Drop (p28)	Dataflow, List, Signals, Source, Structure, Variables, and Wave windows
Automatic window updating (p29)	Dataflow, Process, Signals, and Structure
Finding names, and searching for values (p29)	various windows
Sorting HDL items (p30)	Process, Signals, Source, Structure, Variables and Wave windows
Multiple window copies (p30)	all windows except the VSIM Main window
Menu tear off (p30)	all windows
Tree window hierarchical view (p31)	all windows
Tree window hierarchical view (p31)	Structure, Signals, Variables, and Wave windows

Drag and Drop

Drag and drop of HDL items is possible between the following windows. Using the left mouse button, click and release to select an item, then click and hold to drag it.

- **Drag items from these windows:**
  Dataflow, List, Signals, Source, Structure, Variables, and Wave windows

- **Drop items in these windows:**
  List and Wave windows

Note that drag and drop works to move items *within* the List and Wave windows as well.

Automatic window updating

Selecting an item in the following windows automatically updates other related Model*Sim* windows as indicated below:

Select an item in this window	To update these windows
Dataflow window (p42) (with a process selected in the center of the window)	Process window (p62)
	Signals window (p64)
	Source window (p70)
	Structure window (p74)
	Variables window (p77)
Process window (p62)	Dataflow window (p42)
	Signals window (p64)
	Structure window (p74)
	Variables window (p77)
Signals window (p64)	Dataflow window (p42)
Structure window (p74)	Signals window (p64)
	Source window (p70)

Finding names, and searching for values

- **Find** HDL item names with the **Edit > Find** menu selection in these windows:
  List, Process, Signals, Source, Structure, Variables, and Wave windows.
- **Search** for HDL item values with the **Edit > Search** menu selection in these windows:
  List, and Wave windows.

You can also:

- **Locate** time markers in the List window with the **Markers > Goto** menu selection.
- **Locate** time cursors in the Wave window with the **Cursor > Goto** menu selection.

In addition to the menu selections above, the virtual event **<<Find>>** is defined for all windows. The default binding is to **<Key-F19>** in most windows. You can bind **<<Find>>** to other events with the Tcl/Tk command **event add**. For example,

```
event add <<Find>> <Control-Key-F>
```

Sorting HDL items

Use the **Edit > Sort** menu selection in the windows below to sort HDL items in ascending, descending or declaration order.

Process, Signals, Source, Structure, Variables and Wave windows

Names such as net_1, net_10, and net_2 will sort numerically in the Signals and Wave windows.

Multiple window copies

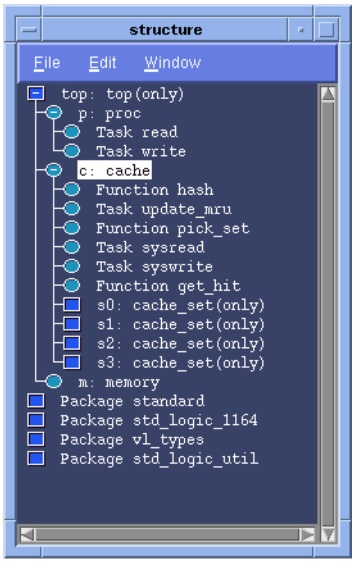
Use the **View > New** menu selection from the VSIM Main window (p35) to create multiple copies of the same window type. The new window will become the default window for that type.

Menu tear off

All window menus may be "torn off " to create a separate menu window. To tear off, click on the menu, then select the dotted-line button at the top of the menu.

Tree window hierarchical view

Model*Sim* provides a hierarchical, or "tree view" of some aspect of your design in the Structure, Signals, Variables, and Wave windows.



```
top: top(only)
  p: proc
    Task read
    Task write
  c: cache
    Function hash
    Task update_mru
    Function pick_set
    Task sysread
    Task syswrite
    Function get_hit
    s0: cache_set(only)
    s1: cache_set(only)
    s2: cache_set(only)
    s3: cache_set(only)
  m: memory
Package standard
Package std_logic_1164
Package vl_types
Package std_logic_util
```

Depending on which window you are viewing, one entry is created for each of the following VHDL and Verilog HDL item within the design:

VHDL items (indicated by a "box" prefix)	**Verilog items** (indicated by a "circle" prefix)
signals, variables, component instantiation, generate statement, block statement, and package	parameters, registers, nets, module instantiation, named fork, named begin, task, and function

Viewing the hierarchy

Whenever you see a tree view, as in the Structure window above, you can use the mouse to collapse or expand the hierarchy. Select the symbols as shown below.

Symbol	Description
[+]	A plus box/circle indicates that you can expand this item to view the structural elements it contains. Do this by clicking the box/circle containing the plus sign.
[-]	A minus box/circle indicates that the hierarchy has been expanded. You can hide the names of structural elements in the region by clicking on the box or circle containing the minus sign.
[ ]	An empty box/circle indicates that the item contains no lower-level structural elements.

Tree window action list

Action	Use	Do
expand a level	left mouse button	click on a "+" box/circle
collapse a level	left mouse button	click on a "-" box/circle
select a single item	left mouse button	click on the HDL item name (not the box/circle prefix)
select multiple contiguous items (not in Structure window)	left mouse button	click on the HDL item name and drag to complete selection
select a range of contiguous items (not in Structure window)	shift + left mouse button	select first item, shift/click on last item in range
select multiple random items (not in Structure window)	control + left mouse button	click on the desired items in any order
move an item	left mouse button	click on an item, then reselect it and drag to reposition

Finding items within tree windows

You can open the find dialog box within all windows (except the VSIM Main, and Source windows) by using this keyboard shortcut:

<Control-f>

Options within the Find dialog box allow you to search unique text-string fields within the specific window. See also,

- "Finding items by name in the List window" (p56),
- "Finding HDL items in the Signals window" (p69), and
- "Finding items by name or value in the Wave window" (p89).

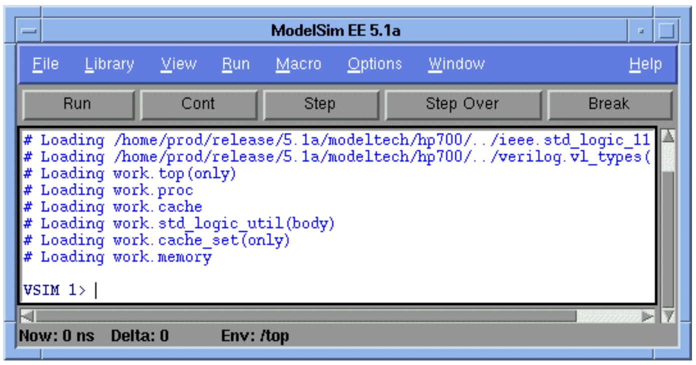
Window overview

Once you start a simulation - whether invoked directly by a VSIM command from the UNIX command line or through the Startup dialog box - nine windows become available for use during simulation. These windows are:

- VSIM Main window (p35)
  The main window from which all subsequent VSIM windows are available.
- Dataflow window (p42)
  Lets you trace signals and nets through your design by showing related processes.
- List window (p44)
  Shows the simulation values of selected VHDL signals, and Verilog nets and register variables in tabular format.
- Process window (p62)
  Displays a list of processes that are scheduled to run during the current simulation cycle.
- Signals window (p64)
  Shows the names and current values of VHDL signals, and Verilog nets and register variables in the region currently selected in the Structure window.
- Source window (p70)
  Displays the HDL source code for the design.
- Structure window (p74)
  Displays the hierarchy of structural elements such as VHDL component instances, packages, blocks, generate statements, and Verilog model instances, named blocks, tasks and functions.
- Variables window (p77)
  Displays VHDL constants, generics, variables, and Verilog register variables in the current process and their current values.
- Wave window (p79)
  Displays waveforms, and current values for the VHDL signals, and Verilog nets and register variables you have selected.

VSIM Main window

The VSIM Main window is pictured below as it appears when VSIM is first invoked from the UNIX command line. Note that your workstation graphic interface provides the window-management frame only; Model*Sim* handles all internal-window features including menus, buttons, and scroll bars.



```
# Loading /home/prod/release/5.1a/modeltech/hp700/../ieee.std_logic_11
# Loading /home/prod/release/5.1a/modeltech/hp700/../verilog.vl_types(
# Loading work.top(only)
# Loading work.proc
# Loading work.cache
# Loading work.std_logic_util(body)
# Loading work.cache_set(only)
# Loading work.memory

VSIM 1> |
```

Now: 0 ns Delta: 0 Env: /top

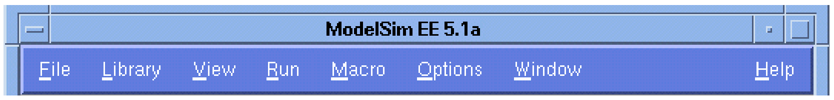
The menu bar at the top of the window provides access to a wide variety of simulation commands and Model*Sim* preferences. The status bar at the bottom of the window gives you information about the data in the active Model*Sim* window. The tool bar (under the menu bar) gives you access to the most common **run** commands no matter which VSIM window is active.

When a simulation is running, the VSIM Main window displays a VSIM prompt, allowing you to enter command-line commands from within the graphic interface. Messages output by VSIM during simulation are also displayed in this window. You can scroll backward and forward through the current work history by using the vertical scrollbar. You can also copy and paste using the mouse within the window, see "Editing the command line, the current source file, and notepads" (p40).

The VSIM Main window menu bar, tool bar, and status bar are detailed below.

The VSIM Main window menu bar

The menu bar at the top of the VSIM Main window lets you access many Model*Sim* commands and features. The menus are listed below with brief descriptions of the command's use.



File menu

Load New Design	start a new simulation via the Load Design dialog box
Restart	restart the current simulation from time zero; a dialog box provides options to keep the List and Wave formats, breakpoints, and logged signals
Save Transcript	save a transcript of the Main window to a text file; CAUTION! this can be a very large file
Save Transcript as...	save another copy of the transcript with a different name (the default name is specified in the modelsim.ini file
Clear Transcript	clear the Main window transcript display
Quit	quit Model*Sim* and return to UNIX command line

Library menu

Browse Libraries	browse all libraries within the scope of the design
Create a New Library	create a new library or map a library to a new name
View Library Contents	view or delete the contents of a library

View menu

All	open all VSIM windows
Source	open and/or view the Source window (p70)

Structure	open and/or view the Structure window (p74)
Variables	open and/or view the Variables window (p77)
Signals	open and/or view the Signals window (p64)
List	open and/or view the List window (p44)
Process	open and/or view the Process window (p62)
Wave	open and/or view the Wave window (p79)
Dataflow	open and/or view the Dataflow window (p42)
New	create a new VSIM window of the specified type

Run menu

Run default	run simulation for one default run length; change the run length with Options > Simulation...
Run -All	run simulation until you stop it
Continue	continue the simulation
Run -Next	run to the next event time
Step	single-step the simulator
Step-Over	execute without single-stepping through a subprogram call

Macro menu

Execute Macro	allows you to browse for and execute a do file (macro)
Macro Helper	invokes the Macro Helper tool
Tcl Debugger	invokes the Tcl debugger, TDebug

Options menu

Simulation ...	returns the Simulation Options dialog box; options include: default radix, default force type, default run length, iteration limit, warning suppression, and break on assertion specification
Edit Preferences...	returns the Preferences dialog box; color preferences can be set for window background, text and graphic items (i.e., waves in the Wave window)
Save Preferences	save current Model*Sim* settings to a Tcl preference file

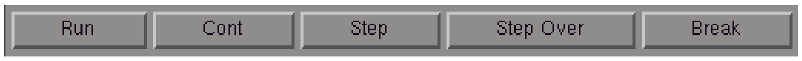
Window menu

<window_name>	lists the currently open windows; select a window name to switch to, or show that window if it is hidden; when the source window is available, the source file name is also indicated; open additional windows from the "View menu" (p36)
Icon Children	iconify all windows except the Main window
Icon All	iconify all windows
Deicon All	restores all windows to their most current open size and position

Help menu

About ModelSim	display Model*Sim* application information
Release Notes	view current release notes
Information about Help	view the readme file pertaining to Model*Sim*'s online documentation
Getting Started with ModelSim	open and read *Getting Started* Acrobat (.pdf) file; PDF files can be read with a free Adode Acrobat reader available through www.adobe.com
ModelSim EE/PLUS Reference Manual	open and read the *EE Reference Manual* Acrobat (.pdf) file; .pdf files can be read with a free Adode Acrobat reader available through www.adobe.com
Tcl Man Pages	open and read Tcl 7.6/Tk 4.2 manual in HTML format (uses a Tcl/Tk HTML viewer)
Technotes	select a technical note to view from the drop-down list

The Main window tool bar

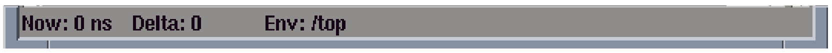


In addition to the functions accessed through the menu bar, you can use the Main window tool bar for several frequently-used **run** commands:

Button	Menu equivalent	VSIM command equivalent
Run	Run > Run <default_run_length>…	**run**
Cont	Run > Cont….	**run -continue**
Step	Run > Step….	**step**
Step Over	Run > Step Over….	**step -over**
Break	none	*

*
You can execute a break from the parent UNIX window with: <control-c>.

The Main window status bar



Fields at the bottom of the VSIM Main window provide the following information about the current simulation:

Field	Description
Now	the current simulation time or a larger time unit if one can be used without a fractional remainder
Delta	the current simulation iteration number

Field	Description
Env	name of the current environment (item selected in the Structure window (p74))

Editing the command line, the current source file, and notepads

The following mouse actions and special keystrokes can be used to edit commands in the entry region of the VSIM Main window. They can also be used in editing the file displayed in the Source window (p70) and all **notepad** windows (enter the notepad command at the VSIM prompt to open the notepad editor). (The mouse operations are of Motif origin and the special keystrokes are of EMACS origin.)

Mouse action	Result
< left-button - click >	move the insertion cursor
< left-button - press > + drag	select
< shift - left-button - press >	extend selection
< left-button - double-click >	select word
< left-button - double-click > + drag	select word + word
< control - left-button - click >	move insertion cursor without changing the selection
< middle-button - click >	paste clipboard
< middle-button - press > + drag	scroll the window

Special keystrokes	Result
< left | right | up | down - arrow >	move the insertion cursor
< control - a >	move insertion cursor to beginning of line
< control - e >	move insertion cursor to end of line
< * meta - "<" >	move insertion cursor to beginning of file

Special keystrokes	**Result**
< * meta - ">" >	move insertion cursor to end of file
< control - p >	move insertion cursor to previous line
< control - n >	move insertion cursor to next line
< control - f >	move insertion cursor forward
< control - b >	move insertion cursor backward
< backspace >	delete character to the left
< control - d >	delete character to the right
< control - k >	delete to the end of line
< control - w >	cut selection
< *meta - w >	copy selection
< control - y >	insert clipboard

The VSIM Main window allows insertions or pastes only after the prompt, therefore, you don't need to set the cursor when copying strings to the command line. Also, when doing <control-a> to move the insertion to the beginning of the line, the insertion point will show left of the cursor, but insertions will happen to the right of the cursor.

*

Which keyboard key functions as the meta key depends on how your X-windows KeySym mapping is set up. You may need help from your system administrator to map a particular key, such as the <alt> key, to the meta KeySym.

Dataflow window

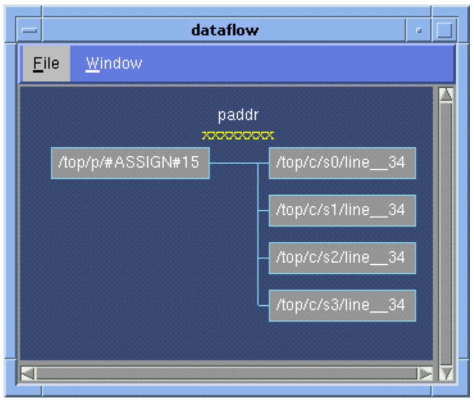
The Dataflow window allows you to trace VHDL signals or Verilog nets through your design.

VHDL signals or processes:

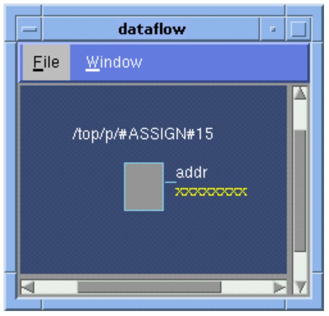
- Shows a signal in the center of the window with all the processes that drive the signal on the left, and all the processes that read the signal on the right, or
- shows a process with all the signals read by the process shown as inputs on the left of the window, and all the signals driven by the process on the right.

Verilog nets or processes:

- Shows a net in the center of the window with all the processes that drive the net on the left, and all the processes triggered by the net on the right, or
- shows a process with all the nets that trigger the process shown as inputs on the left of the window, and all the nets driven by the process on the right.



signal or net "paddr"



process "#ASSIGN#15"

The Dataflow window menu bar

The following menu commands and button options are available from the Dataflow window menu bar.

File menu

Save Postscript	save the current dataflow view as a Postscript file
Selection	Follow Selection updates window when the Process window (p62) or Signals window (p64) changes; Fix Selection freezes the view selected from within the Dataflow window
Close	close this copy of the Dataflow window; you can create a new window with View > New from the "The VSIM Main window menu bar" (p36)

Window menu

<VSIM_window>	switch to the selected (open) <VSIM_window> or show if hidden

Tracing HDL items with the Dataflow window

The Dataflow window is linked with the Signals window (p64) and the Process window (p62). To examine a particular process in the Dataflow window, click on the process name in the Process window. To examine a particular HDL item in the Dataflow window, click on the item name in the Signals window.

with a signal in center of the Dataflow window, you can:

- click once on a process name in the Dataflow window to make the Source and Variable windows update to show that process,
- click twice on a process name in the Dataflow window to move the process to the center of the Dataflow window

with a process in center of the Dataflow window, you can:

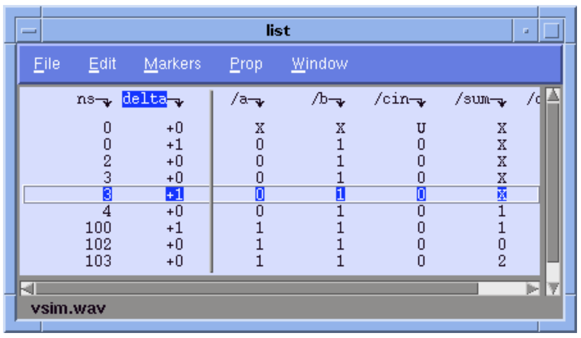
- click once on an item name to make the Source and Signals windows update to show that item,

- click twice on an item name to move that item to the center of the Dataflow window.

The Dataflow window will display the current process when you single-step or when VSIM hits a breakpoint.

List window

The List window displays the results of your simulation run in tabular format. The window is divided into two adjustable panes, which allows you to scroll horizontally through the listing on the right, while keeping time and delta visible on the left.



```
     ns  delta   /a     /b    /cin   /sum   /c
      0     +0    X      X      U      X
      0     +1    0      1      0      X
      2     +0    0      1      0      X
      3     +0    0      1      0      X
      3     +1    0      1      0      X
      4     +0    0      1      0      1
    100     +1    1      1      0      1
    102     +0    1      1      0      0
    103     +0    1      1      0      2
```

HDL items viewed within the List window

The following HDL items for VHDL and Verilog are viewable in the List window.

VHDL items	Verilog items
signals and process variables	nets and register variables

Note:

Constants, generics, parameters, and memories are not viewable in the List or Wave windows.

List window action list

This action list provides a quick reference to menu selections and mouse actions in the List window. See the "Tree window action list" (p32) for additional information.

Action	Menu or mouse	See also
place specific HDL items in the List window	Signals window (p64) menu: View > List (choose Selected Signals, Signals in Region, or Signals in Design) drag and drop: from the Process, Signals, or Signals window	"Adding HDL items to the List window" (p50)
move or delete items already in the List window	menu selection: Edit > (select Cut, Copy, Paste, Delete, Combine, Select All, or Unselect All) drag and drop: within the List window	"Adding HDL items to the List window" (p50)
set display properties such as name length, trigger on options, delta views, and strobe timing	menu selection: Prop > Display Props	"Setting List window display properties" (p47)
format an item's radix, label, width, and triggering properties	menu selection: Prop > Signal Props	"Editing and formatting HDL items in the List window" (p52)
save your listing to an ASCII file	menu selection: File > Write List (select tabular, events or TSSI format)	"Saving List window data to a file" (p60)
save your List window configuration for future use	menu selection: File > Save Format	"Adding HDL items to the List window" (p50)
reuse a List window configuration	menu selection: File > Load Format	"Adding HDL items to the List window" (p50)

Action	Menu or mouse	See also
finding an HDL item by name	menu selection: Edit > Find	"Finding items by name in the List window" (p56)
finding the value of an HDL item	menu selection: Edit > Search	"Searching for item values in the List window" (p56)
set, delete or go to a time marker in the listing	menu selection: Markers > (choose Add Marker, Delete Marker, or Goto)	"Setting time markers in the List window" (p59)

The List window menu bar

The following menu commands and button options are available from the List window menu bar.

File menu

Write List (format)	save the listing as a text file in one of three formats: tabular, events, or TSSI
Load Format	run a List window format do file previously saved with Save Format
Save Format	saves the current List window display and signal preferences to a do (macro) file; running the do file will reformat the List window to match the display as it appeared when the do file was created
Close	close this copy of the List window; you can create a new window with View > New from the "The VSIM Main window menu bar" (p36)

Edit menu

Cut	cut the selected item field from the listing; see "Editing and formatting HDL items in the List window" (p52)
Copy	copy the selected item field
Paste	paste the previously cut or copied item to the left of the currently selected item
Delete	delete the selected item field

Combine	combine the selected fields into a user-defined bus; keep copies of the original items rather than moving them
Find...	find specified item label within the List window
Search...	search the List window for a specified value, or the next transition for the selected signal

Markers menu

Add Marker	add a time marker at the top of the listing page
Delete Marker	delete the selected marker from the listing
Goto	choose the time marker to go to from a list of current markers

Prop menu

Display Props	set display properties for all items in the window: delta settings, trigger on selection, strobe period, and label size
Signal Props	set label, radix, trigger on/off, and field width for the selected item

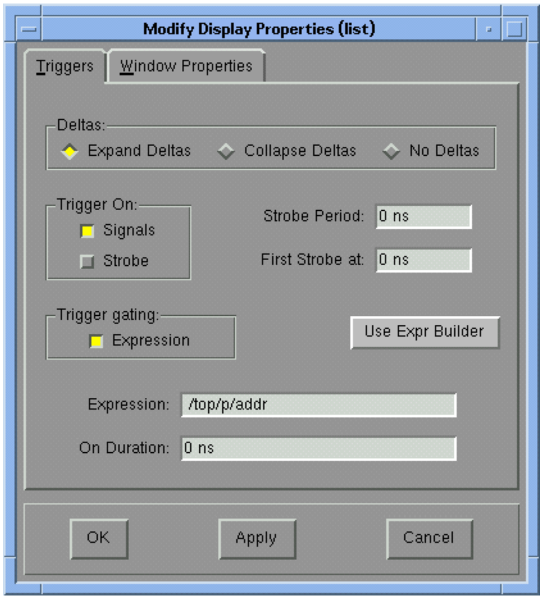
Window menu

<VSIM_window>	switch to the selected (open) <VSIM_window> or show if hidden

Setting List window display properties

Before you add items to the List window you can set the window's display properties. To change when and how a signal is displayed in the List window, make this selection from the List window menu bar: **Prop > Display Props**. The resulting Modify Display Properties dialog box has the following options.

Trigger settings page



Strobe Period: 0 ns
First Strobe at: 0 ns
Expression: /top/p/addr
On Duration: 0 ns

The Triggers page controls the triggering for the display of new lines in the List window. You can specify whether an HDL item trigger or a strobe trigger is used to determine when the List window displays a new line. If you choose **Trigger on: Signals**, then you can choose between collapsed or expanded delta displays. You can also choose a combination of signal or strobe triggers. To use gating, Signals or Strobe or both must be selected.

The Triggers page includes these options:

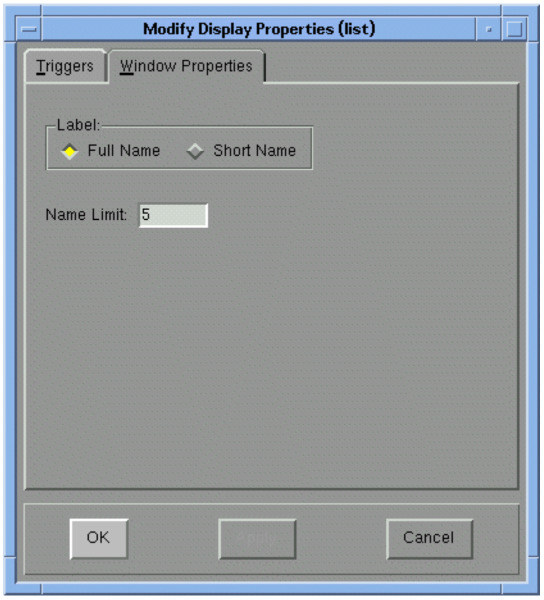
- **Deltas:Expand Deltas**
  When selected with the **Trigger on: Signals** check box, displays a new line for each time step on which items change, including deltas within a single unit of time resolution.
- **Deltas:Collapse Deltas**
  Displays only the final value for each time unit in the List window.
- **Deltas:No Deltas**
  No simulation cycle (delta) column is displayed in the List window.

- **Trigger On: Signals**
  Triggers on signal changes. Defaults to all signals. Individual signals may be excluded from triggering by using the **Prop > Signals Props** dialog box.
- **Trigger On: Strobe**
  Triggers on the **Strobe Period** you specify; specify the first strobe with **First Strobe at:**.
- **Trigger Gating: Expression**
  Enables triggers to be gated on and off by an overriding expression, much like a hardware signal analyzer might be set up to start recording data on a specified setup of address bits and clock edges. Affects the display of data, not the acquisition of the data.
- **Use Expression Builder** (button)
  Opens the Expression Builder to help you write a gating expression.
- **Expression**
  Enter the expression for trigger gating into this field, or use the Expression Builder (select the Use Expression Builder button). The expression is evaluated when the List window would normally have displayed a row of data (given the trigger on signals and strobe settings above).
- **On Duration**
  The duration for gating to remain open after the last list row in which the expression evaluates to true; expressed in x number of default timescale units. Gating is level-sensitive rather than edge-triggered.

List window gating information is saved as configuration statements when the list format is saved. The gating portion of a configuration statement might look like this:

```
.list.tbl config -usegating 1
.list.tbl config -gateduration 100
.list.tbl config -gateexpr {<expression>}
```

Window Properties page



Label: [✓ Full Name] Short Name

Name Limit: 5

The **Window Properties** page includes these options:

- **Label: Full Name**
  Display the full pathname of the item.
- **Label: Short Name**
  Display the item name without the path.
- **Name Limit**
  The maximum number of number of rows in the name pane.

Adding HDL items to the List window

Before adding items to the List window you may want to set the window display properties (see "Setting List window display properties" (p47)). You can add items to the List window in several ways.

Adding items with drag and drop

You can drag and drop items into the List window from the Process, Signals, or Structure window. Select the items in the first window, then drop them into the List window. Depending on what you select, all items or any portion of the design may be added. See the "Tree window action list" (p32) for information about making item selections.

Adding items from the Main window command line

Invoke the **add list** command to add one or more individual items; separate the names with a space:

```
add list <item_name> <item_name>
```

You can add all the items in the current region with this command:

```
add list *
```

Or add all the items in the design with:

```
add list -r /*
```

Adding items with a List window format file

To use a List window format file you must first save a format file for the design you are simulating.

- add the items you want in the List window with any other method shown above
- edit and format the items, see "Editing and formatting HDL items in the List window" (p52) to create the view you want
- save the format to a file with the List window menu selection: **File > Save Format**

To use the format file, start with a blank List window and run the do file in one of two ways:

- use the **do** command on the command line:
  ```
  do <my_list_format>
  ```
- select **File > Load Format** from the List window menu bar

Use **Edit > Select All** and **Edit > Delete** to remove the items from the current List window or create a new, blank List window with the **View > New > List** selection from the "VSIM Main window" (p35). You may find it useful to have two differently formatted windows open at the same time, see "Examining simulation results with the List window" (p55).

Note:
List window format files are design-specific; use them only with the design you were simulating when they were created. If you try to the wrong format file, Model*Sim* will advise you of the HDL items it expects to find.

Editing and formatting HDL items in the List window

Once you have the HDL items you want in the List window, you can edit and format the list to create the view you find most useful. (See also, "Adding HDL items to the List window" (p50))

To edit an item:

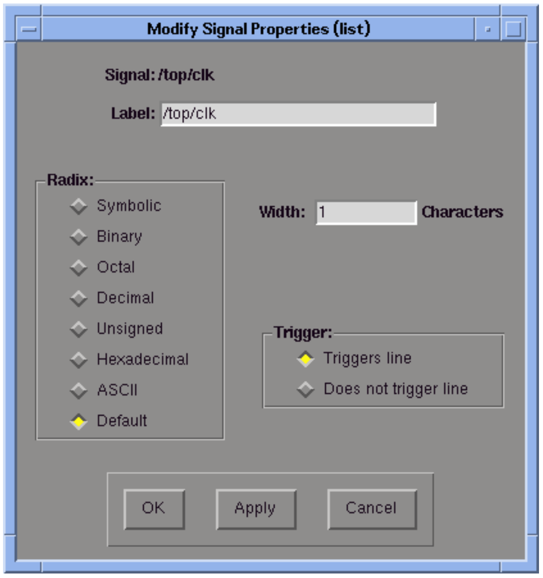
Select the item's label at the top of the List window or one of its values from the listing. Move, copy or remove the item by selecting commands from the List window Edit menu (p46) menu.

You can also click+drag to move items within the window:

- to select several contiguous items:
  click+drag to select additional items to the right or the left of the original selection
- to select several items randomly:
  Control+click to add or subtract from the selected group
- to move the selected items:
  re-click on one of the selected items and drag to the new location

To format an item:

Select the item's label at the top of the List window or one of its values from the listing, then use the **Prop > Signal Props** menu selection. The resulting Modify Signal Properties dialog box allows you to set the item's label, label width, triggering, and radix.



The **Modify Signal Properties** dialog box includes these options:

- **Signal**
  Shows the item you selected with the mouse.
- **Label**
  Allows you to specify the label that is to appear at the top of the List window column for the specified item.
- **Radix**
  Allows you to specify the radix (base) in which the item value is expressed. The default radix is symbolic, which means that for an enumerated type, the List window lists the actual values of the enumerated type of that item.

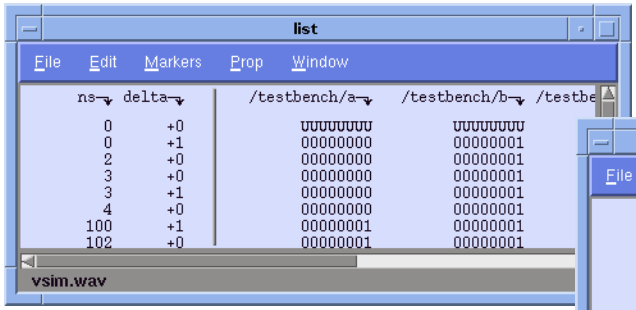
  For the other radixes - binary, octal, decimal, unsigned, hexadecimal, or ASCII - the item value is converted to an appropriate representation in that radix. In the system initialization file, *modelsim.tcl*, you can specify the list translation rules for arrays of enumerated types for binary, octal, decimal, unsigned decimal, or hexadecimal item values in the design unit.

- **Width**
  Allows you to specify the desired width of the column used to list the item value. The default is an approximation of the width of the current value.
- **Trigger: Triggers line**
  Specifies that a change in the value of the selected item causes a new line to be displayed in the List window.
- **Trigger: Does not trigger line**
  Selecting this option in the List Signals window specifies that a change in the value of the selected item does not affect the List window.

The trigger specification affects the trigger property of the selected item. See also, "Setting List window display properties" (p47).

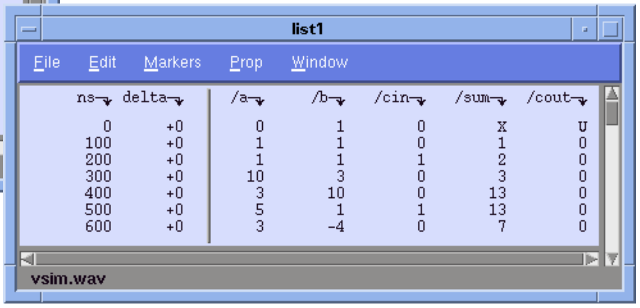
Examining simulation results with the List window

Because you can use the Main window View menu (p36) to create a second List window, you can reformat another List window after the simulation run if you decide a different format would reveal the information you're after. Compare the two illustrations.



```
      ns  delta   /testbench/a   /testbench/b   /testbe
       0     +0       UUUUUUUU       UUUUUUUU
       0     +1       00000000       00000001
       2     +0       00000000       00000001
       3     +0       00000000       00000001
       3     +1       00000000       00000001
       4     +0       00000000       00000001
     100     +1       00000001       00000001
     102     +0       00000001       00000001
```

symbolic item format - item change triggers a new line
- the divider bar separates resolution and delta from values



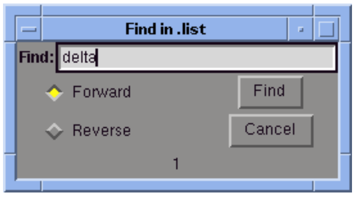
```
      ns  delta   /a    /b    /cin   /sum   /cout
       0     +0    0     1      0      X      U
     100     +0    1     1      0      1      0
     200     +0    1     1      1      2      0
     300     +0   10     3      0      3      0
     400     +0    3    10      0     13      0
     500     +0    5     1      1     13      0
     600     +0    3    -4      0      7      0
```

decimal and symbolic formats - 100ns strobe triggers a new line

In the first List window, the HDL items are formatted as symbolic and use an item change to trigger a line; the field width was changed to accommodate the default label width. The window divider maintains the time and delta in the left pane; signals in the right pane may be viewed by scrolling. For the second listing, the specification for triggering was changed to a 100-ns strobe, and the item radix for **a**, **b**, **cin**, and **sum** is now decimal.

Finding items by name in the List window

From the List window select **Edit > Find** to bring up the Find dialog box.



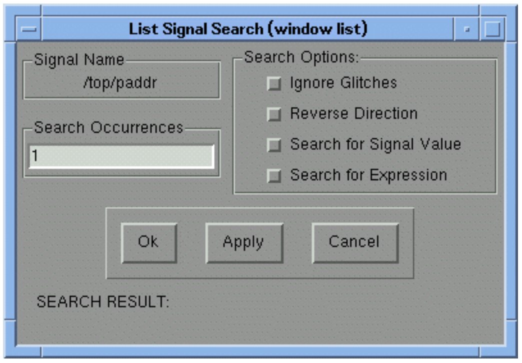
Find: delta

1

Enter an item label and **Find** it by searching **Forward** (right) or **Reverse** (left) through the List window display. The column number of the item found displays at the bottom of the dialog box. Note that you can change an item's label, see "Setting List window display properties" (p47).

Searching for item values in the List window

Select an item in the List window. From the List window menu bar select **Edit > Search** to bring up the List Signal Search dialog box.



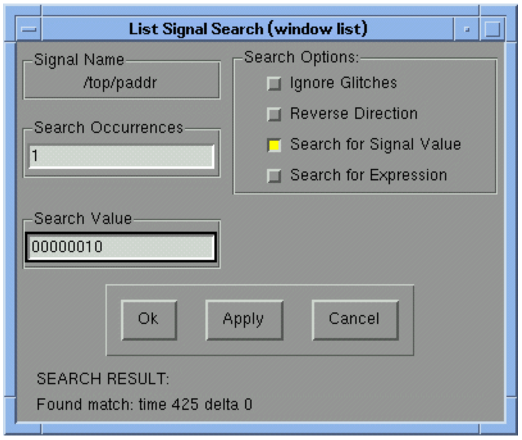
Signal Name: /top/paddr

Search Occurrences: 1

SEARCH RESULT:

The List Signal Search dialog box includes these options:

- **Signal Name <item_label>**
  This indicates the item currently selected in the List window; the subject of the search.
- **Search Options: Ignore Glitches**
  Ignore zero width glitches in VHDL signals and Verilog nets.
- **Search Options: Reverse Direction**
  Select to search the list from bottom to top. Deselect to search from top to bottom.
- **Search Options: Search for Signal Value**
  Reveals the Search Value field; search for the value specified in the Search Value field (the value must be formatted in the same radix as the display). If no value is specified look for transitions.


List Signal Search (window list)

Signal Name: /top/paddr

Search Occurrences: 1

Search Value: 00000010

Search Options:
- [ ] Ignore Glitches
- [ ] Reverse Direction
- [✓] Search for Signal Value
- [ ] Search for Expression

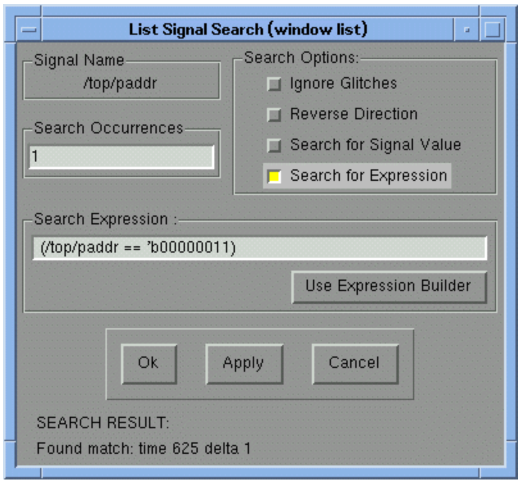
SEARCH RESULT:
Found match: time 425 delta 0

List Signal Search dialog box with Search for Signal Value selected

- **Search Options: Search for Expression**
  Reveals the Search Expression field and the Use Expression Builder button; searches for the expression specified in the Search Expression field evaluating to a boolean true.

  The expression may involve more than one signal but is limited to signals logged in the List window. Expressions may include constants, variables and macros. If no expression is specified, the search will give an error.

  To help build the expression, click the **Use Expression Builder** button.



Signal Name: /top/paddr

Search Occurrences: 1

Search Options:
- Ignore Glitches
- Reverse Direction
- Search for Signal Value
- ✓ Search for Expression

Search Expression : (/top/paddr == 'b00000011)

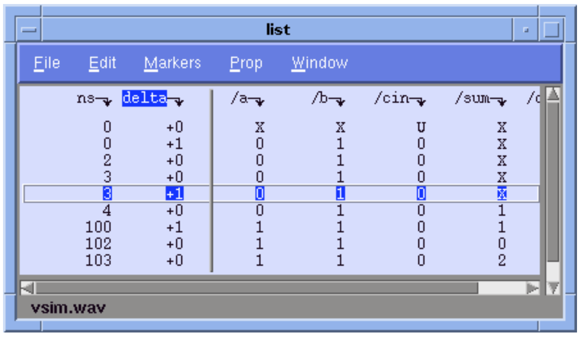
SEARCH RESULT:
Found match: time 625 delta 1

List Signal Search dialog box with Search for Expression selected

- **Search Occurrences**
  You can search for the n-th transition or the n-th match on value or expression; Search Occurrences indicates the number of transitions or matches for which to search.

Setting time markers in the List window

From the List window select **Markers > Add Marker** to tag the selected list line with a marker. The marker is indicated by a thin box surrounding the marked line. The selected line uses the same indicator, but its values are highlighted. Delete markers by first selecting the marked line, then making the **Markers > Delete Marker** menu selection.



```
       ns    delta     /a       /b     /cin     /sum    /c
        0       +0      X        X        U        X
        0       +1      0        1        0        X
        2       +0      0        1        0        X
        3       +0      0        1        0        X
        3       +1      0        1        0        X
        4       +0      0        1        0        1
      100       +1      1        1        0        1
      102       +0      1        1        0        0
      103       +0      1        1        0        2
```

Finding a marker

The marker name (on the **Goto** list) corresponds to the simulation time of the selected line. Choose a specific marked line to view with **Markers > Goto** menu selection.

List window keyboard shortcuts

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

Key	Action
<arrow up>	scroll listing up
<arrow down>	scroll listing down
<arrow left>	scroll listing left
<arrow right>	scroll listing right

Key	Action
<page up>	scroll listing up by page
<page down>	scroll listing down by page
<tab>	searches forward (down) to the next transition on the selected signal
<shift-tab>	searches backward (up) to the previous transition on the selected signal
<Control-f>	opens the find dialog box; find the specified item label within the list display

HP workstation note

<shift-tab> does not function on HP workstations.

Saving List window data to a file

From the List window select **Edit > Write List (format)** to save the List window data in one of these formats:

- **tabular**
  writes a text file that looks like the window listing

```
ns   delta       /a       /b     /cin     /sum    /cout
0      +0        X        X        U        X        U
0      +1        0        1        0        X        U
2      +0        0        1        0        X        U
```

- **event**
  writes a text file containing transitions during simulation

```
@0 +0
/a X
/b X
/cin U
```

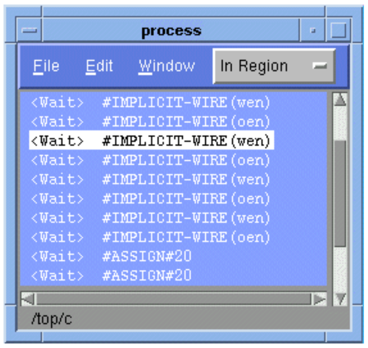
```
/sum X
/cout U
@0 +1
/a 0
/b 1
/cin 0
```

- **TSSI**
  writes a file in standard TSSI format.

```
0 0000000000000001O?????????
2 0000000000000001O???????1?
3 0000000000000001O??????010
4 0000000000000001000000001O
100 000000010000000100000000010
```

Process window

The Process window displays a list of processes and indicates the pathname of the instance in which the process is located.



```
<Wait>  #IMPLICIT-WIRE(wen)
<Wait>  #IMPLICIT-WIRE(oen)
<Wait>  #IMPLICIT-WIRE(wen)
<Wait>  #IMPLICIT-WIRE(oen)
<Wait>  #IMPLICIT-WIRE(wen)
<Wait>  #IMPLICIT-WIRE(oen)
<Wait>  #IMPLICIT-WIRE(wen)
<Wait>  #IMPLICIT-WIRE(oen)
<Wait>  #ASSIGN#20
<Wait>  #ASSIGN#20
/top/c
```

Each HDL item in the scrollbox is preceded by one of the following indicators:

- **<Ready>**
  Indicates that the process is scheduled to be executed within the current delta time.
- **<Wait>**
  Indicates that the process is waiting for a VHDL signal or Verilog net or variable to change or for a specified time-out period.
- **<Done>**
  Indicates that the process has executed a VHDL wait statement without a time-out or a sensitivity list. The process will not restart during the current simulation run.

If you select a "Ready" process, it will be executed next by the simulator.

When you click on a process in the Process window the following windows are updated:

Window updated	Result
Structure window (p74)	shows the region in which the process is located
Variables window (p77)	shows the VHDL variables and Verilog register variables in the process
Source window (p70)	shows the associated source code

Window updated	Result
Dataflow window (p42)	shows the process, the signals and nets the process reads, and the signals and nets driven by the process.

The Process window menu bar

The following menu commands and button options are available from the Process window menu bar.

File menu

Save As	save the process tree to a text file viewable with the Model*Sim* notepad
Environment	Follow Environment: update the window based on the selection in the Structure window (p74); Fix Environment: maintain the current view, do not update
Close	close this copy of the Process window; you can create a new window with View > New from the "The VSIM Main window menu bar" (p36)

Edit menu

Editing options	basic editing options include: Copy, Select All, and Unselect All
Sort	sort the process list in either ascending, descending, or declaration order
Find...	find specified text string within the structure tree; choose the Status (ready, wait or done) or Process label to search and the search direction: forward or reverse

Window menu

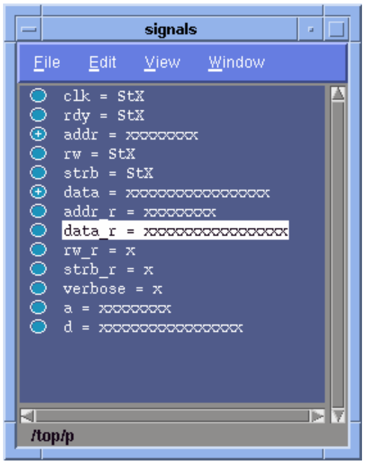
<VSIM_window>	switch to the selected (open) <VSIM_window> or show if hidden

Active/In Region toggle button

Active	Displays all the processes that are scheduled to run during the current simulation cycle.
In Region	Displays any processes that exist in the region that is selected in the Structure window.

Signals window

The Signals window shows the names and values of HDL items in the current region (which is selected in the Structure window). Items may be sorted in ascending, descending, or declaration order.



```
clk = StX
rdy = StX
addr = xxxxxxxx
rw = StX
strb = StX
data = xxxxxxxxxxxxxxxx
addr_r = xxxxxxxx
data_r = xxxxxxxxxxxxxxxx
rw_r = x
strb_r = x
verbose = x
a = xxxxxxxx
d = xxxxxxxxxxxxxxxx
```

HDL items viewed within the Signals window

The following HDL items for VHDL and Verilog are viewable within the Signals window.

VHDL items	Verilog items
signals	nets, register variables, and named events

The names of any VHDL composite types (arrays and record types) are shown in a hierarchical fashion. Hierarchy also applies to Verilog nets and vector memories. (Verilog vector registers do not have hierarchy because they are not internally represented as arrays.) Hierarchy is indicated in typical Model*Sim* fashion with plus (expandable), minus (expanded), and blank (single level) boxes.

See "Tree window hierarchical view" (p31) for more information.

The Signals window menu bar

The following menu commands are available from the Signals window menu bar.

File menu

Save As	save the signals tree to a text file viewable with the Model*Sim* notepad
Environment	Follow Environment: update the window based on the selection in the Structure window (p74); Fix Environment: maintain the current view, do not update
Close	close this copy of the Signals window; you can create a new window with View > New from the "The VSIM Main window menu bar" (p36)

Edit menu

Editing options	basic editing options include: Copy (then paste to Wave or List window), Select All, and Unselect All
Sort	sort the signals tree in either ascending, descending, or declaration order
Force...	apply stimulus to the specified Signal Name; specify Value, Kind (Freeze/Drive/Deposit), Delay, and Repeat
Noforce	removes the effect of any active force command on the selected HDL item
Find...	find specified text string within the Signals window; choose the Name or Value field to search and the search direction: forward or reverse

View menu

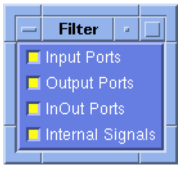
Wave/List/Log	place the Selected Signals, Signals in Region, or Signals in Design in the Wave window (p79), List window (p44), or Log file
Filter	choose the port and signal types to view (Input Ports, Output Ports, InOut Ports and Internal Signals) in the Signals window

Window menu

<VSIM_window>	switch to the selected (open) <VSIM_window> or show if hidden

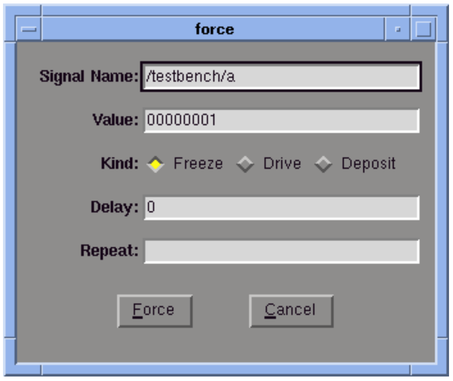
Selecting HDL item types to view

The **View > Filter...** menu selection allows you to specify which HDL items are shown in the Signals window. Multiple options may be selected.



Forcing signal and net values

The **Edit > Force** menu selection displays a dialog box that allows you to apply stimulus to the selected signal or net. You can specify that the stimulus is to repeat at a regular time interval, expressed in the time units set in the Startup window when you invoked the simulator.



Signal Name: /testbench/a

Value: 00000001

Kind: [✓ Freeze] Drive Deposit

Delay: 0

Repeat:

The **Force** dialog box includes these options:

- **Signal Name**
  Specify the signal or net for the applied stimulus.
- **Value**
  Initially displays the current value, which can be changed by entering a new value into the field. A value can be specified in radixes other than decimal by using the form (for VHDL and Verilog, respectively):

  ```
  base#value  -or-  b|o|d|h'value
  ```

  For example, 16#EE or h'EE specifies the hexadecimal value EE.
- **Kind: Freeze**
  Freezes the signal or net at the specified value until it is forced again or until it is unforced with a **noforce** command.
- **Kind: Drive**
  Attaches a driver to the signal and drives the specified value until the signal or net is forced again or until it is unforced with a **noforce** command. This value is illegal for unresolved VHDL signals.
- **Kind: Deposit**
  Sets the signal or net to the specified value. The value remains until there is a subsequent driver transaction, or until the signal or net is forced again, or until it is unforced with a **noforce** command.

Freeze is the default for Verilog nets and unresolved VHDL signals and **Drive** is the default for resolved signals.

If you prefer **Freeze** as the default for resolved and unresolved signals, you can change the default force kind in the *modelsim.ini* file.

- **Delay**
  Allows you to specify how many time units from the current time the stimulus is to be applied.

- **Repeat**
  Allows you to specify the time interval after which the stimulus is to be repeated. A value of 0 indicates that the stimulus is not to be repeated.
- **Force**
  When you click the **Force** button, a **force** command is issued with the parameters you have set and echoed in the Main window.

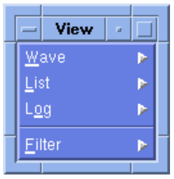
Adding HDL items to the Wave and List windows or a log file

Before adding items to the List or Wave window you may want to set the window display properties (see "Setting List window display properties" (p47)). Once display properties have been set, you can add items to the windows or log file in several ways.

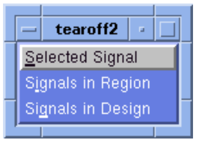
Adding items from the Main window command line

Use the **View** menu with either the **Wave**, **List**, or **Log** selection to add HDL items to the Wave window (p79), List window (p44)or a log file, respectively.

The log file is written as an archive file in binary format and is used to drive the List and Wave window at a later time. Once signals are added to the log file they cannot be removed. If you begin a simulation by invoking VSIM with the -view <logfile_name> option, VSIM reads the log file to drive the Wave and List windows.



Choose one of the following options (Model*Sim* opens the target window for you):



- **Selected signal**
  Lists only the item(s) selected in the Signals window.
- **Signals in region**
  Lists all items in the region that is selected in the Structure window.
- **Signals in design**
  Lists all items in the design.

Adding items from the Main window command line

Another way to add items to the Wave or List window or the log file is to enter the one of the following commands at the VSIM prompt:

```
add list | add wave | log <item_name> <item_name>
```

You can add all the items in the current region with this command:

```
add list | add wave | log *
```

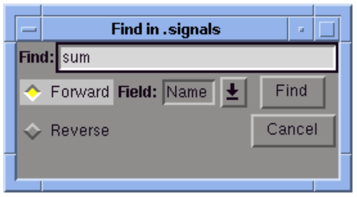
Or add all the items in the design with:

```
add list | add wave | log -r /*
```

If the target window (Wave or List) is closed, Model*Sim* opens it when you when you invoke the command.

Finding HDL items in the Signals window

Find the specified text string within the Signals window; choose the **Name** or **Value** field to search and the search direction: **Forward** or **Reverse**.

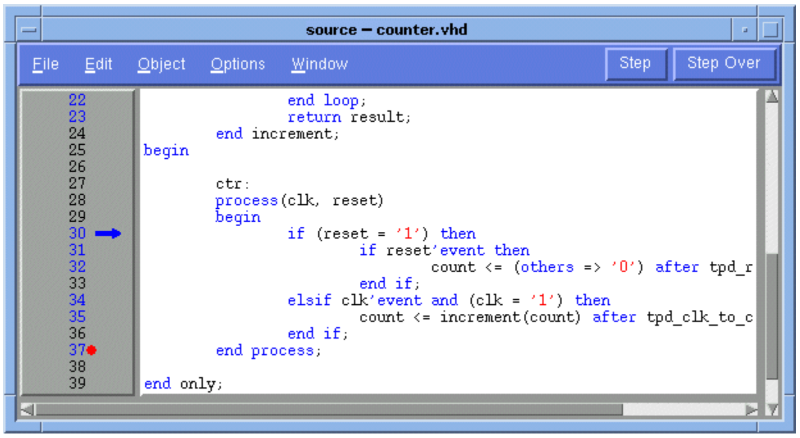


Find: sum

Field: Name

Source window

The Source window allows you to view and edit your HDL source code. Select an item in the Structure window (p74) or use the **File** menu to add a source file to the window, then select a process in the Process window (p62) to view that process; an arrow next to the line numbers indicates the selected process.



```
                end loop;
                return result;
        end increment;
begin

        ctr:
        process(clk, reset)
        begin
                if (reset = '1') then
                        if reset'event then
                                count <= (others => '0') after tpd_r
                        end if;
                elsif clk'event and (clk = '1') then
                        count <= increment(count) after tpd_clk_to_c
                end if;
        end process;

end only;
```

If any breakpoints have been set, each is signified by a colored dot next to a line number at the left side of the window pane. To set a breakpoint, click at or near the line number in the numbered area at the left side of the window. The breakpoints are toggles, so you can click again to delete an existing breakpoint. There is no limit to the number of breakpoints you can set.

To look at a file that is not currently being displayed, use the Structure window (p74) to select a different design unit or use the Source menu selection: **File** > **Open**. The pathname of the source file is indicated in the header of the Source window.

You can copy and paste text between the Source window and the VSIM Main window (p35); select the text you want to copy, then paste it into the Main window with the middle mouse button.

The Source window menu bar

The following menu commands are available from the Source window menu bar.

File menu

New	edit a new source file
Open	select a source file to open
Save	save the current source file
Save_As	save the current source file with a different name
Close	close this copy of the Source window; you can create a new window with View > New from the "The VSIM Main window menu bar" (p36)

Edit menu

To edit a source file, make sure the **Read Only** option in the Source Options dialog box is *not* selected (use the Source menu **Options > Options** selection).

<editing option>	basic editing options include: Cut, Copy, Paste, Select All, and Unselect All; see: "Editing the command line, the current source file, and notepads" (p40)
Find...	find the specified text string within the source file; match case option

Object menu

Describe	displays information about the selected HDL item; the item name is shown in the title bar
Examine	displays the current value of the selected HDL item; the item name is shown in the title bar

Options menu

Options	open the Source Options dialog box, see "Setting Source window options" (p72)

Window menu

<VSIM_window>	switch to the selected (open) <VSIM_window> or show if hidden

Editing the source file in the Source window

Several mouse actions and special keystrokes can be used to edit the source file in the Source window. See "Editing the command line, the current source file, and notepads" (p40) for a list of editing options; they can also be used to edit command line input in the VSIM Main window (p35).

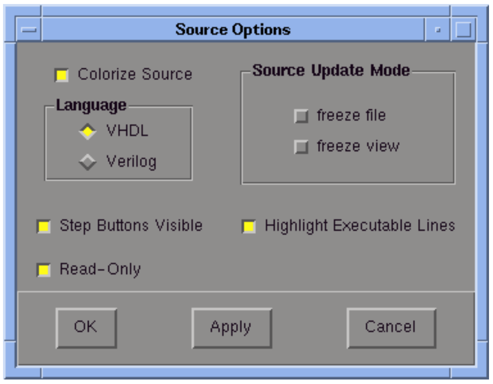
Checking HDL item values and descriptions

There are two quick methods to determine the value and description of an HDL item displayed in the Source window:

- select an item, then chose **Object > Examine** or **Object > Description** from the Source window menu
- select an item with the right mouse button to view examine pop-up (select "now" to examine the current simulation time in VHDL code)

Setting Source window options

Access the Source window options with this Source menu selection: **Options > Options**.

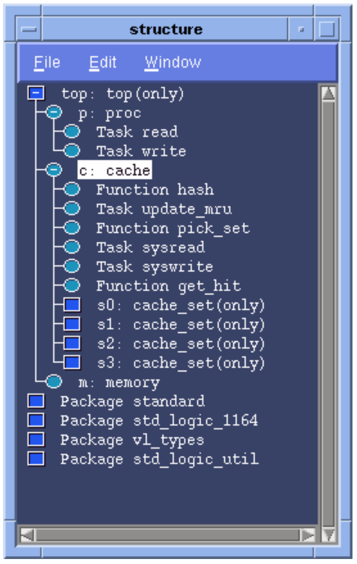


The **Source Options** dialog box includes these options:

- **Language**
  select either **VHDL** or **Verilog**; sets language for key word colorizing
- **Source Update Mode**
  select **freeze file** to maintain the same source file in the Source window (useful when you have two Source windows open; one can be updated from the Structure window (p74), the other frozen) or **freeze view** to disable updating the source view from the Process window (p62)
- **Colorize Source**
  colorize key words, variables and comments
- **Step Buttons Visible**
  select to add **Step** and **Step Over** buttons to the Source window menu bar
- **Read-Only**
  sets the source file to read-only mode
- **Highlight Executable Lines**
  highlights the line number of executable lines

Structure window

The Structure window provides a hierarchical view of the structure of your design. An entry is created by each HDL item within the design.



```
top: top(only)
  p: proc
    Task read
    Task write
  c: cache
    Function hash
    Task update_mru
    Function pick_set
    Task sysread
    Task syswrite
    Function get_hit
    s0: cache_set(only)
    s1: cache_set(only)
    s2: cache_set(only)
    s3: cache_set(only)
  m: memory
Package standard
Package std_logic_1164
Package vl_types
Package std_logic_util
```

HDL items viewed within the Structure window

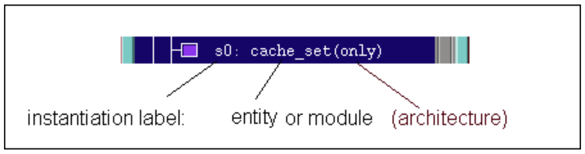
The following HDL items for VHDL and Verilog are represented by hierarchy within Structure window.

VHDL items	Verilog items
component instantiation, generate statement, block statement, and package	module instantiation, named fork, named begin, task and function

Within the Structure window, VHDL items are indicated by a box and Verilog items are indicated by a circle.You can expand and contract the display to view the elements by clicking on the boxes or circles at the left of the Window. The first line of the Structure window indicates the top-level design unit being simulated.

Instance name components in the Structure window

An instance name displayed in the Structure window consists of the following parts:



where:

- **instantiation label**
  Indicates the label assigned to the component or module instance in the instantiation statement.
- **entity or module**
  Indicates the name of the entity or module that has been instantiated.
- **architecture**
  Indicates the name of the architecture associated with the entity (not present for Verilog).

When you select a region in the Structure window, it becomes the *current region* and is highlighted; the Source window (p70) and Signals window (p64) change dynamically to reflect the information for that region. This feature provides a useful method for finding the source code for a selected region because the system keeps track of the pathname where the source is located and displays it automatically, without the need for you to provide the pathname.

Also, when you select a region in the Structure window, the Process window (p62) is updated if **In Region** is selected in that window; the Process window will in turn update the Variables window (p77).

The Structure window menu bar

The following menu commands are available from the Structure window menu bar.

File menu

Save_As	save the structure tree to a text file viewable with the Model*Sim* **notepad**
Close	close this copy of the Structure window; you can create a new window with View > New from the "The VSIM Main window menu bar" (p36)

Edit menu

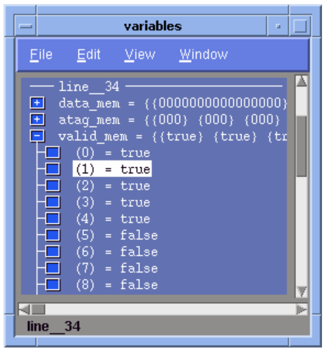
Editing options	basic editing options include: Copy, Select All, and Unselect All
Sort	sort the structure tree in either ascending, descending, or declaration order
Find...	find specified text string within the structure tree; choose the label for instance, entity/module or architecture to search for and the search direction: forward or reverse

Window menu

<VSIM_window>	switch to the selected (open) <VSIM_window> or show if hidden

Variables window

The Variables window lists the names of HDL items within the current process, followed by the current value(s) associated with each name. The pathname of the current process is displayed at the bottom of the window.



```
—— line__34 ——————————
[+] data_mem = {{0000000000000000}
[+] atag_mem = {{000} {000} {000}
[-] valid_mem = {{true} {true} {tr
    (0) = true
    (1) = true
    (2) = true
    (3) = true
    (4) = true
    (5) = false
    (6) = false
    (7) = false
    (8) = false
line__34
```

The names of any VHDL composite types (arrays and record types) are shown in a hierarchical fashion. Hierarchy also applies to Verilog vector memories. (Verilog vector registers do not have hierarchy because they are not internally represented as arrays.) Hierarchy is indicated in typical Model*Sim* fashion with plus (expandable), minus (expanded), and blank (single level) boxes. See "Tree window hierarchical view" (p31) for more information.

To change the value of a VHDL variable, constant, generic or Verilog register variable, move the pointer to the desired name and click to highlight the selection. Then select **Edit > Change** from the Variables window menu. This brings up a dialog box that lets you specify a new value. Note that "Variable Name" is a term that is used loosely in this case to signify VHDL constants and generics as well as VHDL and Verilog register variables. You can enter any value that is valid for the variable. An array value must be specified as a string (without surrounding quotation marks). To modify the values in a record, you need to change each field separately.

HDL items viewed within the Variables window

The following HDL items for VHDL and Verilog are viewable within the Variables window.

VHDL items	Verilog items
constants, generics, and variables	register variables

The Variables window menu bar

The following menu commands are available from the Variables window menu bar.

File menu

Save As	save the variables tree to a text file viewable with the Model*Sim* **notepad**
Environment	Follow Environment: update the window based on the selection in the Structure window (p74); Fix Environment: maintain the current view, do not update
Close	close this copy of the Variables window; you can create a new window with View > New from the "The VSIM Main window menu bar" (p36)

Edit menu

Editing options	basic editing options include: Copy (then paste to Wave or List window) , Select All, and Unselect All
Sort	sort the variables tree in either ascending, descending, or declaration order
Change	change the value of the selected HDL item
Find...	find specified text string within the variables tree; choose the Name or Value field to search and the search direction: forward or reverse

View menu

Wave/List/Log	place the Selected Variables, Variables in Region, or Variables in Design in the Wave window (p79), List window (p44), or Log file

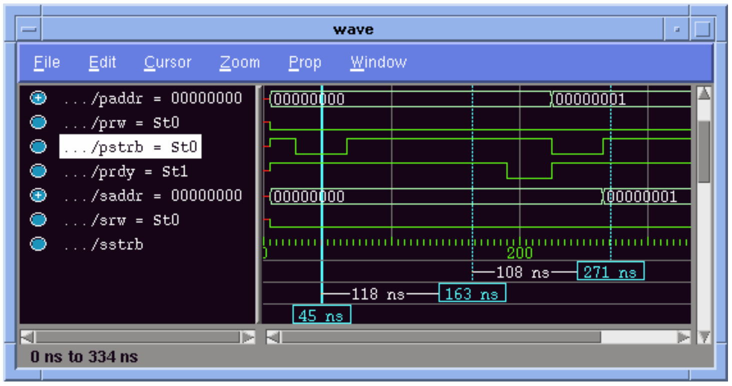
Window menu

<VSIM_window>	switch to the selected (open) <VSIM_window> or show if hidden

Wave window

The Wave window, like the List window, allows you to view the results of your simulation. In the Wave window, however, you can see the results as HDL item waveforms and their values.

The Wave window is divided into two windowpanes: the left pane displays item names and their values at the active cursor (located in the right pane); the right pane displays waveforms corresponding to each item and any cursors you may have added.



```
.../paddr = 00000000
.../prw = St0
.../pstrb = St0
.../prdy = St1
.../saddr = 00000000
.../srw = St0
.../sstrb
```

The data in the item values windowpane is very similar to the Signals window, except that the values change dynamically whenever a cursor in the waveform windowpane is moved.

At the bottom of the waveform windowpane you can see a time line, tick marks, and a readout of the cursor(s) position(s). As you click and drag to move a cursor, the time value at the cursor location is updated at the bottom of the cursor.

You can resize the windowpanes by clicking and dragging the bar between the two windowpanes.

Waveform and signal-name formatting are easily changed via the Prop menu (p83). You can reuse any formatting changes you make by saving a Wave window format file, see "Adding items with a Wave window format file" (p85).

HDL items viewed within the List window

The following HDL items for VHDL and Verilog are viewable in the Wave window.

VHDL items	Verilog items
signals and process variables	nets, register variables, and named events

Note:

Constants, generics, parameters, and memories are not viewable in the List or Wave windows.

Wave window action list

This action list provides a quick reference to menu selections and mouse actions in the Wave window. See the "Tree window action list" (p32) for additional information.

Action	Menu or mouse	See also
collapse and expand composites	left mouse button: click on a "-" or "+" box/ circle	"Tree window action list" (p32)
move items, make single and multiple item selections	left mouse button and control + left mouse button	"Tree window action list" (p32)
search for transitions in the waveform display and signal values	menu selection: Edit > Find...	"Searching for item values in the Wave window" (p89)
find an item name or value	menu selection: Edit > Find	"Finding items by name or value in the Wave window" (p89)
find a specific cursor	menu selection: Cursor > Goto	"Making cursor measurements" (p92)
sort items: ascending, descending, declaration order	menu selection: Edit > Sort	"Sorting a group of HDL items" (p89)
making cursor measurements	menu selection: Cursor > Add Cursor	"Making cursor measurements" (p92)

Action	Menu or mouse	See also
reformat items, change their display colors, and position them within the window	menu selection: Prop > Display Props...	"Editing and formatting HDL items in the Wave window" (p86)
zoom in or out to change the amount of simulation time shown in the window	menu selection: Zoom > (select zoom option) center mouse button: click and drag to zoom rubberband section	"Zooming - changing the waveform display range" (p93)
change signal properties: radix, color, height, format (analog/literal/logic/event)	menu selection: Prop > Signal Props	"Setting Wave window display properties" (p83)
save the Wave window configuration; load a new configuration	menu selection: File > Save (or Load) Format	
adding and editing items in the name/value pane	menu selection: Edit > (select edit option)	"Adding HDL items in the Wave window" (p84)

The Wave window menu bar

The following menu commands and button options are available from the Wave window menu bar. If you see a dotted line at the top of a drop-down menu, click and drag the dotted line to create a separate menu window.

File menu

Write Postscript	save the waveform display as a Postscript file
Load Format	run a Wave window format (do) file previously saved with Save Format
Save Format	saves the current Wave window display and signal preferences to a do (macro) file; running the do file will reformat the Wave window to match the display as it appeared when the do file was created
Close	close this copy of the Wave window; you can create a new window with View > New from the "The VSIM Main window menu bar" (p36)

Edit menu

Cut	cut the selected item from the wave name pane; see "Editing and formatting HDL items in the Wave window" (p86)
Copy	copy the selected item and waveform
Paste	paste the previously cut or copied item above the currently selected item
Combine	combine the selected fields into a user defined bus
Sort	sort the top-level items in the name pane; sort with full path name or viewed name; use ascending, descending or declaration order
Delete	delete the selected item and its waveform
Select All Unselect All	select, or unselect, all item names in name pane
Find...	find specified item label within the Wave name window
Search...	search the waveform display for a specified value, or the next transition for the selected signal; see: "Searching for item values in the Wave window" (p89)

Cursors menu

Add Cursor	add a cursor to the center of the waveform window
Delete Cursor	delete the selected cursor from the window
Goto	choose a cursor to go to from a list of current cursors

Zoom menu

Zoom <selection>	selection: Full, In, Out, Last, or Range to change the waveform display range

Prop menu

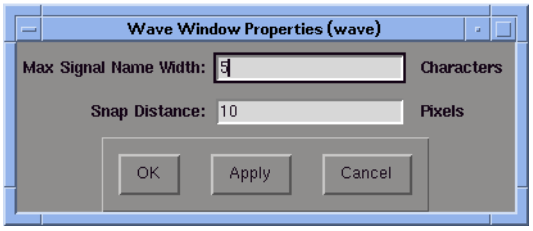
Display Props	set display properties for all items in the window: delta settings, trigger on selection, strobe period, and label size
Signal Props	set label, radix, trigger on/off, and field width for the selected item (use the menu selections below to quickly change individual properties)
Radix	set the selected item’s radix
Format	set the waveform format for the selected item
Color	set the color for the selected item from a color palette
Height	set the waveform height in pixels for the selected item

Window menu

<VSIM_window>	switch to the selected (open) <VSIM_window> or show if hidden

Setting Wave window display properties

You can define the item name width and the cursor snap distance of all items in the Wave window with the **Prop > Display Props...** menu selection.



Wave Window Properties (wave)

Max Signal Name Width: 5 Characters

Snap Distance: 10 Pixels

The **Wave Window Properties** dialog box includes these options:

- **Max Signal Name Width**
  Sets the item name width. This is especially useful for items that have a long pathname. Choose a maximum name width setting, say 10 characters, and then item pathnames longer than 10 characters are truncated on the left. All truncations take place at the slash boundary so you will never see a partial item name. The default value for this field is 0, which means to display the full path. Negative numbers allow you to specify the number of regions, instead of characters, to display.
- **Snap Distance**
  Specifies the distance the cursor needs to be placed from an item edge to jump to that edge (a 0 specification turns off the snap). The value displayed in the item value windowpane is updated to reflect the snap.

Adding HDL items in the Wave window

Before adding items to the Wave window you may want to set the window display properties (see "Setting Wave window display properties" (p83)). You can add items to the Wave window in several ways.

Adding items from the Signals window with drag and drop

You can drag and drop items into the Wave window from the Process, Signals, or Structure window. Select the items in the first window, then drop them into the Wave window. Depending on what you select, all items or any portion of the design may be added. See the "Tree window action list" (p32) for information about making item selections.

Adding items from the Main window command line

To add specific HDL items to the window, enter (separate the item names with a space):

```
add wave <item_name> <item_name>
```

You can add all the items in the current region with this command:

```
add wave *
```

Or add all the items in the design with:

```
add wave -r /*
```

Adding items with a Wave window format file

To use a Wave window format file you must first save a format file for the design you are simulating.

- add the items you want in the Wave window with any other method shown above
- edit and format the items, see "Editing and formatting HDL items in the Wave window" (p86) to create the view you want
- save the format to a file with the Wave window menu selection: **File** > **Save Format**

To use the format file, start with a blank Wave window and run the do file in one of two ways:

- use the do command on the command line:
  ```
  do <my_wave_format>
  ```
- select **File** > **Load Format** from the Wave window menu bar

Use **Edit** > **Select All** and **Edit** > **Delete** to remove the items from the current Wave window, use the **delete** command with the **wave** option, or create a new, blank Wave window with the **View** > **New** > **Wave** selection from the VSIM Main window (p35).

Note:
Wave window format files are design-specific; use them only with the design you were simulating when they were created.

Editing and formatting HDL items in the Wave window

Once you have the HDL items you want in the Wave window, you can edit and format the list in the name/value pane to create the view you find most useful. (See also, "Adding HDL items in the Wave window" (p84).)

To edit an item:

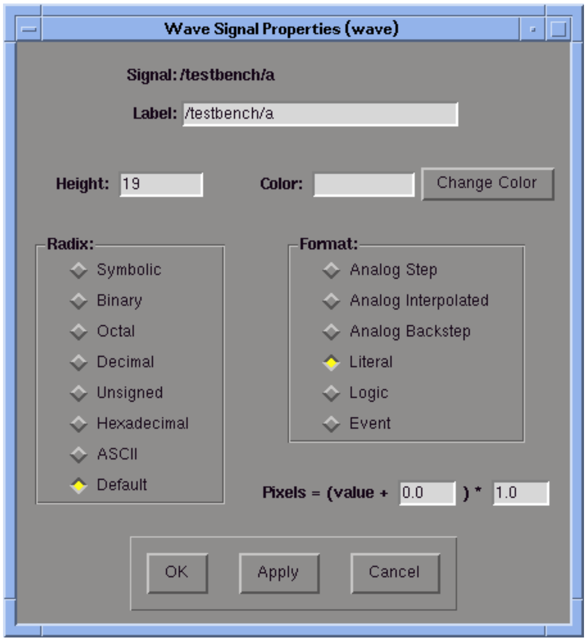
Select the item's label in the left name/value windowpane or its waveform in the right windowpane. Move, copy or remove the item by selecting commands from the Wave window Edit menu (p82) menu.

You can also **click+drag** to move items within the name/value windowpane:

- to select several contiguous items:
  click+drag to select additional items above or below the original selection
- to select several items randomly:
  Control+click to add or subtract from the selected group
- to move the selected items:
  re-click on one of the selected items and drag to the new location

To format an item:

Select the item's label in the left name/value pane or its waveform in the right windowpane, then use the **Prop > Signal Props...** menu selection. The resulting Wave Signal Properties dialog box allows you to set the item's height, color, format, range, and radix.



Wave Signal Properties (wave)

Signal: /testbench/a

Label: /testbench/a

Height: 19

Color:

Change Color

Radix:
- Symbolic
- Binary
- Octal
- Decimal
- Unsigned
- Hexadecimal
- ASCII
- [✓ Default]

Format:
- Analog Step
- Analog Interpolated
- Analog Backstep
- [✓ Literal]
- Logic
- Event

Pixels = (value + 0.0) * 1.0

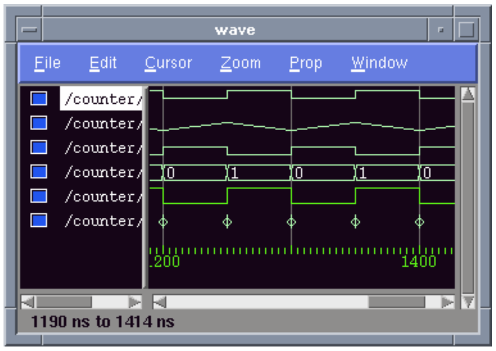
OK Apply Cancel

The **Wave Signal Properties** dialog box includes these options:

- **Signal**
  Indicates the name of the currently selected signal.
- **Label**
  Allows you to specify a new label (in the name/value pane) for the selected item.
- **Height**
  Allows you to specify the height (in pixels) of the waveform.
- **Color**
  Lets you override the default color of a waveform by selecting a new color from the color palette, or by entering an X-Windows color name.
- **Format: Analog Step**
  Displays a waveform of an integer, real or time type, with the height and offset determined by the **Pixels =** specification and the value of the item.

- **Format: Analog Interpolated**
  Displays the waveform in interpolated style.
- **Format: Analog Backstep**
  Displays the waveform in backstep style. Used for power calculations.
- **Format: Literal**
  Displays the waveform as a box containing the item value (if the value fits the space available). This is the only format that can be used to list a record.
- **Format: Logic**
  Displays values as 0, 1, X, Z, H, L, U, or -.
- **Format: Event**
  Marks each transition during the simulation run.

  The illustration below shows the same item displayed in each wave format:



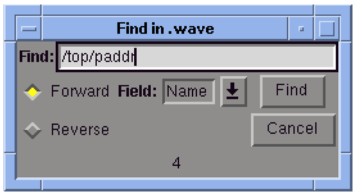
- **Pixels = (value + <offset>) * <scale factor>**
  This choice works with analog items only and allows you to decide on the scale of the item as it is seen on the display. Value is the value of the signal at a given time, <offset> is the number of pixels offset from zero. The <scale factor> reduces (if less than 1) or increases (if greater than 1) the number of pixels displayed.
- **Radix**
  The explicit choices are Symbolic, Binary, Octal, Decimal, Unsigned, Hexadecimal, and ASCII. If you select Default the signal's radix changes whenever the default is changed using the **radix** command. Item values are not translated if you select Symbolic.

Sorting a group of HDL items

Use the **Edit > Sort** menu selection to sort the items in the name/value pane.

Finding items by name or value in the Wave window

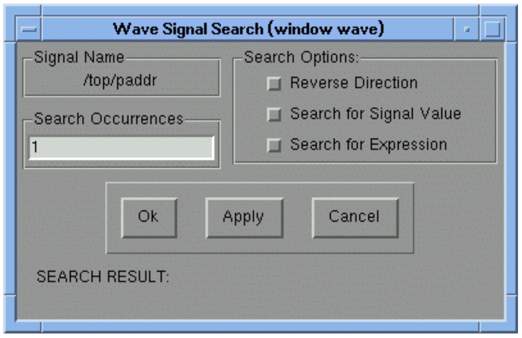
From the Wave window select **Edit > Find** to bring up the Find dialog box.



Choose either the Name or Value field to search from the drop-down menu, and enter the value to search for in the Find field. **Find** the item by searching **Forward** (down) or **Reverse** (up) through the Wave window display.

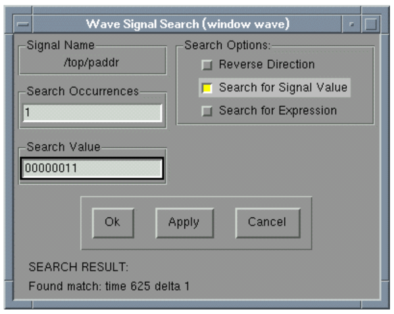
Searching for item values in the Wave window

Select an item in the Wave window. From the Wave window menu bar select **Edit > Search** to bring up the Wave Signal Search dialog box.



The **Wave Signal Search** dialog box includes these options:

- **Signal Name <item_label>**
  This indicates the item currently selected in the Wave window; the subject of the search.
- **Search Options: Ignore Glitches**
  Ignore zero width glitches in VHDL signals and Verilog nets.
- **Search Options: Reverse Direction**
  Search the list from right to left. Deselect to search from left to right.
- **Search Options: Search for Signal Value**
  Reveals the Search Value field; search for the value specified in the Search Value field (the value must be formatted in the same radix as the display). If no value is specified the search will look for transitions.



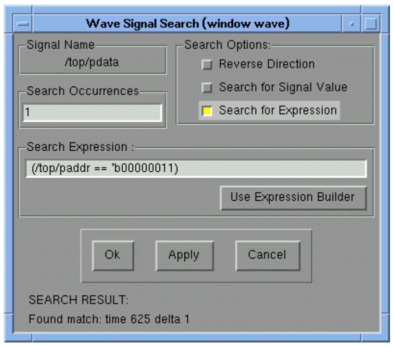
Signal Name: /top/paddr
Search Occurrences: 1
Search Value: 00000011
Search Options: Search for Signal Value (selected)

SEARCH RESULT:
Found match: time 625 delta 1

Wave Signal Search dialog box with Search for Signal Value selected

- **Search Options: Search for Expression**
  Reveals the Search Expression field and the Use Expression Builder button; searches for the expression specified in the Search Expression field evaluating to a boolean true.

  The expression may involve more than one signal but is limited to signals logged in the List window. Expressions may include constants, variables, and macros. If no expression is specified, the search will give an error. See the *ModelSim EE/PLUS Reference Manual* for more information on expression syntax and the use of the Expression Builder.


Wave Signal Search (window wave)

Signal Name: /top/pdata

Search Occurrences: 1

Search Options:
- [ ] Reverse Direction
- [ ] Search for Signal Value
- [✓] Search for Expression

Search Expression : (/top/paddr == 'b00000011)

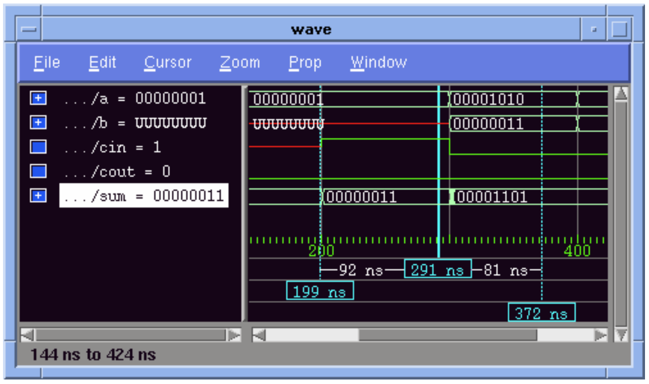
SEARCH RESULT:
Found match: time 625 delta 1

Wave Signal Search dialog box with Search for Expression selected

- **Search Occurrences**
  You can search for the n-th transition or the n-th match on value or expression; Search Occurrences indicates the number of transitions or matches for which to search.

Using time cursors in the Wave window

When the Wave window is first drawn, there is one cursor in it at time zero. Clicking anywhere in the waveform display brings that cursor to the mouse location.You can add additional cursors to the waveform pane with the **Cursor > Add Cursor** menu selection. The selected cursor is drawn as a solid line; all other cursors are drawn with dotted lines. Remove cursors by selecting them and choosing using the **Cursor > Delete Cursor** menu selection.



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. VSIM also adds a delta measurement showing the time difference between the two cursor positions.

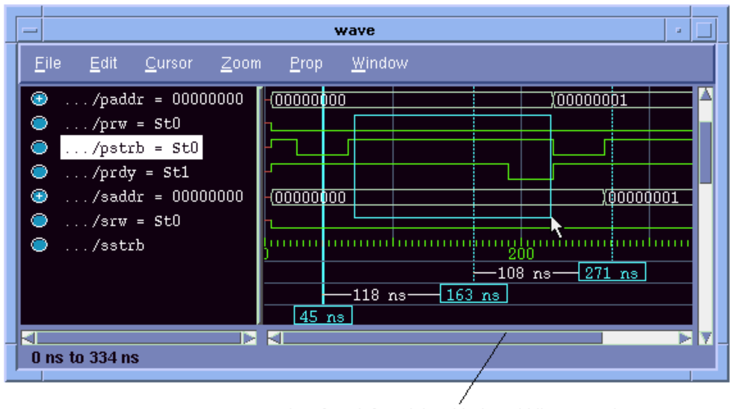
If you click in the waveform display, the cursor closest to the mouse position is selected and then moved to the mouse position. Another way to position multiple cursors is to use the mouse in the time box tracks at the bottom of the display. Clicking anywhere in a track selects that cursor and brings 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. You can control the snap distance from "Wave category" in the dialog box available from the **Properties** > **Display** menu selection; see "Setting Wave window display properties" (p83).

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

Zooming - changing the waveform display range

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



drag from left to right with the middle mouse button to zoom

Using the Zoom menu

You can use the Wave window menu bar, or call up a **Zoom** menu window with the right mouse button in the right windowpane. The 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, cropping the view on the right. The starting time is held static.
- **Zoom Out**
  Zooms out by a factor of two, decreasing the resolution and increasing the visible range horizontally, extending the view on the right. The starting time is held static.
- **Zoom Last**
  Restores the display to where it was before the last zoom operation.

- **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 mouse

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

Zooming keyboard shortcuts

See "Wave window keyboard shortcuts" (p94) for a complete list of Wave window keyboard shortcuts.

Wave window keyboard shortcuts

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

Key	Action
<page down>	scroll waveform display down by page
<tab>	searches forward (right) to the next transition on the selected signal
<shift-tab>	searches backward (left) to the previous transition on the selected signal
<Control-f>	opens the find dialog box; search within the specified field in the wave-name pane for text strings

4 - Tutorial: Using Model*Sim* EE

Chapter contents

Choose the lessons appropriate for your simulator version:

Assumptions

We assume that you are familiar with the graphical interface on your workstation: either OpenWindows or OSF/Motif on SPARCstations and OSF/Motif on IBM and Hewlett-Packard workstations; your workstation interface provides the window-management frame, while Model*Sim* controls all internal-window features including menus, buttons, and scroll bars.

We also assume that you have a working knowledge of HDL design. Although Model*Sim* is an excellent tool to use while learning HDL concepts and practices, this guide is not written to support that goal.

Additional details for VHDL, Verilog, and mixed VHDL/Verilog simulation can be found in the *ModelSim EE/PLUS Reference Manual*. See "Where to find our documentation" (p15).

Basic VHDL simulation

The goals for the first lesson are:

- create a library
- compile a VHDL file
- start the simulator
- understand the basic VSIM windows, mouse, and menu conventions
- run VSIM 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
- use the Wave window

Many of the steps in this lesson are accomplished by a button or menu selection. When appropriate, command-line equivalents for button and menu selections are shown in parentheses at the end of the step.

As you work on the lessons keep an eye on the Main window transcript. The commands invoked by buttons and menu selections are echoed there.

(Note that neither the prompt nor the Return that ends a command line is shown in the examples.)

Step 1.

Create a new directory and go there by entering the UNIX shell commands:

```
mkdir <directory_name>
cd <directory_name>
```

Step 2.

Copy the VHDL files from the */<install_dir>/modeltech/examples* directory into the current directory by entering the command (note the period specifying the current directory):

```
cp /<install_dir>/modeltech/examples/*.vhd .
```

Step 3.

Before you compile any HDL code, you need a design library to hold the compilation results. To create a new design library, enter the following:

```
vlib work
```

This creates a subdirectory named *work*. This subdirectory contains a special file named *_info*. Do not create these using UNIX shell commands—always use the **vlib** command.

Step 4.

Compile the file *counter.vhd* by entering the command:

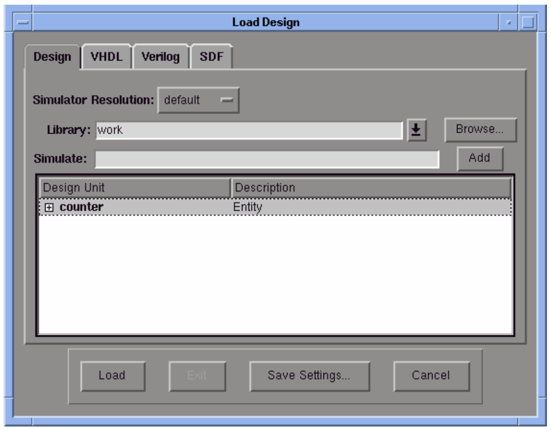
```
vcom counter.vhd
```

Step 5.

Start the simulator by entering the command:

```
vsim
```

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



Simulator Resolution: default

Library: work

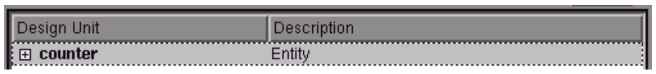
Simulate:

Design Unit	Description
counter	Entity

The Load Design dialog box allows you to select the library and the 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 ns)
- Library: work
- Design Unit: counter
- Description: entity

Note:
If the Design Unit is an entity you can click on the plus-box prefix (shown below) to view any associated architectures.

➔ 

Design Unit	Description
⊞ **counter**	Entity

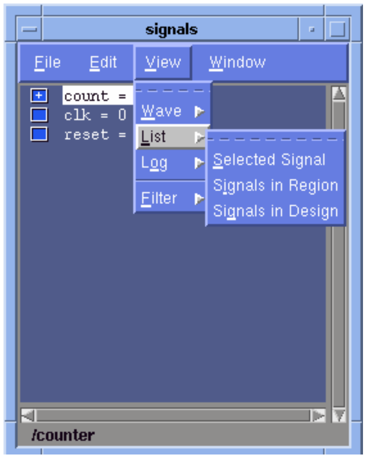
Step 6.
Select the entity **counter** and choose **Load** to accept these settings.

Step 7.
Now you can open all of the VSIM windows with this Main window menu selection:
View > All.

(on the command line use: view *)

Step 8.
To display the top-level signals in the List window, select the Signals window and make this Signal menu selection: **View > List > Signals in Region**.

(on the command line use: add list /counter/*)



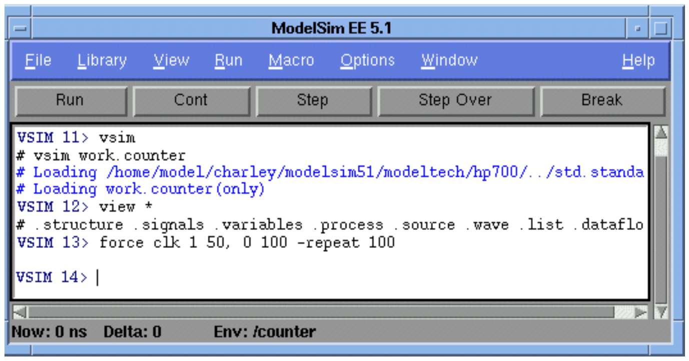
Step 9.

Next add top-level signals to the Wave window with a similar Signals menu selection: **View > Wave > Signals in Region**.

Step 10.

You can apply stimulus to the clock input by moving the pointer to the Main window and entering the following command at the VSIM prompt, as shown in the illustration below:

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


ModelSim EE 5.1
File Library View Run Macro Options Window Help
Run Cont Step Step Over Break
Now: 0 ns Delta: 0 Env: /counter

```
VSIM 11> vsim
# vsim work.counter
# Loading /home/model/charley/modelsim51/modeltech/hp700/../std.standa
# Loading work.counter(only)
VSIM 12> view *
# .structure .signals .variables .process .source .wave .list .dataflo
VSIM 13> force clk 1 50, 0 100 -repeat 100

VSIM 14> |
```

The **force** command is interpreted by VSIM to mean:

- 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

You will see the effects of this **force** command as soon as you tell the simulator to run.

Step 11.

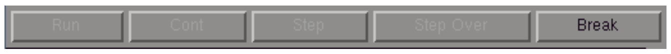
Now you will exercise two different **Run** functions. Make these selections from the Main window menu bar:

Run > Run 100. This causes the simulation to run and then stop after 100 ns (ns is the default resolution. (on the command line use: run 100)

Run > Run -All. This causes the simulator to run forever. To stop the run, go on to the next step. (on the command line use: run -all)

Step 12.

Select the **Break** button on the Main window tool bar.

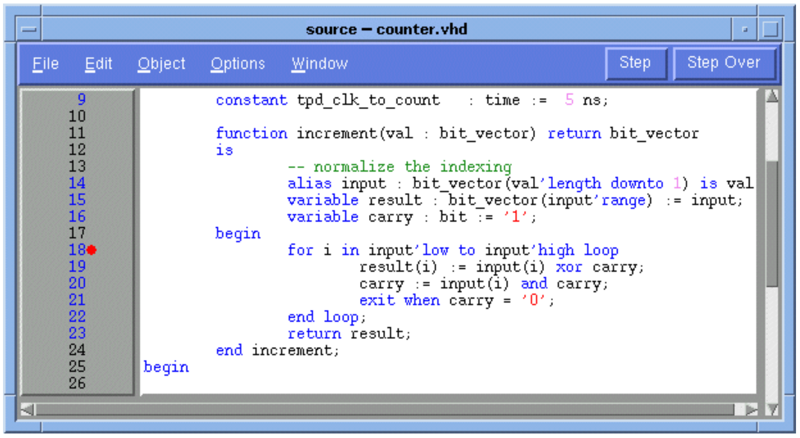


The arrow points to the next HDL statement to be executed.

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

Step 13.

Move the pointer to the VSIM Source window. Using the vertical scroll bar, scroll until line 18 is visible. Click at or near line number 18 to set the breakpoint. You should see a dot next to the line number where the breakpoint is set. This breakpoint can be toggled on and off by clicking it.



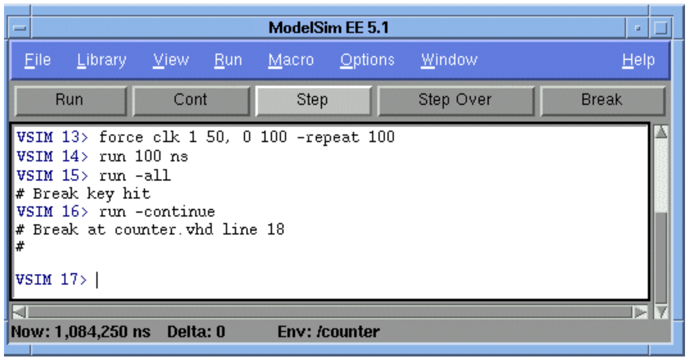
```
 9              constant tpd_clk_to_count   : time :=  5 ns;
10
11              function increment(val : bit_vector) return bit_vector
12              is
13                      -- normalize the indexing
14                      alias input : bit_vector(val'length downto 1) is val
15                      variable result : bit_vector(input'range) := input;
16                      variable carry : bit := '1';
17              begin
18●                     for i in input'low to input'high loop
19                              result(i) := input(i) xor carry;
20                              carry := input(i) and carry;
21                              exit when carry = '0';
22                      end loop;
23                      return result;
24              end increment;
25      begin
26
```

Step 14.

Select **Continue** in the Run menu in the main VSIM window to resume the run that you interrupted. VSIM will hit the breakpoint, as shown by an arrow in the VSIM Source window and by a message in the main VSIM window. Also note that the parameters and variables within the function are displayed in the VSIM Variables window.

Step 15.

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

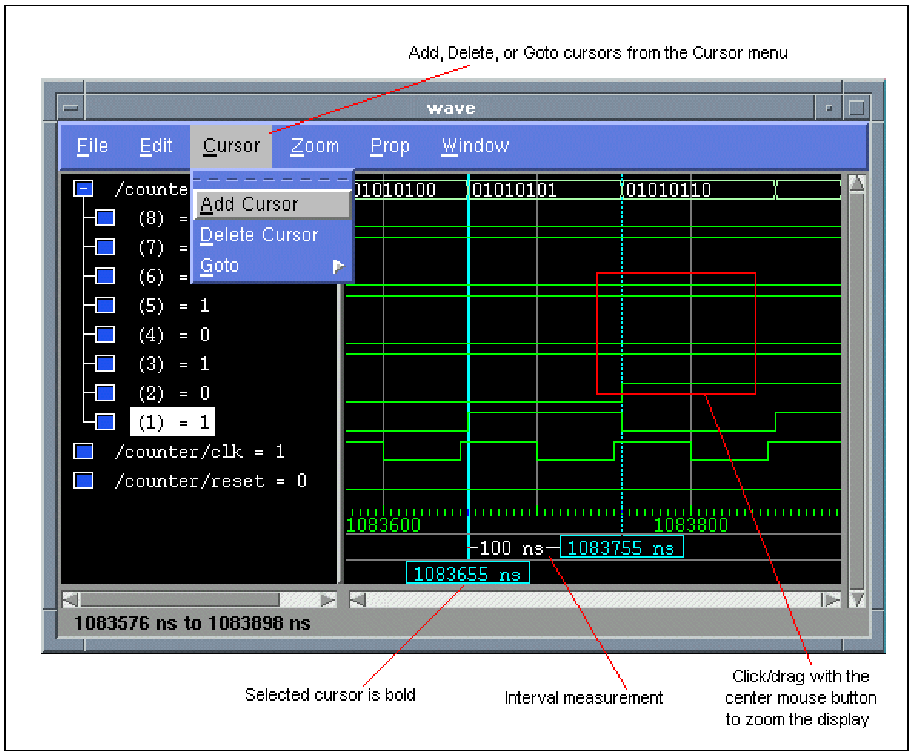


```
VSIM 13> force clk 1 50, 0 100 -repeat 100
VSIM 14> run 100 ns
VSIM 15> run -all
# Break key hit
VSIM 16> run -continue
# Break at counter.vhd line 18
#

VSIM 17> |
```

Now: 1,084,250 ns   Delta: 0   Env: /counter

Now let’s take a look at the simulation in the Wave window. In the Wave window, you can use cursors to:

- **probe for values**
  Signal values update whenever you move the cursor.
- **find signal transition times**
  Click on a signal edge, the cursor displays the time.
- **measure time intervals**
  Intervals are measured and displayed between two cursors.



Step 16.

Experiment with using the cursors, scrolling, and zooming (see page 155 for Wave window shortcuts).

Step 17.

When you’re done experimenting, quit the simulator by entering the command:

```
VSIM> quit -force
```

This command exits VSIM without saving data. Your window positions will be saved in the *modelsim.ini* file and the windows will close.

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 VSIM Variables window
- change the value of a variable
- use a strobe to trigger lines in the VSIM List window
- change the radix of signals displayed in the VSIM List window

Step 1.

Return to the directory you created in "Basic VHDL simulation" (p99), and enter the following command at the shell prompt to create the a new library:

```
vlib library_2
```

Step 2.

Compile the source files into the new library by entering this command at the shell prompt:

```
vcom -work library_2 gates.vhd adder.vhd testadder.vhd
```

Step 3.

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
```

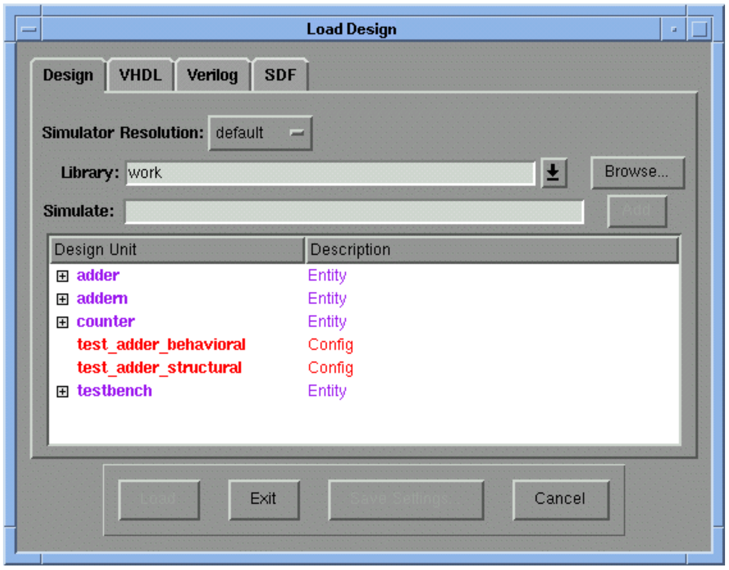
Model*Sim* modifies the *modelsim.ini* file for you.

Step 4.

Start the simulator by entering the following command at the shell prompt:

```
vsim
```

The Load Design dialog box is displayed, as shown below.



Simulator Resolution: default

Library: work

Simulate:

Design Unit	Description
⊞ adder	Entity
⊞ addern	Entity
⊞ counter	Entity
test_adder_behavioral	Config
test_adder_structural	Config
⊞ testbench	Entity

Step 5.

Perform the following steps in this window:

- Make sure that the simulator resolution is **ns (default)**.
- Look in the Design Unit scroll box and select the configuration named **test_adder_structural**.
- Click **Load** to accept the settings.

Step 6.

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

```
VSIM> view *
```

Model*Sim* will open all the windows in the positions you left them in at the end of the last exercise if no one has run the simulator since then.

Step 7.

To list the top-level signals, enter the command:

```
VSIM> add list /*
```

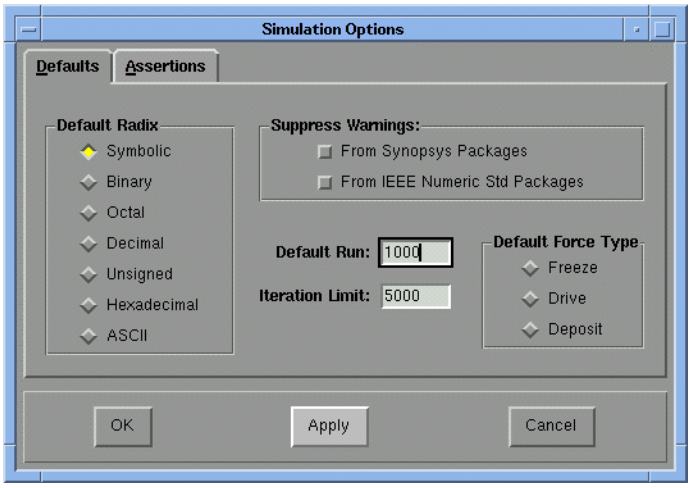
Step 8.

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

```
VSIM> add wave /*
```

Step 9.

To change the length of the default simulation run make this Main window menu selection: **Options > Simulation**. The Simulation Options dialog box is displayed.

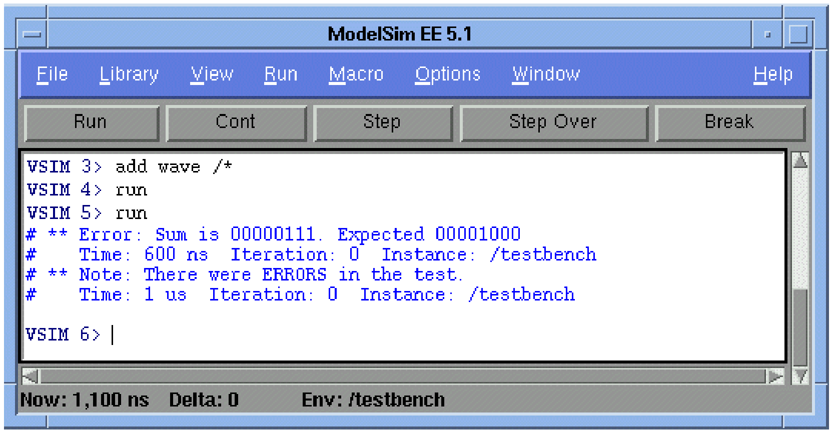


Default Run: 1000
Iteration Limit: 5000

Change the **Default Run** field to **1000**. Do not change the other settings. Click **OK** to accept the new settings.

Step 10.

Next, you will run the simulator. Select the **Run** button on the Main window toolbar.



```
VSIM 3> add wave /*
VSIM 4> run
VSIM 5> 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 6> |
```

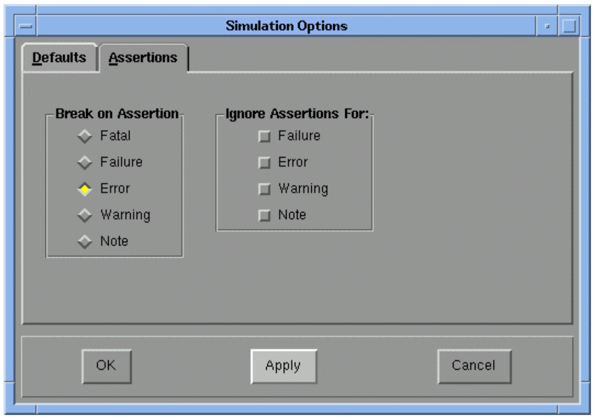
Now: 1,100 ns Delta: 0 Env: /testbench

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

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

Step 11.

Let’s change the simulation options again. Make this Main window menu selection: **Options > Simulation**.



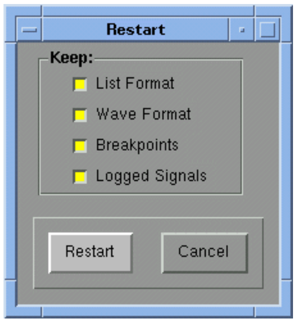
Break on Assertion
- [ ] Fatal
- [ ] Failure
- [✓] Error
- [ ] Warning
- [ ] Note

Ignore Assertions For:
- [ ] Failure
- [ ] Error
- [ ] Warning
- [ ] Note

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 statement *after* the assertion is displayed.

Step 12.

To restart the simulation make this Main window menu selection: **File > Restart**.



Keep:
- [✓] List Format
- [✓] Wave Format
- [✓] Breakpoints
- [✓] Logged Signals

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

Step 13.

From the Main window menu bar select **Run > Run 1000 ns**. Notice that the arrow in the Source window is pointing to the statement after the assertion.

Step 14.

If you turn to 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.

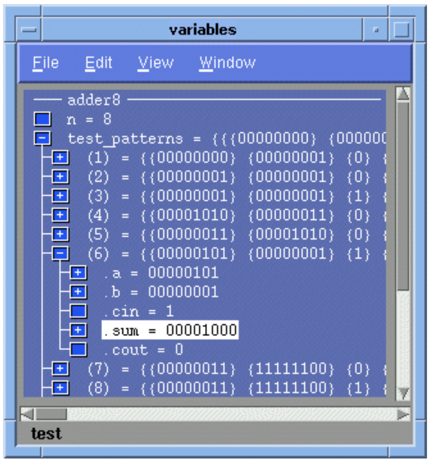
Step 15.

Expand the variable named **test_patterns** by clicking the [+]. (You may need to resize the window for a better view.)

Step 16.

Also expand the sixth record in the array, that is, **test_patterns(6)**, by clicking the [+].

The Variables window should be similar to the one below.



```
adder8
n = 8
test_patterns = {{{00000000} {000000
  (1) = {{00000000} {00000001} {0}
  (2) = {{00000001} {00000001} {0}
  (3) = {{00000001} {00000001} {1}
  (4) = {{00001010} {00000011} {0}
  (5) = {{00000011} {00001010} {0}
  (6) = {{00000101} {00000001} {1}
    .a = 00000101
    .b = 00000001
    .cin = 1
    .sum = 00001000
    .cout = 0
  (7) = {{00000011} {11111100} {0}
  (8) = {{00000011} {11111100} {1}
```

The assertion shows that the signal **sum** does not equal the **sum** field in **test_patterns(6)**. 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.

Step 17.

In the main VSIM window, type:

```
VSIM> restart -f
```

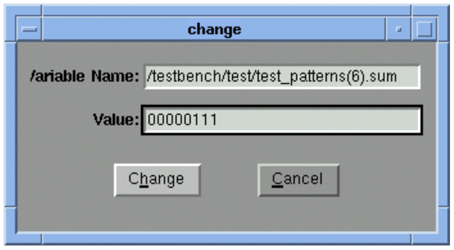
The **-f** option causes VSIM to restart without popping up the confirmation dialog.

Step 18.

In the Process window select the **test** to add variables to the Variables window.

Step 19.

In the Variables window, expand **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.



/ariable Name: /testbench/test/test_patterns(6).sum

Value: 00000111

Step 20.

Select the last four bits in the value field **1000** 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.)

Step 21.

Select **Run** from the Main window toolbar. At this point, the simulation will run without any errors.

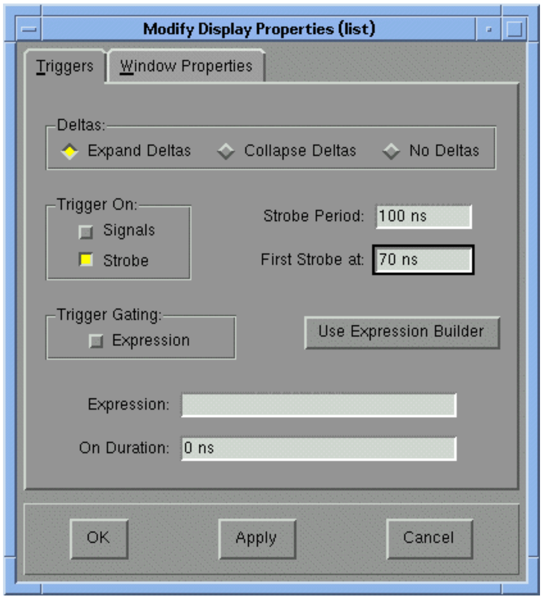
```
# ** Note: Test completed with no errors.
#    Time: 1 us  Iteration: 0  Instance: /
```

Next you will learn how to change the new-line triggering for the List window.

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.

Step 22.

In the List window make this menu selection: **Prop > Display Props**.



Deltas: [✓ Expand Deltas] Collapse Deltas No Deltas

Trigger On: Signals [✓ Strobe]

Strobe Period: 100 ns

First Strobe at: 70 ns

Trigger Gating: Expression

Expression:

On Duration: 0 ns

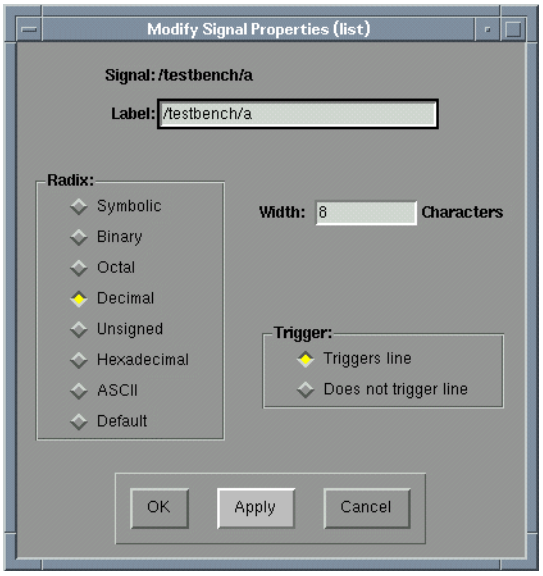
Perform these steps on **Triggers** page in the Modify Display Properties (list) dialog box :

- 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.

Step 23.

Your next action will be to change the radix for a, b, and sum to decimal.

Make this List window menu selection: **Prop > Signal Props**. This will open the Modify Signal Properties (list) dialog box.



Signal: /testbench/a

Label: /testbench/a

Radix: Symbolic [✓ Decimal] Binary Octal Unsigned Hexadecimal ASCII Default

Width: 8 Characters

Trigger: [✓ Triggers line] Does not trigger line

Step 24.

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 VSIM without saving data by entering the following command at the VSIM prompt:

```
VSIM> quit -force
```

Running a batch-mode simulation

The goals for this lesson are:

- run a batch-mode VHDL simulation
- execute a macro file
- view a saved simulation

Step 1.

To set up for this lesson you'll need to create a new directory and library, and copy and compile a VHDL file. Once you have created and moved to your new directory, invoke these commands:

```
cp /<install_dir>/modeltech/examples/counter.vhd .

vlib work

vcom counter.vhd
```

Step 2.

You will use a macro file that provides stimulus for the counter. For your convenience, a macro file has been provided with Model*Sim* EE. You need to copy this macro file from the installation directory by entering the command:

```
cp /<install_dir>/modeltech/examples/stim.do  .
```

Step 3.

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 format list counter.lst
```

Step 4.

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

```
vsim -wav saved.wav counter < yourfile
```

This is what you just did in Step 4:

- invoked the VSIM simulator on a design unit called counter that was provided with your software
- you used the **-wav** switch to instruct the simulator to save the simulation results in a log file named *saved.wav*
- 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

Step 5.

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

```
vsim -view saved.wav
```

Step 6.

Open these windows with the VIEW button or the equivalent command:

```
VSIM> view structure 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.wav* was used to reconstruct the current windows.

Step 7.

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

```
VSIM> add wave *
VSIM> add list *
```

Step 8.

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

```
VSIM> quit
```

If you want to read more about the batch and command line modes, see the *ModelSim EE/PLUS Reference Manual*.

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

Step 1.

For this lesson, you will use a macro file that provides startup information. For convenience, a startup file has been provided with the Model*Sim* program. You need to copy this macro file from the installation directory by entering the command (note the period specifying the current directory):

```
cp /<install_dir>/modeltech/examples/startup.do  .
```

Step 2.

Next, you will edit the system initialization file in the current directory to specify a command that is to be executed after the design is loaded. To do this, open the *modelsim.ini* file using a text editor and add the following line in the [vsim] section of the file:

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

Step 3.

Take a look at the macro file. It uses the predefined variable **$entity** to do different things at startup for different designs.

Step 4.

Start the simulator and specify the top-level design unit to be simulated by entering the following command at the shell 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 macro that you executed at startup.

Step 5.

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

```
VSIM> quit -force
```

To read more about the *modelsim.ini* file and batch/command line modes, see the *ModelSim EE/PLUS Reference Manual*.

Tcl/Tk and Model*Sim*

This lesson is divided into several Tcl examples intended to give you a sense of Tcl/Tk's function within Model*Sim*. The examples include a custom simulation interface created with Tcl/Tk. You must be using Model*Sim* EE/PLUS or Model*Sim* EE/VHDL to complete these exercises.

Examples in this lesson

More information on Tcl/Tk

Please note that this lesson is not intended as a Tcl tutorial. If you would like additional Tcl/Tk information, check out our "Resources" (p161) chapter; it contains additional information on the following Tcl/Tk reference sources.

- Books (p161)
- Online resources (p167)

How Tcl/Tk works with Model*Sim*

Model*Sim* incorporates Tcl as an embedded library package. The Tcl library consists of a parser for the Tcl language, routines to implement the Tcl built-in commands, and procedures that allow Tcl to be extended with additional commands specific to Model*Sim*.

Model*Sim* generates Tcl commands and passes them to the Tcl parser for execution. Commands may be generated by reading characters from an input source, or by associating command strings with Model*Sim*'s user interface features, such as menu entries, buttons, or keystrokes.

When the Tcl library receives commands it parses them into component fields and executes built-in commands directly. For commands implemented by Model*Sim*, Tcl calls back to the application to execute the commands. In many cases commands will invoke recursive invocations of the Tcl interpreter by passing in additional strings to execute (procedures, looping commands, and conditional commands all work in this way).

Model*Sim* gains a programming advantage by using Tcl for its command language. Model*Sim* can focus on simulation-specific commands, while Tcl provides many utility commands, graphic interface features, and a general programming interface for building up complex command procedures.

By using Tcl, Model*Sim* need not re-implement these features, a benefit that allows it's graphic interface to remain consistent on all workstation platforms. (The only vestige of the host platform's graphic interface is the window frame manager.)

The custom-traffic-light interface

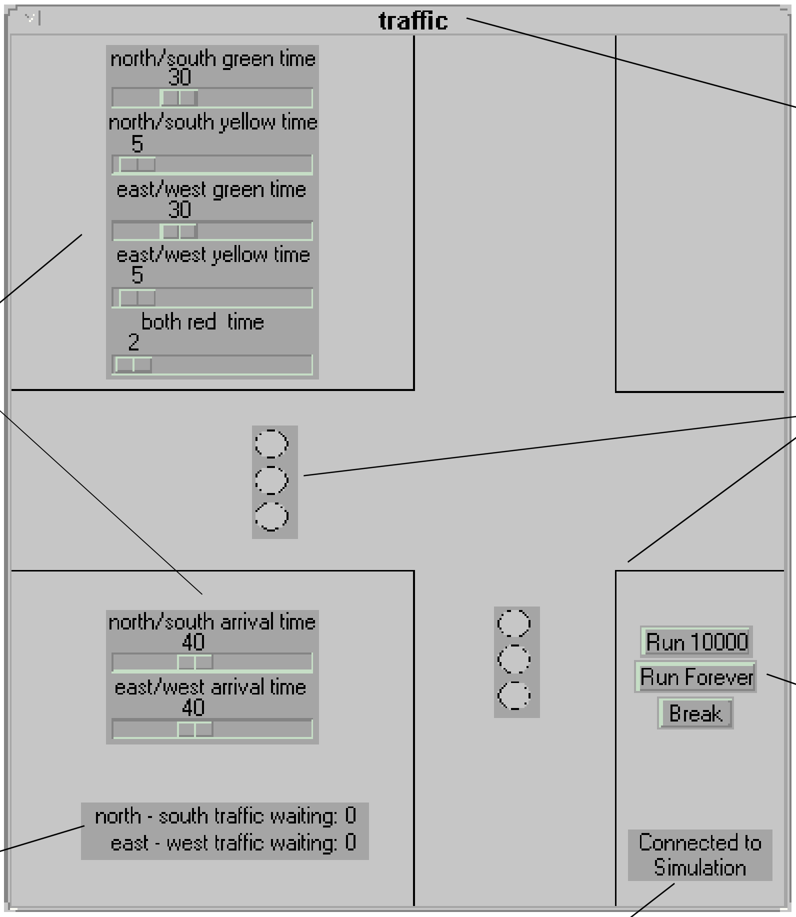
The subject of our Tcl/Tk lesson is a simple traffic-light controller. The system is comprised of three primary components: a state machine, a pair of traffic lights, and a pair of traffic sensors. The components are described in three VHDL files: traffic.vhd (the state machine), queue.vhd (the traffic arrival queue) and tb_traffic.vhd (the testbench).

You could, of course, simulate this system with Model*Sim*'s familiar interface, but Tcl/Tk provides us the option to try something different. Since we're simulating something most of us have seen and experienced before, we can create an intuitive interface unique to the simulation.

The example assumes that source files were previously compiled. Here's how it works:

➔ **VHDL source files** describe the system	➔ **Tcl procedures** create and connect the interface, plus the source files, to Model*Sim*	➔ **Model*Sim* commands** are run via the new interface using the Tcl procedures
	draw_intersection	
traffic.vhd queue.vhd tb_traffic.vhd	connect_lights	vsim -lib vhdl/work tb_traffic examine -value <light_timing>
	draw_queues	
	draw_controls	force -freeze $var $val ns

The result is a traffic intersection interface similar to this illustration:



wm widget
Calls to the workstation window manager to create the "traffic" window.

frame and scale widget
A scale widget within a frame widget creates an analog entry device for a minimum to a maximum value and invokes the VSIM force command

canvas widget
The background, lines and traffic lights are created with the canvas widget.

button widgets
Each button invokes the indicated VSIM run or break command.

entry widget
The entry widget (contained within a frame widget) can facilitate entry, or in this case, provides a display for a VSIM examine command.

label widget
A static label is added to the end of the connect_lights procedure to indicate connection to the simulator.

Tk widgets

The intersection illustration points out several Tcl/Tk "widgets". A widget is simply a user interface element, like a menu or scrolled list. Tk widgets are referenced within Tcl procedures to create graphic interface objects. The Tk tool box comes with several widgets, additional widgets can be created using these as a base.

Running the simulation

The components of the intersection interface have the following effect within Model*Sim*:

Control used	Effect in Model*Sim*
Run 10000 button	invokes the run command for 10000 ns
Run Forever button	invokes the run -all command
Break button	invokes the break command
light timing control	invokes the force command with the arguments for the indicated signal and time
arrival time control	invokes the force command with the arguments for the indicated direction and time
waiting queue	any time you change a control the examine command is invoked to display the value of the waiting queue

Since each control change invokes a new VSIM command, this custom interface would save you time compared to invoking the commands from the command line or Model*Sim* menus.

Copies of the original example files

Additional copies of the Tcl example files from these exercises are located in the *../examples/ tcl_getstart/originals* directory.

Viewing source files

If you would like to view the source for any of the Tcl files in our examples, use the **notepad** command at either the Model*Sim* or VSIM prompt. Here's the command for viewing the *examples.tcl* file:

```
notepad examples.tcl
```

The file is opened in readme mode by default; you can edit the file by making the **Edit > read only** menu selection in the Main window (make sure read only is NOT selected). See "The Main window tool bar" (p39) for more information.

The example procedure

The Tcl **source** command reads the *examples.tcl* file and defines the procedures within the file for use within the current environment; the **example** procedure from this file is used for all of our examples.

Example syntax

The Tcl procedure "example" (defined in examples.tcl) is used to invoke all of our examples from the Model*Sim* prompt:

```
example <tcl filename> <tcl procedure name>
```

Arguments

```
<tcl filename>
```
the Tcl file used for this example

```
<tcl procedure name>
```
one of the Tcl procedures defined within <tcl filename>

Preparing for the Tcl/Tk examples

These steps must be completed before running the Tcl examples.

Step 1.
Create, and change to a new working directory for the Tcl/Tk exercises. Copy the lesson files to your new directory with this command:

```
cp -r /<install_dir>/modeltech/examples/tcl_getstart .
```

Step 2.
Change to the *tcl_getstart/vhdl* directory:

```
cd tcl_getstart/vhdl
```

and create a work library:

```
vlib work
```

Step 3.
Compile the VHDL example files with this command:

```
vcom traffic.vhd queue.vhd tb_traffic.vhd
```

and return to the *tcl_getstart* directory:

```
cd ..
```

Step 4.
All Tcl examples are run from the Model*Sim* prompt after invoking VSIM.

To arrive at the Model*Sim* prompt, simply invoke VSIM and dismiss the Load Design dialog box by clicking **Cancel** (don't use Exit, or you will quit Model*Sim*).

Step 5. Begin the Tcl examples with this command at the Model*Sim* prompt:

```
 source examples.tcl
```

You're now ready to run the examples.

Example 1 - create a "hello world" button widget

Before you begin the examples make sure you have completed "Preparing for the Tcl/Tk examples" (p126), steps 1 through 3.

In this example you will study a "hello world" button that prints a message when pressed.

Step 1.

Invoke VSIM from within the *../tcl_getstart* directory, and **Cancel** the resulting Load Design dialog box.

This will cause VSIM to use the proper *modelsim.ini* and *modelsim.tcl* files.

Step 2.

Begin the this example with the **source** command at the Model*Sim* prompt:

```
 source examples.tcl
```

The Tcl **source** command reads the *examples.tcl* file and defines the procedures within the file for use within the current environment.

Step 3.

Now invoke this command from the Model*Sim* prompt:

```
example hello.tcl hello_example
```

The file *hello.tcl* was read into the VSIM Tcl interpreter. The instructions in *hello.tcl* were then executed by VSIM; "Hello World" was printed on the standard output.

You’ve just created your first top-level widget!

Example 2 - add a procedure that gets called by a button push

Before you begin the examples make sure you have completed "Preparing for the Tcl/Tk examples" (p126), steps 1 through 3.

In this example you will study a larger Tcl example, and add a procedure that gets called by a button push.

This example will display all of the gif images in the images directory. Each button has a binding attached to it for enter events and a binding for a mouse button press. When the mouse enters the button, the image file name is printed on the standard output. When the mouse button is pushed its "widget" name will be printed on the standard output.

Step 1.

Invoke VSIM from within the *../tcl_getstart* directory, and **Cancel** the resulting Load Design dialog box.

This will cause VSIM to use the proper *modelsim.ini* and *modelsim.tcl* files.

Step 2.

Run the image viewer by invoking this command:

```
example images.tcl image_example
```

Step 3.

Drag the mouse across the buttons and notice what happens on the standard output.

Push one of the buttons; you will see an error dialog box. You can solve this problem by modifying the *images.tcl* file.

Step 4.

To view the source file press the **See Source Code** button at the bottom of the image display or invoke **notepad** at the Model*Sim* prompt:

```
notepad images.tcl
```

You'll find that the pushme procedure is missing; it's commented out in *images.tcl*.

Remove the comments to return the function to your source.

Step 5.

Once the **pushme** procedure is in place it will print its one parameter, the object name, to stdout.

After you have added the **pushme** procedure to your source, you need to reload and rerun the tcl file with this command:

```
example images.tcl image_example
```

Press all the buttons and notice the object names in the standard out.

Example 3 - introduction of the traffic intersection widget

This example introduces the traffic intersection widget; we'll study the structure of the widget and see how it works.

The traffic intersection is the basis for the traffic light simulation. The three following examples (examples 4, 5, and 6) will add more detail to the traffic intersection display.

In order to run any of the three following examples, you need to have run this example first. The intersection should always be displayed.

Once again, make sure you have completed "Preparing for the Tcl/Tk examples" (p126), steps 1 through 3 before working this example.

Step 1.

Draw the intersection by invoking this command at the VSIM prompt:

```
example intersection.tcl draw_intersection
```

Step 2.

From the VSIM prompt, use the procedure set_light_state to change the color of the lights:

```
set_light_state green .traffic.i.ns_light

set_light_state green .traffic.i.ew_light
```

Step 3.

View the source code with this command at the VSIM prompt:

```
notepad intersection.tcl
```

The set_light_state procedure is located toward the middle of the file.

Example 4 - connect traffic lights to the simulation

Using the intersection widget you will add **when** statements so that the lights get "connected" to the real simulation. Once the connection is made you will simulate the traffic light controller and watch the lights change.

We'll use VSIM **when** statements to condition the simulation to call our Tcl program when a desired simulation condition happens.

For our example, the desired condition is the state of the lights. Whenever the state of the light in the simulation changes, we want to change the color of the light on the screen.

Step 1.
If the intersection widget is not already displayed, invoke this command:

```
example intersection.tcl draw_intersection
```

If the intersection widget is already displayed, you don't need to re-display it.

Step 2.
Load the VHDL libraries using this command at the Model*Sim* prompt:

```
vsim -lib vhdl/work tb_traffic
```

Be sure you invoke this command before the start of the connect_lights procedure, if you don't load the libraries, you won't have a design to simulate.

Step 3.
Connect the lights to the simulation with this command:

```
 example lights.tcl connect_lights
```

Step 4.
Edit *lights.tcl* with the notepad to add a **when** statement for the North/South light. You need to add this because the current statement is for the East/West light only.

You’ll find the code commented out toward the end of the file.

Example 5 - add widgets to display simulation information

For this example you will add queue widgets to display the sum of the length of each pair of queues as we simulate.

Step 1.

If the intersection widget is not already displayed, run this series of commands.

Draw the intersection:

```
example intersection.tcl draw_intersection
```

Load the design:

```
vsim -lib vhdl/work tb_traffic
```

Connect the lights:

```
example lights.tcl connect_lights
```

If the intersection widget is already displayed, you don't need to re-draw it. If the "Connected to Simulation" message is displayed, you don't need to load the design or re-connect the lights.

Step 2.

The East/West widget for displaying the total East/West queue length is already provided. Let's edit the source to add a display for the North/South direction. Use this command:

```
notepad queues.tcl
```

Cut, paste and modify the East/West widget to display the North/South total queue length.

The Queue Display widget consists of an enclosing frame with two label widgets. The first label is a simple text string. The second label is the value of the queue length.

The text in the second label will be updated whenever the queue lengths change.

The update is accomplished by using a **when** statement.

Step 3.

After you have added your North/South widget, run your program by invoking this command:

```
example queues.tcl draw_queues
```

Example 6 - add "slider" widgets to control the simulation

This example will add Tk "scale" widgets that will control the arrival rates and the length of the lights.

Step 1.

If the intersection widget is not already displayed, run this series of commands.

Draw the intersection:

```
example intersection.tcl draw_intersection
```

Load the design:

```
vsim -lib vhdl/work tb_traffic
```

Connect the lights to the simulation:

```
example lights.tcl connect_lights
```

Draw the queue-length display:

```
example queues.tcl draw_queues
```

If the intersection widget is already displayed, you don't need to re-draw it. If the "Connected to Simulation" message is displayed, you don't need to load the design or re-connect the lights.

If the queue-length widget is displayed, you don't need to re-draw it.

Step 2.

The East/West widget for controlling the East/West queue inter-arrival time is provided. You'll edit the source code to add controls for the North/South direction. Use this command:

```
notepad controls.tcl
```

You can cut, paste and modify the East/West widget to control the North/South queue inter-arrival time. (You can also remove the comments in the code to make this change.)

Similarly, add the North/South widget for controlling the length of the lights. The East/West widget for light control is provided. (You can remove the comments in the code to make this change as well.)

These control widgets are implemented using the Tk "scale" widgets, enclosed in a frame.

When the value of the scale widget changes, it calls the command specified with the **-command** option on each scale.

See the "set_time_variable" command.

Step 3.

After you have added your North/South widgets, run your program with this command:

```
example controls.tcl draw_controls
```

Step 4.

Use the controls you have added to experiment with the parameters of the simulation and observe the results.

Contrast this experience with "traditional" methods for exploring the parameter space.

Example 7 - draw a state machine that represents the simulation

In this final example you will draw a state machine representing the simulation, and connect it to the state signal inside the traffic light controller. Each transition that the controller makes is displayed as it happens.

Step 1.

Run the state machine with this command:

```
example state-machine.tcl draw_state_machine
```

Let’s make some changes to the light colors and transition arrows.

Step 2.

Open the source file with this command:

```
notepad state-machine.tcl
```

Note the "Example 7, part 1" comments in the file. Change "both_red" state coordinates as indicated.

Step 3.

Note the "Example 7, part 2" comments in the file. Change the transition arrow coordinates as indicated.

Step 4.

Note the "Example 7, part 3" comments in the file. Change the active color from "black" to "purple".

Step 5.

When you're ready to run your program, use this command:

```
example state-machine.tcl draw_state_machine
```

Notice the changes. Try some additional changes if you wish.

This is the end of the Tcl/Tk examples. Continue to modify and test the examples if you wish; you can recover the original files at any time in the tcl_getstart/originals directory.

Basic Verilog simulation

You must be using Model*Sim* EE/PLUS or Model*Sim* EE/VLOG for this lesson.

The goals for this lesson are:

- compile a Verilog design
- examine the hierarchy of the design
- list signals in the design
- change list attributes
- set a breakpoint
- add and remove cursors in the waveform display

If you've completed any previous VHDL lesson you'll notice that the Verilog simulation process is almost identical to the VHDL process.

Step 1.
Create a new directory and go there by entering these UNIX shell commands:

```
mkdir <directory_name>
cd <directory_name>
```

Step 2.
Copy the Verilog files from the */<install_dir>/modeltech/examples* directory into the current directory by entering the command (note the period specifying the current directory):

```
cp /<install_dir>/modeltech/examples/*.v .
```

Before you can compile a Verilog design, you need to create a design library in the new directory. If you are only familiar with interpreted Verilog simulators such as Cadence Verilog XL this will be a new idea for you. Since Model*Sim* is a compiled Verilog, it requires a target design library for the compilation. Model*Sim* can compile both VHDL and Verilog code into the same library if desired.

Step 3.
To create a new design library, enter the following:

```
vlib work
```

Remember, a library directory should not be created using UNIX shell commands - always use the Main window Library menu or the **vlib** command.

Next, you need to compile the Verilog design.

The example design we'll be using 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.

Step 4.
Compile the Verilog files by entering the command:

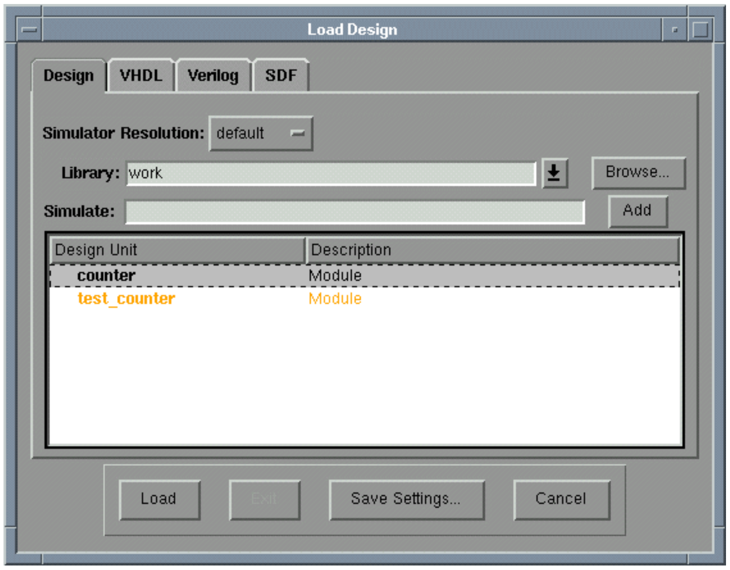
```
vlog counter.v tcounter.v
```

Note that the order in which you compile the two Verilog modules is not important. This may again seem strange to Verilog XL users who understand the possible problems of interface checking between design units, or compiler directive inheritance. Model*Sim* defers such checks until the design is loaded by VSIM (the HDL simulator). So it doesn't matter here if you choose to compile *counter.v* before or after *tcounter.v*.

Step 5.
Start the simulator by entering the command:

```
vsim
```

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



Simulator Resolution: default

Library: work

Simulate:

Design Unit	Description
counter	Module
test_counter	Module

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 ns (default).

Step 6.

Select **Design Unit: test_counter** and click **Load** to accept these settings.

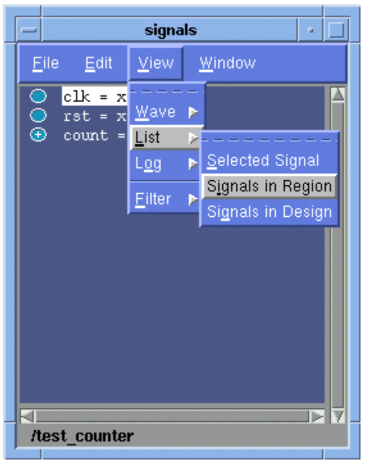
Step 7.

Bring up the Signals, List and Wave windows by entering the following line at the VSIM prompt within the Main window:

```
VSIM> view signals list wave
```

Step 8.

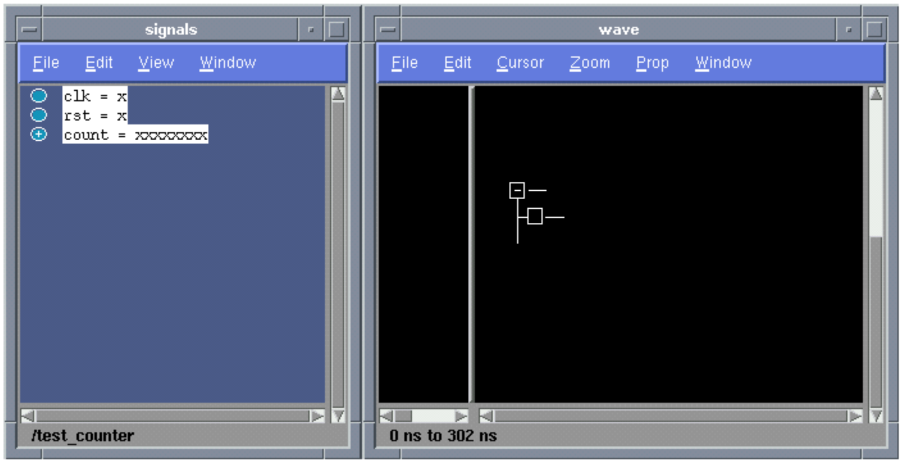
To list the top-level signals, move the pointer to the Signals window and make this View menu selection: **View > List > Signals in Region**.



Step 9.

Now let’s add signals to the Wave window with Model*Sim*’s drag and drop feature.

In the Signals window, **Control-click** on the *clk*, *rst*, and *count* signals to make a group selection. Click on the group one more time and drag it either pane of the Wave window.



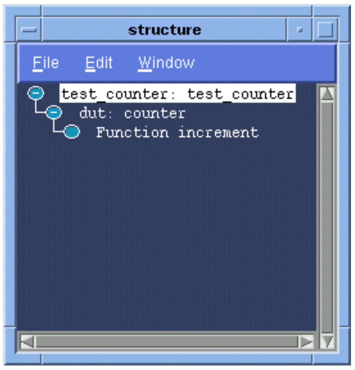
HDL items can also be copied from one window to another (or within the Wave and List windows) with the Edit > Copy and Edit > Paste menu selections. You can also delete selected items with the Edit > Delete selection.

Step 10.

Next open the Structure and Source windows. From the Main window make these menu selections: **View > Structure** and **View > Source**.

Step 11.

Rearrange the windows to give yourself a clear view of all open windows (VSIM, Signals, List, Wave, Structure and Source), then click inside the Structure window.



```
test_counter: test_counter
  dut: counter
    Function increment
```

Notice how this window describes the hierarchical structure of the design. In the illustration the Structure window shows three hierarchical levels: **test_counter**, **counter** and the function called **increment**. (If **test_counter** is not displayed you simulated **counter** instead of **test_counter**.)

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 Model*Sim*.

Step 12.

Click on **Function increment** and notice how other VSIM 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 window 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 window.

Step 13.

Now you will exercise different Run functions. They are found in the Main window under the **Run** menu. (You may need to expand the window to view all of the menus.)

Step 14.

Select **Run > 100 ns**. This causes the simulation to run and then stop after 100 ns (the default simulation length).

Instead of clicking the **RUN** button you could also type the **run** command.
To try it now, type:

```
run 500
```

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

Step 15.

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

```
run @ 3000
```

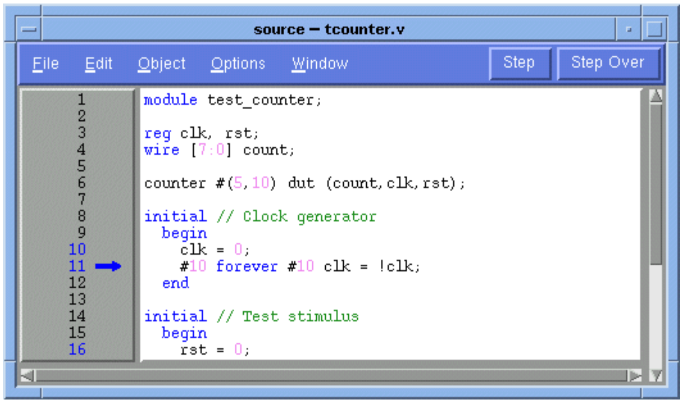
This advances the simulation to time 3000ns. Note that the simulation actually ran for 2400ns (3000 - 600).

Step 16.

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

Step 17.

Select the **Break** button; an arrow in the Source window points to the next HDL statement to be executed.



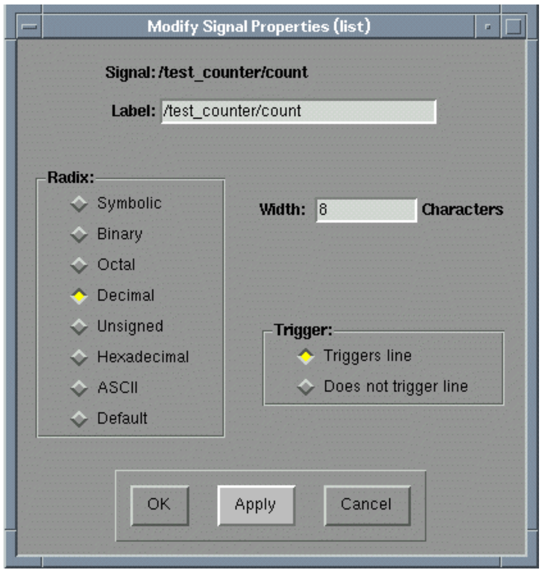
```
module test_counter;

reg clk, rst;
wire [7:0] count;

counter #(5,10) dut (count,clk,rst);

initial // Clock generator
  begin
    clk = 0;
    #10 forever #10 clk = !clk;
  end

initial // Test stimulus
  begin
    rst = 0;
```

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

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

Step 18.

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.



Signal: /test_counter/count

Label: /test_counter/count

Radix: Symbolic, Binary, Octal, [✓ Decimal], Unsigned, Hexadecimal, ASCII, Default

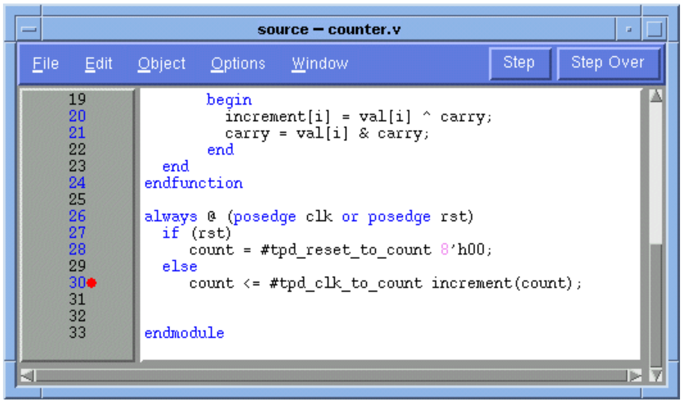
Width: 8 Characters

Trigger: [✓ Triggers line], Does not trigger line

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.

Step 19.

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 window. Move the cursor to the Source window and scroll the window to display line 30. Click at or near line number 30 to set a breakpoint (the executable line number is displayed in green).



```
19          begin
20            increment[i] = val[i] ^ carry;
21            carry = val[i] & carry;
22          end
23      end
24  endfunction
25
26  always @ (posedge clk or posedge rst)
27      if (rst)
28         count = #tpd_reset_to_count 8'h00;
29      else
30●        count <= #tpd_clk_to_count increment(count);
31
32
33  endmodule
```

You will see a dot appear next to the line number. This indicates that a breakpoint has been set for that line. Clicking line 30 once more would remove the breakpoint but don't do that now.

Step 20.

Select **Run** from the Main window toolbar to resume execution of the simulation. When the simulation hits the breakpoint, it stops running, highlights the Source window with an arrow, and issues a message in the Main window.

Step 21.

Typically when a breakpoint is reached you will be interested in one or more signal values. There are a few ways to determine this. You can:

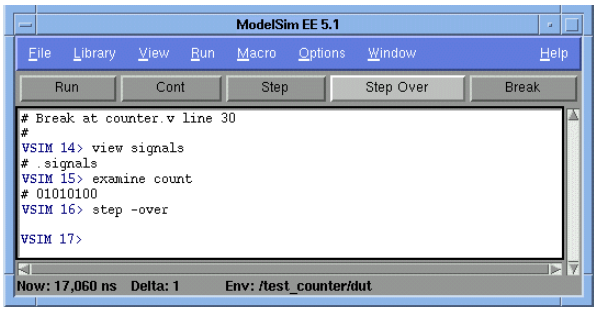
- look at the values shown in the Signals window, or
- examine the current value of the signal count by typing:

```
examine count
```

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

Step 22.

Let's move through the Verilog source functions with Model*Sim*'s Step and Step Over commands. Click **Step Over** on the toolbar.


ModelSim EE 5.1
File Library View Run Macro Options Window Help
Run Cont Step Step Over Break

```
# Break at counter.v line 30
#
VSIM 14> view signals
# .signals
VSIM 15> examine count
# 01010100
VSIM 16> step -over

VSIM 17>
```

Now: 17,060 ns   Delta: 1   Env: /test_counter/dut

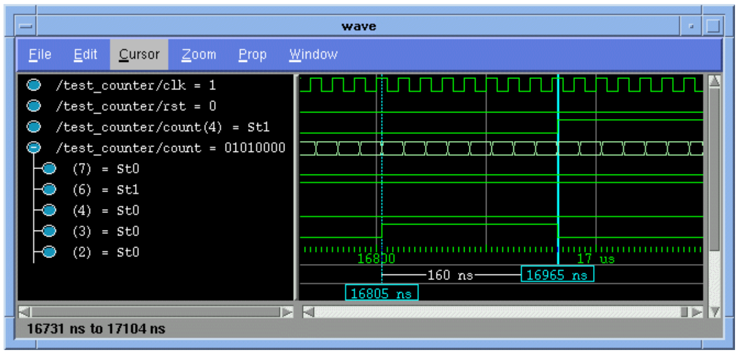
This causes the simulator to step over the function call on line 30. The Step button on the toolbar would have single-stepped the simulator, including each line of the increment function.

Step 23.
Experiment by yourself for awhile; setting and clearing breakpoints as well as Step'ing and Step Over'ing function calls until you feel comfortable with the operation of these commands.

Step 24.
Now let's get a Wave window view of the simulation; activate the Wave window.

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

Up to ten cursors can be present at the same time. Cursors are 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.



```
/test_counter/clk = 1
/test_counter/rst = 0
/test_counter/count(4) = St1
/test_counter/count = 01010000
  (7) = St0
  (6) = St1
  (4) = St0
  (3) = St0
  (2) = St0
```

Step 25.

Try adding or removing cursors with the **Cursor** > **Add Cursor | Delete Cursor** commands.

When you add a cursor, it is drawn in the middle of the display. Once you have more than one cursor, VSIM adds a delta measurement showing the time difference between the two cursor positions. The selected cursor is drawn as a solid line; all other cursors are drawn with dotted lines.

Step 26.

Click in the waveform display. Notice how 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 of the mouse pointer. You can position a cursor without snapping by dragging in the area below the waveforms.

Step 27.

Experiment with using the cursors, scrolling, and zooming (see page 155 for Wave window shortcuts).

Step 28.

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

```
VSIM> quit -force
```

Mixed VHDL/Verilog simulation

You must be using Model*Sim* EE/PLUS for this lesson.

The goals for this lesson are:

- compile multiple VHDL and Verilog files
- simulate a mixed VHDL and Verilog design
- list VHDL signals and Verilog nets and registers
- view the design in the Structure window
- view the HDL source code in the Source window

Step 1.
First, return to the directory you created in "Basic VHDL simulation" (p99).

```
cd <directory_name>
```

Next copy the VHDL and Verilog files to the directory (note the period specifying the current directory):

```
cp /<install_dir>/modeltech/examples/mixedHDL/*.vhd .
cp /<install_dir>/modeltech/examples/mixedHDL/*.v .
```

Step 2.
A group of Verilog files can be compiled in any order. Note, however, in a mixed VHDL/ Verilog design the Verilog files must be compiled before the VHDL files. Compile the Verilog files with this command:

```
vlog cache.v memory.v proc.v
```

Step 3.
Depending on the design, the compile order of VHDL files can be very specific. In the case of this lesson, the file *top.vhd* must be compiled last. Compile the VHDL files with this command:

```
vcom util.vhd set.vhd top.vhd
```

Step 4.

Invoke **vsim** with the name of the top level design:

```
vsim top
```

Step 5.

At the VSIM prompt type:

```
VSIM> view *
```

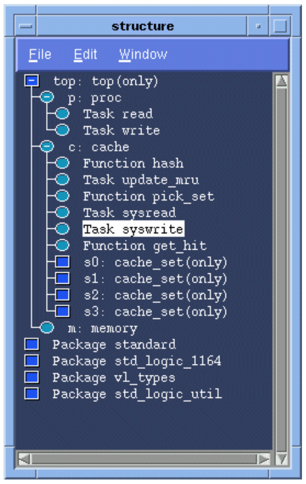
Step 6.

This time you will use the VSIM command line to add all of the HDL items in the region to the List and Wave windows:

```
VSIM> add list *
VSIM> add wave *
```

Step 7.

Take a look at the Structure window. Notice the hierarchical mixture of VHDL and Verilog in the design. VHDL levels are indicated by a square “prefix”, while Verilog levels are indicated by a circle “prefix.” Try expanding (+) and contracting (-) the structure layers. You’ll find Verilog modules have been instantiated by VHDL architectures, and similar instantiations of VHDL items by Verilog.



```
top: top(only)
  p: proc
    Task read
    Task write
  c: cache
    Function hash
    Task update_mru
    Function pick_set
    Task sysread
    Task syswrite
    Function get_hit
    s0: cache_set(only)
    s1: cache_set(only)
    s2: cache_set(only)
    s3: cache_set(only)
  m: memory
Package standard
Package std_logic_1164
Package vl_types
Package std_logic_util
```

Let’s take another look at the design.

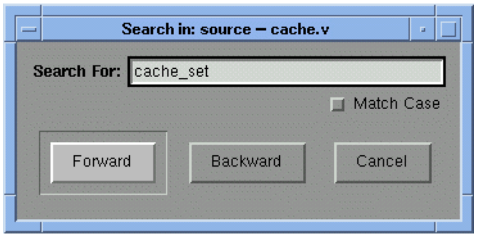
Step 8.

In the Structure window, click on the Verilog module **c: cache**.

The source code for the Verilog module is now shown in the Source window.

Step 9.

We’ll use Model*Sim*’s Find/Search function to find the declaration of cache_set within *cache.v*. From the Source window menu select: **Edit > Find**; the Search dialog box is displayed.

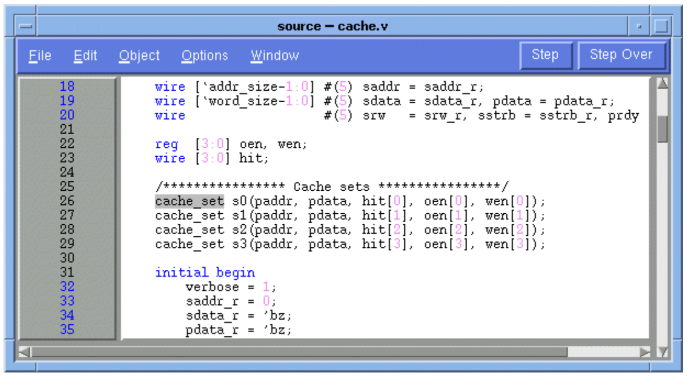


Search For: cache_set

Match Case

In the **Search For:** field type **cache_set** and click **Forward**. The cache_set declaration is now displayed in the Source window.

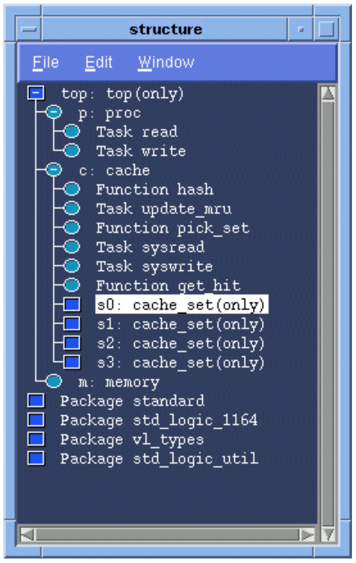
Note that the declaration of cache_set is a VHDL entity instantiated within the Verilog file *cache.v*.



```
    wire [`addr_size-1:0] #(5) saddr = saddr_r;
    wire [`word_size-1:0] #(5) sdata = sdata_r, pdata = pdata_r;
    wire                  #(5) srw   = srw_r, sstrb = sstrb_r, prdy

    reg  [3:0] oen, wen;
    wire [3:0] hit;

    /**************** Cache sets ****************/
    cache_set s0(paddr, pdata, hit[0], oen[0], wen[0]);
    cache_set s1(paddr, pdata, hit[1], oen[1], wen[1]);
    cache_set s2(paddr, pdata, hit[2], oen[2], wen[2]);
    cache_set s3(paddr, pdata, hit[3], oen[3], wen[3]);

    initial begin
        verbose = 1;
        saddr_r = 0;
        sdata_r = 'bz;
        pdata_r = 'bz;
```

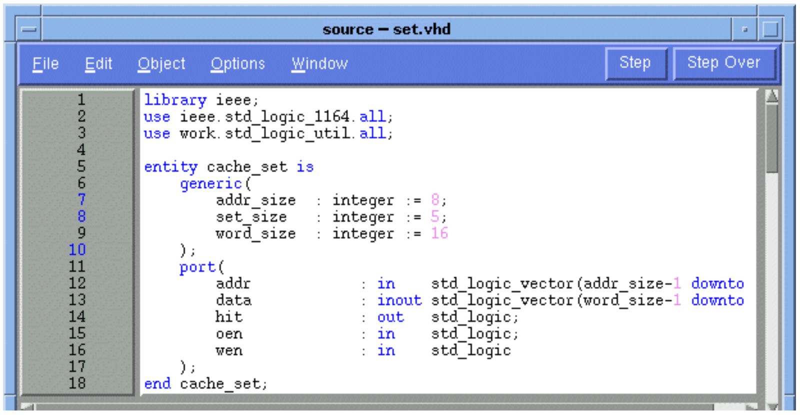
Step 10.

Now click on the line **"s0:cache_set(only)"** in the Structure window.



```
top: top(only)
  p: proc
    Task read
    Task write
  c: cache
    Function hash
    Task update_mru
    Function pick_set
    Task sysread
    Task syswrite
    Function get_hit
    s0: cache_set(only)
    s1: cache_set(only)
    s2: cache_set(only)
    s3: cache_set(only)
  m: memory
Package standard
Package std_logic_1164
Package vl_types
Package std_logic_util
```

The Source window now shows the VHDL code for the cache_set entity.



```
library ieee;
use ieee.std_logic_1164.all;
use work.std_logic_util.all;

entity cache_set is
    generic(
        addr_size   : integer := 8;
        set_size    : integer := 5;
        word_size   : integer := 16
    );
    port(
        addr             : in    std_logic_vector(addr_size-1 downto
        data             : inout std_logic_vector(word_size-1 downto
        hit              : out   std_logic;
        oen              : in    std_logic;
        wen              : in    std_logic
    );
end cache_set;
```

Before you quit, try experimenting with some of the commands you've learned from Lesson 1. Note that in this design, "clk" is already driven, so you won't need to use the **force** command.

Step 11.

When you're ready to quit simulating, enter the command:

```
VSIM> quit -force
```

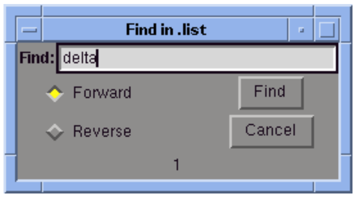
Finding names, and searching for values

You can easily locate HDL item names and values within Model*Sim*'s windows. Start any of the lesson simulations to try out the Find and Search functions illustrated below.

Finding items by name in tree windows

You can find HDL item names with the **Edit > Find** menu selection in these windows: List, Process, Signals, Source, Structure, Variables, and Wave windows.

Select **Edit > Find** to bring up the Find dialog box (List window version shown).

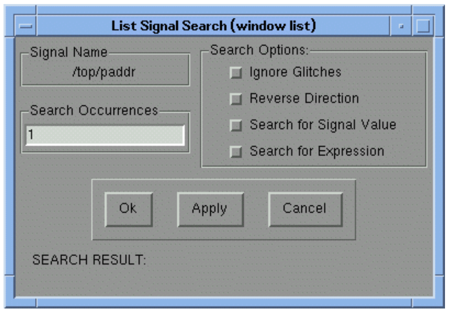


Find: delta

Enter an item label and **Find** it by searching **Forward** (right) or **Reverse** (left) through the window display.

Searching for item values in the List and Wave windows

You can search for HDL item values in the List, and Wave windows. Select **Edit > Search** from the window's menu to bring up the Signal Search dialog box (List window version shown).



The **List Signal Search** dialog box includes these options:

You can locate values for the **Signal Name: <item_label>** shown at the top of the dialog box. The search is based on these options (multiple **Search Options** may be selected):

- **Search Options: Ignore Glitches**
  Ignore zero width glitches in VHDL signals and Verilog nets.
- **Search Options: Reverse Direction**
  Search the list from bottom to top.
- **Search Options: Search for Signal Value**
  Activates the **Search Value** field; search for the value specified in the **Search Value** field, otherwise just look for transitions.
- **Search Options: Search for Expression**
  Activates the **Search Expression** field and the **Use Expression Builder** button; searches for the expression specified in the Search Expression field evaluating to a boolean true.

  The expression may involve more than one signal but is limited to signals logged in the List window. Expressions may include constants, variables, and macros. If no expression is specified, the search will give an error.

  See the *ModelSim EE/PLUS Reference Manual* for more information on expression syntax and the use of the Expression Builder.

- **Search Occurrences**
  You can search for the n-th transition or the n-th match on value; **Search Occurrences** indicates the number of transitions or matches for which to search.
- **Search Value**
  Valid only if **Use signal value** is selected; specifies the search value; must be formatted in the same radix as displayed.

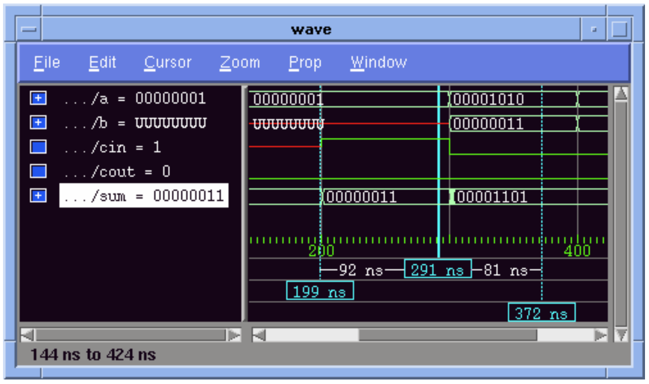
The result of your search is indicated at the bottom of the dialog box (**Found 2 matches: last one at time 2265 delta 0** in the illustration above).

Using the Wave window

This practice involves the use of Wave window time cursors, zooming the waveform display, and Wave window keyboard shortcuts. Any of the previous lesson simulations may be used with this practice.

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.You can add additional cursors to the waveform pane with the **Cursor > Add Cursor** menu selection. The selected cursor is drawn as a solid line; all other cursors are drawn with dotted lines. Remove cursors by selecting them and choosing using the **Cursor > Delete Cursor** menu selection.



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. VSIM also adds a delta measurement showing the time difference between the two cursor positions.

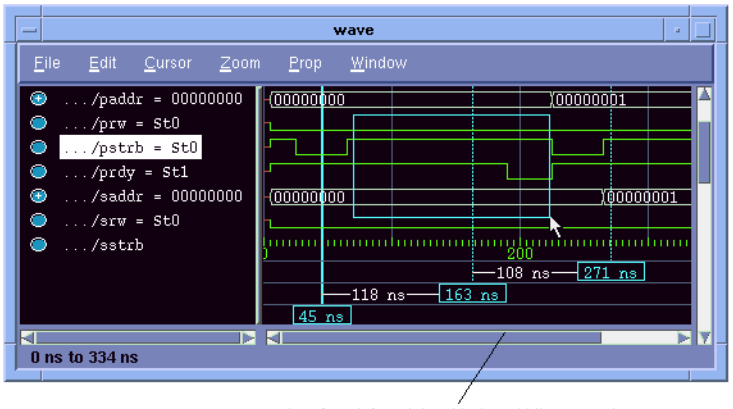
If you click in the waveform display, the cursor closest to the mouse position is selected and then moved to the mouse position. Another way to position multiple cursors is to use the mouse in the time box tracks at the bottom of the display. Clicking anywhere in a track selects that cursor and brings 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. You can control the snap distance from "Wave category" in the dialog box available from the Wave window **Properties > Display** menu selection.

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

Zooming - changing the waveform display range

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



drag from left to right with the middle mouse button to zoom

Using the Zoom menu

You can use the Wave window menu bar, or call up a **Zoom** menu window with the right mouse button in the right windowpane. The 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, cropping the view on the right. The starting time is held static.
- **Zoom Out**
  Zooms out by a factor of two, decreasing the resolution and increasing the visible range horizontally, extending the view on the right. The starting time is held static.
- **Zoom Last**
  Restores the display to where it was before the last zoom operation.
- **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 mouse

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

Zooming keyboard shortcuts

See "Wave window keyboard shortcuts" (p155) for a complete list of Wave window keyboard shortcuts.

Zooming with VSIM commands

The **.wave.tree zoomfull** provides the same function as the **Zoom > Zoom Full** menu selection and the **.wave.tree zoomrange** provides the same function as the **Zoom > Zoom Range** menu selection.

Use this syntax with the **.wave.tree zoomrange** command:

Syntax

```
.wave.tree zoomrange f1 f2
```

Arguments

`f1 f2`
Sets the waveform display to zoom from time f1 to f2, where f1 and f2 are floating point numbers.

Wave window keyboard shortcuts

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; search within the specified field in the wave-name pane for text strings

Continuing with Model*Sim* for workstations

More information on Model*Sim* commands, functions and techniques for use can be found in the following locations:

- **Model*Sim* EE/PLUS Reference Manual**
  an Adobe Acrobat (.pdf) version of our reference manual is available in the *<install_dir>/modeltech/docs* directory after installation
- **Tips and Techniques appendix of the reference manual**
- **technote text files installed with Model*Sim***
  located in the *<install_dir>/modeltech/docs* directory

A - Help, Updates, and Licensing

Help

Technical Support for Mentor Graphics Customers

For customers who purchased from Mentor Graphics in North America, the support line number is 800-547-4303. The web site is http://supportnetweb.mentorg.com.

For customers who purchased from Mentor Graphics outside of North America, please contact your local support organization.

Technical Support for Model Technology Customers

For customer who purchased from Model Technology, please contact Model Technology via email: support@model.com. The telephone number for the support line is 503-641-1340 from 8:00 AM to 5:00 PM Pacific Time. Be sure to have your server hostID or hardware security key serial number handy. You can also use our web site for technical support: http://www.model.com.

Technical Support for Other Channels

For customers who purchased Model*Sim* as part of a bundled product from an OEM or VAR, support is available at the following:

- **Annapolis Microsystems**
  email: wftech@annapmicro.com
  telephone: 410-841-2514
  web site: http://www.annapmicro.com
- **Exemplar Logic**
  email: support@exemplar.com
  telephone: 510-789-3333
  web site: http://www.exemplar.com
- **Hewlett Packard EEsof**
  email: hpeesof_support@hp.com
  telephone: 1-800-HPEESOF (1-800-473-3763)
  web site: http://www.hp.com/go/hpeesof

- **Lucent Technologies**
  Bell Labs Design Automation
  email: info@blda.lucent.com
  telephone: 800-875-6590
  web site: http://www.bell-labs.com/org/blda
- **Synplicity**
  email: support@synplicity.com
  telephone: 408-617-6000
  web site: http://www.synplicity.com

Updates

Getting the latest version information

If you want the latest version information via email from Model Technology, make sure your email address is current in our customer database. If you think it is not, send email to license@model.com. Make the subject line read "register email address". Put your hostID or hardware security key number and your email address in the body of the message.

Getting the latest version

You can ftp the latest version of the software from the web site at ftp.model.com. Instructions are there as well. A valid license file from either Model Technology or Mentor Graphics is needed to uncompress the Model*Sim* EE files. A password from Model Technology is needed to uncompress the Model*Sim* PE files. Contact license@model.com if you are a current PE customer and need a password.

Model*Sim* EE Licensing

Model*Sim* EE Licenses from Model Technology

Starting with 5.0, Model*Sim* workstation products use a version of Globetrotter Software’s license manager with a FEATURE line that looks like this:

```
FEATURE vsim modeltech 1997.090 01-apr-1998 2 6C92577EC335F4C9568D ck=61
```

After the product description is the maintenance expiration date in the form yyyy.mmm. This date dictates which Model*Sim* software version can be run. Any software built before or on that date will run; nothing built afterward will run. Next on the line is the "stop date" with the form dd-mmm-yyyy. It is six months later than the maintenance expiration date.

No software, regardless of its build date, runs after the stop date. Next is the number of licenses, followed by the auth code itself, and lastly a checksum that can be used to determine if someone made a mistake transcribing a license file.

Maintenance Renewals

When maintenance is renewed, Model Technology will send a new authorization code that incorporates the new maintenance expiration date. If maintenance is not renewed, the authorization code will still permit the use of any version of the software built before the maintenance expired until the stop date is reached.

Server Changes

Model Technology charges a fee for server changes. The fees are based on the number of license files required and can be waived if server changes are made as licenses expire. Contact sales@model.com for more information.

ModelSim EE Licenses from Mentor Graphics

Model*Sim* is completely compatible with licenses issued by Mentor Graphics.

Getting help

The best thing is to email us a test case. Our email address is support@model.com. The next best thing is to copy/print the following form, fill it out, and fax it to us at 503-526-5473.

==

Model Technology Fax Support Form

Your name:

Your company:

Your phone number:

Your FAX number:

Your email address:

Model*Sim* Version:
(Use the Help About dialog box with Windows; type **vcom** for workstations.)

Description of the problem (please include the exact wording of any error messages):

PC hardware security key serial number:

Host ID of licence server for workstations:

B - Resources

Appendix contents

Books

Applications of VHDL to Circuit Design
Randolph E. Harr and Alec G. Stanculescu, Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061, USA. 1991.

Chip-Level Modeling with VHDL
James R. Armstrong, Prentice Hall, Englewood Cliffs, New Jersey 07632, USA. 1988.

The Designer's Guide to VHDL
Peter J. Ashenden, Morgan Kaufmann Publishers, Inc., 340 Pine Street, Sixth Floor, San Francisco, California 94104-3205, USA. 1995. (ISBN: 1-55860-270-4)

A Guide to VHDL Syntax
Jayaram Bhasker, Prentice Hall, Englewood Cliffs, New Jersey 07632, USA. 1995.

IEEE Standard VHDL Language Reference Manual
ANSI/IEEE Std 1076-1993. Published by The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, New York 10017-2394, USA. ISBN:1-55937-376-8, June 1994. To purchase the standard, write IEEE Customer Service, Hoes Lane, Tiscataway NJ 08855-1331, or phone 800-678-4333 (908-562-5420 from outside the U.S.), 908-981-9667 fax.

Practical Programming in Tcl and Tk
Brent Welch, Prentice Hall, Englewood Cliffs, New Jersey 07632, USA. 1995.

Tcl and the Tk Toolkit
John K. Ousterhout, published by Addison-Wesley Publishing Company, Inc. (ISBN 0-201-63337-X).

Verilog Language Reference Manual 2.0
Open Verilog International, 15466 Los Gatos Blvd., Suite 109-071, Los Gatos CA 95032
Phone: (408) 353-8899 , fax: (408) 353-8869 , email: ovi@netcom.com, Contact: Lynn Horobin

VHDL, 2nd Edition
Douglas L. Perry, McGraw-Hill, Inc., Professional Publishing Group, 11 West 19th Street, New York, New York 10011, USA. 1993.

VHDL: Analysis and Modeling of Digital Systems
Zainalabedin Navabi, Northeastern University, McGraw-Hill Publishing Company, College Division, 1221 Avenue of the Americas, New York, New York, 10020, USA. 1993.

VHDL Coding Styles and Methodologies
Ben Cohen, Kluwer Academic Publishers, Boston, Massachusetts, USA. 1995. (ISBN: 0-7923-9598-0)

VHDL: Hardware Description and Design
Roger Lipsett, Carl Schaefer, & Cary Ussery, Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061, USA. 1989.

VHDL Primer
Jayaram Bhasker, Prentice Hall, Englewood Cliffs, New Jersey 07632, USA. 1992.

Organizations

IEEE

The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, New York 10017-2394, USA. Fax:212-705-7453.

Open Verilog International

15466 Los Gatos Blvd., Suite 109-071, Los Gatos CA 95032
Phone: (408) 353-8899 , fax: (408) 353-8869 , email: ovi@netcom.com, web: http://www.ovi.org Contact: Lynn Horobin The SDF mapping rules described in the Model Development Specification are based on version 2.1 of OVI's Standard Delay Format Specification. This document is available from Open Verilog International.

VHDL International

407 Chester Street, Menlo Park, CA 94025-3718, USA. Phone: 800-554-2550 or 415-329-0578, fax: 415-324-3150. A consortium of VHDL tool vendors and users that exists to promote the usage and standardization of VHDL. For information about VHDL International membership and mailings, contact Conference Management Services.

VHDL International Users' Forum

407 Chester Street, Menlo Park, CA 94025-3718, USA. Phone: 800-554-2550 or 415-329-0578, fax: 415-324-3150. The users' group within VHDL International. This group currently holds meetings twice a year in the spring and fall, which are attended by several hundred delegates. For information about VIUF meetings, membership, newsletters, and bulletin board services, contact Conference Management Services.

Corporations & Consultants

Automata Publishing Company
1072 South Saratoga-Sunnyvale Road, BLDG A107, San Jose CA 95129 USA. Phone: 800-8-VIPER1 or 408-255-0705, fax 408-253-7916, email: info@apco.com. PCI VHDL simulation models. Contact: Mona Singh.

CAST Computer Aided Software Technologies, Inc.
24 White Birch Drive, Pomona NY 10970 USA. Phone: 914-354-4945, Fax: 914-354-0325; email: info@cast-inc.com, web: http://www.cast-inc.com, VHDL services including model development, FPGA and ASIC design kits, training, simulation consulting. Contact: Newton Abdalla.

Comit Systems Inc.
1250 Oakmead Parkway, Suite 210, Sunnyvale CA 94088 USA. Phone: 408-988-2988; fax: 408-988-2133; email: bala@comit.com. FPGA design, VHDL modeling for embedded systems and related areas. Contact: Balachandran Sreekandath.

Daemon Modeling
834 SW Westwood Drive, Portland OR 97201 USA. Phone: 503-977-0554, email: 74051.3421@compuserve.com. Model development with emphasis on processor models. Contact: Daemon Anastas.

DP Consulting
9015 NE 139th Street, Kirkland WA 98034 USA. Phone: 206-821-9124. Model development. Contact: Richard Pado.

EDA Solutions Pty. Ltd.
Level 3, South Tower, 1-5 Railway Street Chatswood NSW 2067, Australia. Phone: 011 61 2 9413-4611, fax: 011 61 2 9413-4622, email: info@eda.com.au. Provides VHDL training, consulting services, and VHDL model development. Contact: Alain Legrand.

EEC Consultants, Inc.
3243 Springwood Drive, Clearwater FL 34621 USA. Phone: 813-786-2929. VHDL models. Contact: Kent Ulrich.

Hardi Electronics AB

Box 966, S220 09 Lund, Sweden. Phone: (46) 11-77-90; fax: (46) 13-15-99. Provides VHDL training, consulting services, and product distribution. Contact: Lars-Eric Lundgren.

LEDA SA

35 Avenue du Granier, 38240 Meylan, France. Phone: (33) 76-41-92-43. fax: (33) 76-41-92-44. Provides VHDL training, consulting services, and product distribution. Contact: Francis Sourbier.

Logic Modeling

19500 NW Gibbs Drive, Beaverton OR 97006 USA. Phone: 503-690-6900, fax: 503-690-6906. Provides modeling libraries.

Paolo Tabacco

Via Dei Berio 91, Rome Italy (00155). Phone: 39-6-22595-306; fax: 39-6-2280739; email: tabacco@space.it. Designs for boards and ASICs using Model*Sim* and Exemplar CORE in aerospace and commercials applications. Contact: Paolo Tabacco.

Papillon Research Corporation

52 Domino Drive, Concord MA 01742 USA. Phone: 508-371-9115; fax: 508-371-9175; email: info@papillonres.com; internet: http://www.papillonres.com/~papillon. Full product design including ASIC, FPGA, synthesis and modeling, analog, EMI and mechanical engineering. Contact: Dave Matthews.

Proxy Modeling

1580 Washington Blvd., Fremont CA 94539 USA. Phone: 510-440-VHDL. Provides modeling and consulting. Contact Keith G. Irwin or Pam G. Rissmann.

Saros Technology Limited

St. Michaels House, 94 High Street, Wallingford, Oxon OX10 OBW, United Kingdom. Phone: 011-44-1491-837787, fax: 011-44-1491-837477, email: carey@saros.co.uk, internet: http://www.saros.co.uk. Providers of Model*Sim* support in the UK. Developers of the VHDL editor, Turbo Writer. Contact: Carey Sayer

Seva Technologies Inc.
200 Brown Road, Suite 103, Fremont, CA 94539 USA.
Phone: 510-249-9085, fax: 510-249-9082, email: lfs@seva.com.
VHDL training, consulting services. Contact: Larry Saunders.

System Simulation Solutions
1100 NW Compton Drive, Beaverton, OR 97006, USA.
Phone: 503-690-1200. VHDL modeling services. Contact: James B. Morris.

Topdown Design Solutions
71 Spit Brook Road, Suite 301, Nashua, NH 03060, USA.
Phone: 603-888-8811, fax: 603-888-7694, email: sales@topdown.com. VHDL training courses and consulting services are also available. Contact: Art Pisani.

TM Associates Training Specialists
14420 S. Kelmsley Drive Oregon City, OR 97045, Phone: 503-656-4457, fax: 503-656-4775, email: twille@europa.com HDL training specialists. Contact: Tom Wille

TransGate Technologies
19310 Oak Street, Aloha OR 97007-4422 USA. Phone: 503-591-1232, email: ptkwt@teleport.com. Model development with emphasis on communications and application-specific memories, ASIC/PLD design. Contact: Phillip Tomson.

VAutomation Inc.
20 Trafalgar Square, Suite 443, Nashua, NH 03063 USA. Phone: 603-882-2282; fax: 603-882-1587; email: eric@vautomation.com; internet: http://www.vautomation.com Sythesizable HDL models. Contact: Eric Ryherd.

VHDL Technology Group
100 Broadhead Road, Suite 140, Bethlehem, PA 18017 USA.
Phone: 610-882-3130, fax: 610-882-3133, email: wdb@vhdl.com. VHDL training and consulting services. Contact: William D. Billowitch.

Widman Associates
70 Crestmont Drive, Oakland, CA 94619 USA. Phone: 510-482-3564,
fax 510-482-5339. Contact: Bob Widman.

Willamettte HDL, Inc.
14314 SW Allen Blvd. Suite 625, Beaverton, OR 97005 USA. Phone: 503-590-8499, fax 503-645-9728, email: info@whdl.com, internet: http://www.whdl.com. Top-down design methodologies for system, board, and ASIC design.

Online resources

email:
omf_info@cfi.org

newsnet news groups:
sci.electronics.cad
comp.arch.fpga
comp.lang.tcl
comp.lang.verilog
comp.lang.vhdl
comp.lsi
comp.lsi.testing

home pages:
http://www.model.com
http://www.vhdl.org/vhdl_intl
http://www.e2w3.com
http://www.translogiccorp.com
http://www.translogic.nl

web references:
Tcl/Tk man pages: http://www.elf.org/tcltk-man-html/contents.htm
SunScript (Tcl information): http://sunscript.sun.com
Open Verilog International: http://www.ovi.org

Index