

Discovery Visual Environment User Guide

Version C-2009.06
June 2009

Comments?

E-mail your comments about this manual to:

vcs_support@synopsys.com.

SYNOPSYS[®]

Copyright Notice and Proprietary Information

Copyright © 2008 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

Right to Copy Documentation

The license agreement with Synopsys permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:

This document is duplicated with the permission of Synopsys, Inc., for the exclusive use of _____ and its employees. This is copy number _____. ”

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Registered Trademarks (®)

Synopsys, AMPS, Cadabra, CATS, CRITIC, CSim, Design Compiler, DesignPower, DesignWare, EPIC, Formality, HSIM, HSPICE, iN-Phase, in-Sync, Leda, MAST, ModelTools, NanoSim, OpenVera, PathMill, Photolynx, Physical Compiler, PrimeTime, SiVL, SNUG, SolvNet, System Compiler, TetraMAX, VCS, Vera, and YIELDirector are registered trademarks of Synopsys, Inc.

Trademarks (™)

AFGen, Apollo, Astro, Astro-Rail, Astro-Xtalk, Aurora, AvanWaves, Columbia, Columbia-CE, Cosmos, CosmosEnterprise, CosmosLE, CosmosScope, CosmosSE, DC Expert, DC Professional, DC Ultra, Design Analyzer, Design Vision, DesignerHDL, Direct Silicon Access, Discovery, Encore, Galaxy, HANEX, HDL Compiler, Hercules, Hierarchical Optimization Technology, HSIMplus, HSPICE-Link, iN-Tandem, i-Virtual Stepper, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, Liberty, Libra-Passport, Library Compiler, Magellan, Mars, Mars-Xtalk, Milkyway, ModelSource, Module Compiler, Planet, Planet-PL, Polaris, Power Compiler, Raphael, Raphael-NES, Saturn, Scirocco, Scirocco-i, Star-RCXT, Star-SimXT, Taurus, TSUPREM-4, VCS Express, VCSi, VHDL Compiler, VirSim, and VMC are trademarks of Synopsys, Inc.

Service Marks (SM)

MAP-in, SVP Café, and TAP-in are service marks of Synopsys, Inc.

SystemC is a trademark of the Open SystemC Initiative and is used under license.
ARM and AMBA are registered trademarks of ARM Limited.
Saber is a registered trademark of SabreMark Limited Partnership and is used under license.
All other product or company names may be trademarks of their respective owners.

Contents

1. Getting Started

Overview	1-2
General Requirements	1-2
Enabling Debugging	1-2
Compile-time Options	1-3
Required Files	1-4
Invoking DVE	1-5
Informational	1-5
64-bit Platform Support	1-5
Post Process	1-5
Interactive	1-6
Scripts	1-7
DVE Log Files	1-7
Running a Simulation from the Command Line	1-8
VCS MX and VHDL	1-8
Pure VHDL	1-8
Mixed Simulation with Verilog on Top	1-8
Mixed Simulation with VHDL on Top	1-9
Methodology for Checkpoint Restore	1-9

Passing DVE Command Line Arguments from Simulator Runtime Command Line	1-10
Arguments not Supported	1-12
Running a Simulation from the GUI.	1-12
Starting an Interactive Session	1-12
Using the Console Pane	1-14
Loading VPD Files.	1-15
Saving a Session or Layout.	1-15
Loading a Saved Session	1-17
Closing a Database	1-17
Exiting DVE	1-18
 2. Using the Graphical User Interface	
Overview of DVE Window Configuration.	2-2
Managing DVE Panes and Views	2-4
Managing Target Views	2-4
Docking and Undocking Views and Panes	2-6
Dragging and Dropping Docked Windows	2-7
Using the Menu Bar and Toolbar.	2-7
Setting the Simulation Time.	2-8
Searching Instances of Signal.	2-8
Using Filters	2-10
Editing Preferences	2-11
 3. Using the Hierarchy and Data Panes	
The Hierarchy Pane	3-1

Scope Types and Icons	3-4
Navigating Open Designs	3-5
Filtering the Data in the Hierarchy Pane	3-5
Expanding and Collapsing the Scope	3-6
Rearranging Columns in the Hierarchy Pane	3-6
Populating Other Views and Panes	3-7
Displaying Variables in the Data Pane	3-7
Dragging and Dropping Scopes	3-7
Using Context-Sensitive Menu.	3-9
Dumping Signal Values	3-10
Forcing Signal Values	3-11
The Data Pane	3-14
Viewing Signals and Values.	3-14
Viewing Interfaces as Ports	3-15
Viewing \$unit Signals.	3-17
The Watch Pane	3-19
 4. Using Source Views	
Loading Source Code	4-2
Loading a Source View from the Hierarchy Pane	4-2
Loading a Source view from the Assertion View	4-3
Displaying Source Code from a File	4-4
Using the Mouse in the Source View.	4-5
Working with the Source Code	4-5
Expanding and Collapsing Source Code View	4-6
Editing Source Code	4-6
Selecting and Copying Text to the Clipboard.	4-6
Navigating the Design from the Source View	4-7

Navigating Code in Interactive Simulation	4-8
Setting Breakpoints in Interactive Simulation	4-8
Managing Breakpoints	4-11
Annotating Values	4-13
 5. Using Wave Views	
Viewing Waveform Information	5-2
Set the Target View	5-2
Viewing a Waveform	5-3
Viewing Nanosim Analog Signals	5-4
Using the Signal Pane	5-5
Expanding Verilog Vectors, Integers, Time, and Real Numbers	5-6
Renaming Signal Groups	5-7
Creating Multiple Groups when Adding Multiple Scopes	5-7
Filtering Signals	5-8
Adding Signal Dividers	5-8
Customizing Duplicate Signal Display	5-8
Using User-defined Radices	5-9
Managing User-defined Radix	5-9
Using the Waveform Pane	5-11
Customizing Waveforms Display	5-13
Overlapping Analog Signals	5-16
Deleting Signal Group	5-16
Cursors and Markers	5-17
Using Cursors	5-17
Creating Markers	5-19
Zooming In and Out	5-21
Drag Zooming	5-22

Visualizing X at all Zoom Levels	5-22
Expanding and Contracting Wave Signals.	5-23
Searching Signals	5-23
Comparing Signals, Scopes, and Groups	5-24
Creating a Bus	5-28
Modifying Bus Components	5-30
Editing the Bus Bit Range	5-31
Creating an Expression	5-32
Viewing Bus Values	5-34
Creating and Updating Counters.	5-34
Shifting Signals	5-36
Printing Waveforms	5-37
 6. Using the List View	
The List View	6-2
Displaying Data	6-3
Dragging and Dropping Signals into the List view	6-4
Opening a Database	6-4
Loading a Session	6-5
Navigating Simulation Data	6-5
Viewing Data in the List view	6-6
Using Markers	6-7
Setting Signal Properties	6-8
Comparing Signals	6-9
Saving a List Format	6-10
 7. Using Schematics	
Overview	7-2

Viewing Schematic	7-2
Opening a Design Schematic View	7-3
Selecting a Signal in the Schematic View	7-5
Annotating Values	7-6
Opening a Path Schematic View.	7-6
Displaying Connections in a Path Schematic	7-8
Finding Signals in a Schematic View.	7-10
Manually Tracing Signals.	7-10
Searching for Signals.	7-10
Following a Signal Across Boundaries	7-11
Tracing X Values in a Design	7-13
Printing Schematics.	7-15
 8. Working with Assertions and Cover Properties	
Compiling SystemVerilog Assertions.	8-1
Viewing Assertion Results.	8-2
Setting Display Criteria	8-3
Debugging Assertions	8-5
Viewing an Assertion in the Wave view	8-7
Viewing an Assertion Failure Time Delta	8-8
Navigating Assertions in the Wave view	8-10
Navigating Source Code	8-11
 9. Tracing Drivers and Loads	
Overview of Tracing Functionality	9-1
Supported Functionality	9-3
Trace Drivers.	9-4

Trace Loads.	9-5
10. DVE Testbench Debugger	
Overview	10-1
Enabling Testbench for Debugging	10-2
Using the Testbench Debugger GUI	10-3
Understanding the Testbench Debugger GUI	10-3
Testbench Debugger Panes	10-4
Stack Pane	10-5
Local Pane	10-8
Watch Pane	10-8
Testbench Debugger Menu Options	10-10
View Menu	10-10
Signal Menu	10-10
Simulator Menu	10-11
Window Menu.	10-12
Setting Testbench Debugger GUI Preferences	10-12
11. Using the C, C++, and SystemC Debugger	
Getting Started	11-16
Using a Specific gdb Version	11-16
Attaching the C-Source Debugger in DVE.	11-16
Detaching the C-source Debugger.	11-18
Displaying C-source Files in the Source Pane.	11-18
Commands Supported by the C Debugger	11-19
Using Breakpoints	11-26
Set a Breakpoint from the Breakpoints Dialog Box	11-26
Control Line Breakpoints in the Source view	11-26

Set a Breakpoint from the Command Line	11-26
Deleting a Line Breakpoint.	11-28
Stepping Through C-source Code	11-28
Stepping within C Sources:	11-29
Cross-stepping between HDL and C Code	11-29
Cross-stepping in and out of Verilog PLI Functions	11-30
Cross-Stepping in and out of VhPI Functions.	11-31
Cross-stepping from C into HDL:	11-32
Cross-Stepping in and out of SystemC Processes.	11-33
Direct gdb Commands	11-34
Add Directories to Search for Source Files	11-35
Common Design Hierarchy	11-36
Post-processing Debug Flow.	11-40
Interaction with the Simulator	11-42
Prompt Indicates Current Domain	11-42
Commands affecting the C domain:.	11-43
Combined Error Message:.	11-43
Update of Time, Scope and Traces	11-44
Configuring CBug	11-44
Startup Mode	11-44
Attach Mode.	11-45
cbug::config add_sc_source_info auto always explicit	11-45
Supported platforms	11-46
Using SYSTEMC_OVERRIDE.	11-47
Example: A Simple Timer	11-48

A. Menu Bar and Toolbar Options

Menu Bar Options	A-2
File Menu	A-2
Edit Menu	A-4
View Menu	A-6
Simulator Menu	A-10
Signal Menu	A-12
Scope Menu	A-15
Trace Menu	A-17
Window Menu	A-19
Help Menu	A-21
Editing Preferences	A-22
Keyboard Shortcuts	A-25
File Command Shortcuts	A-26
Edit Command Shortcuts	A-26
View Command Shortcuts	A-26
Simulator Command Shortcuts	A-27
Signal Command Shortcuts	A-27
Scope Command Shortcuts	A-27
Trace Command Shortcuts	A-27
Help Command Shortcuts	A-28
Toolbar Reference	A-28
Edit	A-28
File	A-29
Scope	A-30
Trace	A-31
View	A-32
Signal	A-33

Simulator	A-35
Time Operations	A-36
Zoom	A-36
Zoom and Pan History	A-37
Using the Context-sensitive Menu.	A-38
Schematic View Toolbar Options	A-40
Using the Schematic View CSM	A-40
Driver Pane Menu	A-43
Using the Command Line	A-43

1

Getting Started

This chapter describes getting started using DVE (Discovery Visual Environment) and includes the following topics:

- “Overview”
- “General Requirements”
- “Enabling Debugging”
- “Invoking DVE”
- “Running a Simulation from the Command Line”
- “Running a Simulation from the GUI”
- “Loading VPD Files”
- “Loading a Saved Session”
- “Closing a Database”

- [“Exiting DVE”](#)

Overview

Discovery Visual Environment (DVE) is an interactive GUI (Graphical User Interface) that you can use for debugging your design. You can use DVE for debugging your SystemVerilog, VHDL, Verilog, and System-C designs. You can simply drag-and-drop your signals in various views or use the menu options to view the signal source, trace drivers, compare waveforms, and view schematics.

General Requirements

You must use the same version of VCS and DVE to ensure problem-free debugging of your simulation. There are three ways to check the DVE version:

- Enter the `dve -v` command-line option
- Enter `gui_get_version` on the DVE command line
- Use the **Help > About** menu option

Enabling Debugging

This section describes how to enable debugging options for your simulation.

Note:

If you run DVE in a directory where you do not have file write privileges, DVE will be unable to write log files. In this case, you will receive a warning message indicating that DVE is unable to write files.

Compile-time Options

-debug_pp

Enables r/w access and callbacks to design nets, and enables memory callback and assertion debug.

In addition, -debug_pp enables VCS DKI and VPI routine usage. The -debug_pp is ideally suited for using DVE in post processing mode. You can also run interactive simulation when the design is compiled with the -debug_pp option, but certain capabilities like breakpoints and force will not be enabled.

-debug

Provides force net and reg capabilities in addition to all capabilities of the -debug_pp option. This option is best suited for interactive simulation.

-debug_all

Instruments all design capabilities in the design, and consequently adds significant compile time overhead. This option enables the same capabilities as the -debug option, and also enables setting breakpoints and stepping through the code.

This option is recommended only for interactive simulations where breakpoint and line-stepping capabilities are essential.

Required Files

DVE requires the following input files to enable its debug functionality:

- **VPD file** - VPD files are platform-independent, versioned files into which you can dump selected signals during simulation. DVE gets hierarchy, value change, and some assertion information from these files. You can perform debugging in post process using a VPD file.

However, VPD files are not guaranteed to contain the entire design hierarchy because you can selectively choose subsets of the design to be dumped to the VPD file.

For all DVE functionality to be available while debugging, it is imperative that the VCS version used to generate the VPD, and the DVE version used to view the VPD, are identical.

- **Assertion Library** - DVE uses this library for advanced assertion debugging. This library is produced when a design contains OVA/SVA/PSL assertions and the correct VCS compile options are used. The library is platform dependent.
- **Coverage databases** - In DVE, you specify one of three types of coverage databases to display coverage information. If other coverage databases for different types of coverage exist, DVE automatically opens them as well.

You can select either of the following two kinds of databases:

- A code coverage directory (by default, VCS and VCS MX names this directory, `simv.cm`).

- An OpenVera or SystemVerilog assertions database directory (by default, VCS names this directory, `simv.vdb`).

Invoking DVE

This section describes how to invoke DVE.

Informational

```
dve -help
```

Displays DVE basic commands.

```
dve -v | -V
```

Displays version information.

64-bit Platform Support

```
-full64
```

You can use the `-full64` option to invoke DVE in 64-bit mode. To use the `-full64` option, you must download and install the 64-bit VCS binaries. By default, DVE is invoked in 32-bit mode.

To activate 64-bit support enter the following:

```
dve -full64
```

Post Process

```
dve
```

Invokes an empty DVE top-level window with no arguments. From this point, DVE usage can be post-processing or interactive mode.

```
dve -vpd filename
```

Invokes DVE, reads and loads the specified VPD file, and opens the top-level scope for that design.

```
dve -vpd test.vpd -session mysession.tcl
```

Invokes DVE with the VPD file, `test.vpd`, and applies settings from the session file, `mysession.tcl`.

Interactive

```
dve -nogui
```

The DVE GUI is not displayed; instead UCLI mode is invoked and the simulator is not attached. To start the simulator, at the UCLI prompt, enter the following:

```
start <simv or executable name>
```

```
simv -ucli
```

Runs VCS/VCS MX for UCLI debugging. The DVE GUI is not displayed.

```
simv -gui
```

Opens DVE with the `simv` attached to simulator at time 0.

```
vcs -gui -R
```

Opens DVE with the `simv` attached to simulator at compile-time.

```
dve -toolexe name -toolargs simulator args
```

Invokes DVE, connects executable (*name*) to the simulator, and runs it with the arguments specified in the *args*.

Scripts

```
dve -cmd "cmd"
```

Invokes DVE and executes the Tcl command enclosed in quotation marks. You can specify multiple commands separated by semi-colons.

```
dve -script name
```

Invokes DVE and reads in a Tcl script specified by *name*.

```
dve -session name
```

Invokes DVE and reads in a session file. If the `-session` and `-script` options are combined, the session is read first and then the script.

DVE Log Files

DVE produces the following two log files in the `DVEfiles` directory, which gets created in the current working directory. These log files are useful in the event of a problem.

- `dve_gui.log` – contains all input and output to the console log
- `dve_history.log` – contains all commands that occur during the lifetime of a debug session; useful for capturing scripts for replay

Running a Simulation from the Command Line

To enable DVE or to set DVE as the default command-line interface, you must compile with `-debug`, `-debug_all` or `-debug_pp`. The option `-debug_pp` is for debugging in PP mode and the options `-debug` and `-debug_all` is for interactive debugging

To run DVE with VCS, enter VCS commands with options enabling DVE:

```
vcs (-debug | -debug_all | -debug_pp) [-sverilog] [-ntb]
    [VCS_options] design.v [testbench_files]
simv -gui [runtime_options]
```

VCS MX and VHDL

Pure VHDL

To run a VHDL simulation with DVE, enter the VCS MX commands with options enabling DVE:

```
vhdlan design.vhd
vcs cfg_tb (-debug | -debug_all)
simv -gui [runtime_options]
```

Mixed Simulation with Verilog on Top

To run a mixed Verilog/VHDL simulation with Verilog on top, enter the commands with options enabling DVE:

```
vlogan Verilog_files [options]
vhdlan vhd_filename -vlib Verilog
vcs (-debug | -debug_all) [options] design.v
simv -gui [runtime_options]
```

Mixed Simulation with VHDL on Top

To run a mixed Verilog/VHDL simulation with VHDL on top, enter the commands with DVE enabling options:

```
vlogan Verilog_files [options]
vhdlan vhdl_filename -vlib Verilog
vcs cfg_tb (-debug | -debug_all)
simv -gui
```

Methodology for Checkpoint Restore

When restoring a saved simulation, use the same technology or flow to restore that you used to save the checkpoint, for example:

- Save using UCLI commands and restore using UCLI commands
- Save in DVE and restore in DVE
- Save using SCL commands and restore using SCL commands
- Save using CLI commands and restore using CLI commands

Do not mix the technologies for saving and restoring, for example:

- Save using UCLI commands and restore using SCL commands
- Save in DVE and restore with UCLI commands
- Save using UCLI commands and restore using DVE
- Save using CLI commands and restore using UCLI commands

Also, if you are running an external application that communicates with VCS MX using the VHPI or PLI, and if there are files opened for this application, you must close these files before you save and open them again after you restore.

Passing DVE Command Line Arguments from Simulator Runtime Command Line

You can pass DVE command line options to DVE from the simv command line. This allows you to load previous session files and run custom Tcl scripts. For example, loading a session file with simv:

```
simv -gui -session=mysession.tcl
```

You can use the argument called `-dve_opt` to pass arguments from simv to DVE. Each DVE argument must be preceded by `-dve_opt` argument. In cases where the argument requires an additional option, the `=` sign should be used.

The following commands are for passing DVE command line arguments from simulator command line:

- DVE takes multiple arguments, for example, for printing `-v` and for logging off as follows:

```
dve -v -nolog
```

For VCS, use the following runtime command:

```
simv -gui -dve_opt -script=myscript.tcl -dve_opt -nolog
```

- DVE takes the following argument that requires an additional token, such as specifying a session file.

```
dve -session=mySession.tcl
```

For VCS, use the following runtime command:

```
simv -gui -dve_opt -session=mySession.tcl
```

- DVE takes the following argument that requires multiple additional tokens, such as the `-cmd` argument.

```
dve -cmd 'puts "hello world"'
```

For VCS, use the following runtime command:

```
simv -gui -dve_opt -cmd='puts "hello world"'
```

Arguments not Supported

The following arguments are not supported in DVE:

- `-vpd`: This command is not processed since the simulator already uses `inter.svpd` of file specified by `-vpd_file`, so `-vpd` is not supported.
- `-toolexe`, `-toolargs`: These commands are automatically produced by `simv`.
- `-servermode`: This command is not applicable for `simv`.
- `-full64`: This command is not supported, because when `simv` is generated using `-full64`, DVE will be in 64-bit mode by default.
- `-dbdir`: This command is not supported, because it is not used in interactive mode.
- `-ucli`: This command is not supported.

Running a Simulation from the GUI

You can invoke DVE and start the simulation from the GUI.

Starting an Interactive Session

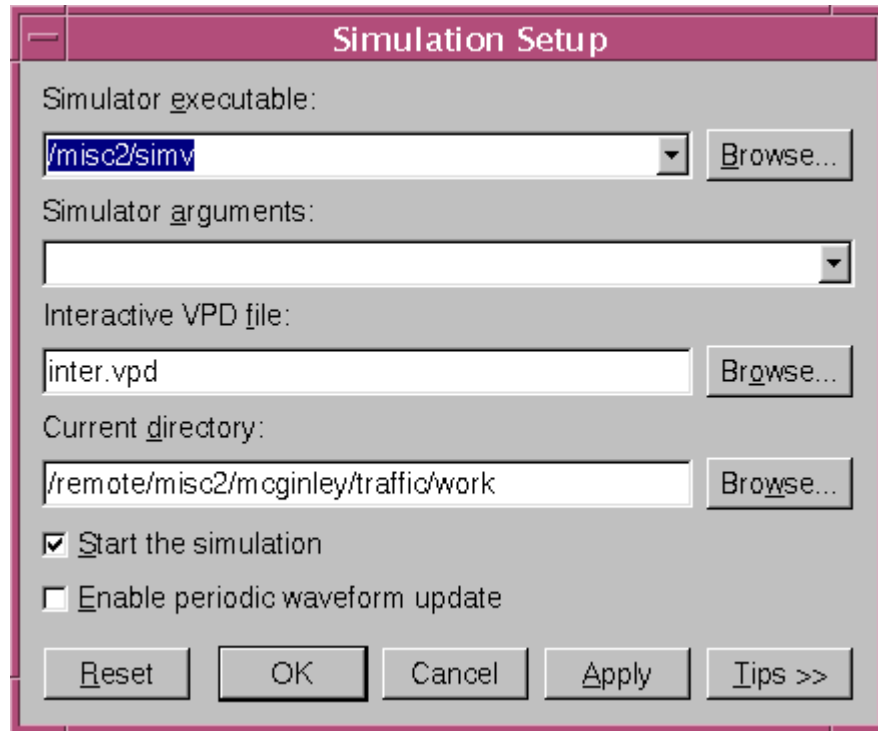
In addition to loading VPD files for post-processing, you can also setup and run a simulation interactively in real-time using a compiled Verilog, VHDL, or mixed design. To do so, perform the following procedure:

1. Invoke DVE from the command line:

`%dve`

2. Select **Simulator > Setup**.

The Simulation Setup dialog box opens.



3. Select the **Start the simulation** check box.
4. Browse and select a simulator executable.
5. Enter simulator arguments.
6. Set the name of the VPD file or select an existing file that will be written during this interactive session.
7. Click **OK**.

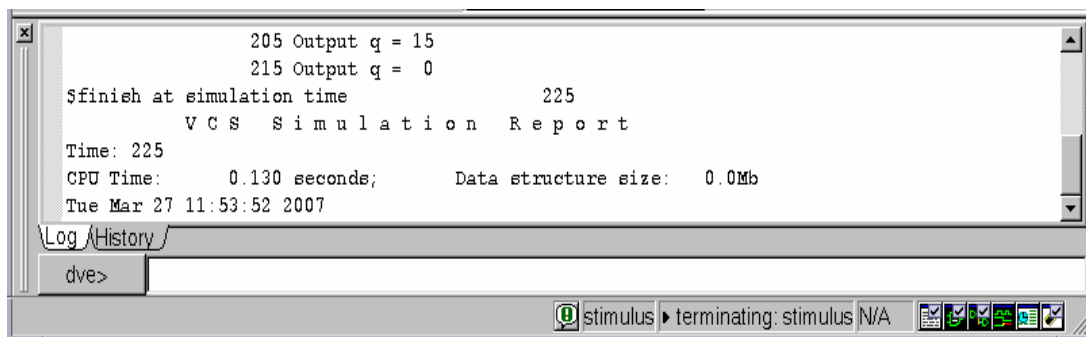
The simulation is set.

Using the Console Pane

When you start the simulation, DVE activates toolbar and menu commands for running and controlling the simulation. For more information about toolbar and menu bar commands, see the appendix entitled, “[Menu Bar and Toolbar Reference](#)” on page A-1.

Use the command line at the bottom of the DVE top-level window to enter DVE and Unified Command-line Interface (UCLI) commands. [Figure 1-1](#) shows the command line where you enter commands with the results displayed in the Log tab above the command line.

Figure 1-1 Command Line with the Log tab



To view DVE commands, enter the following:

```
help -gui
```

To quickly view the UCLI commands and their usage, enter one of the following commands at the DVE prompt:

```
help -ucli
```

Displays a list of UCLI commands and a short description.

```
help -ucli [argument]
```

Displays a description and the command syntax.

Loading VPD Files

You can load and display any number of VPD files for post-processing.

To load a VPD file

1. Select **File > Open Database**.

The Open Database dialog box appears.

2. Browse and select the name of the VPD file.
3. Enter or accept a Designator for your design.
4. Enter a time range to load.

The default range is the start of simulation to the end of simulation.

5. Click **Open**.

The VPD file is loaded.

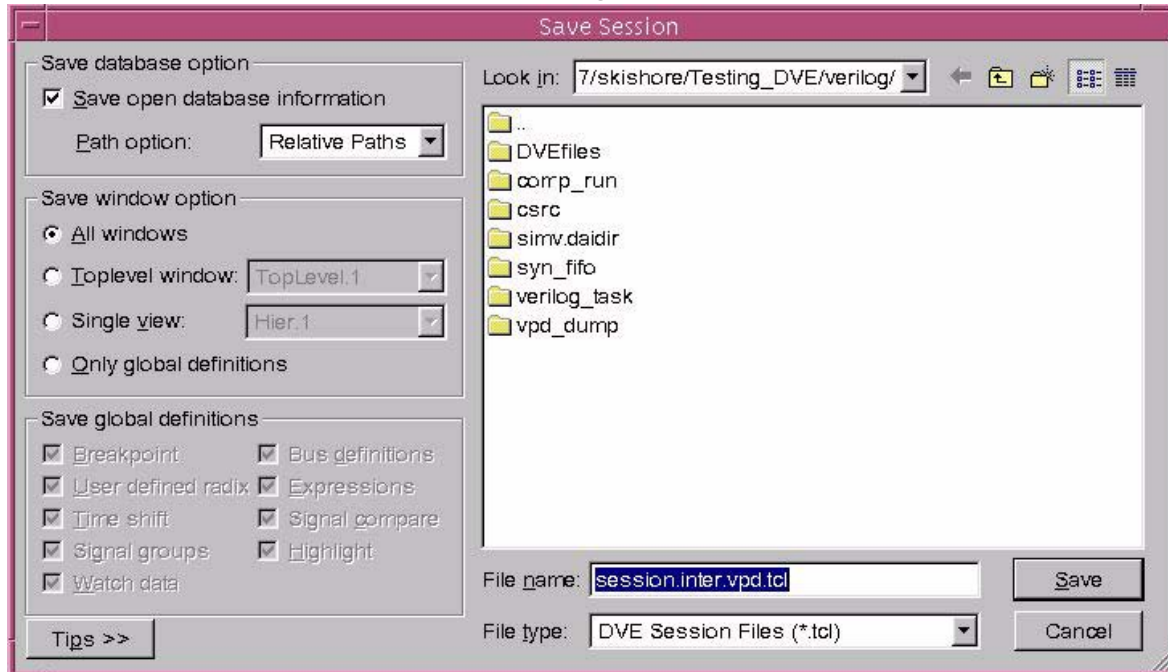
Saving a Session or Layout

You preserve session data display layout, and VPD path options using the Save Sessions dialog box.

To save a session

1. Select **File > Save Session**.

The Save Session dialog box appears.



2. Enter a File name.
3. Select a File type.
4. Select the option **Save open database information** if you want to save the database that you have currently opened in DVE.
5. Select a path option:
 - Relative Path - (default) Path for opened VPDs (relative to the directory where the session file is stored).
 - Full path - Absolute path for opened VPDs.
6. Select the window options as follows:
 - All windows - Saves all windows in the session.
 - Toplevel window - Saves only the top-level window.
 - Single view - Saves only a single view.

- Only global definitions - Saves the global definitions.

Enables the **Save global definitions** options. Select the available check boxes, as appropriate.

7. Click **Save**.

Loading a Saved Session

To load a saved session

1. Load a VPD file.
2. Select **File > Load Session**.

The Load Session dialog box appears.

3. Browse to the session and select it from the list of saved session Tcl files.
4. Click **Load**.

The session is loaded.

Closing a Database

To close a currently open database

1. Select **File > Close Database**.

The Close Database dialog box appears.

2. Make sure the correct database is selected, then click **OK**.

DVE closes the display of the selected database in the Hierarchy pane.

Exiting DVE

To exit DVE, select **File > Exit**.

2

Using the Graphical User Interface

This chapter describes the basic usage of the DVE graphical user interface and management of the windows, and includes the following topics:

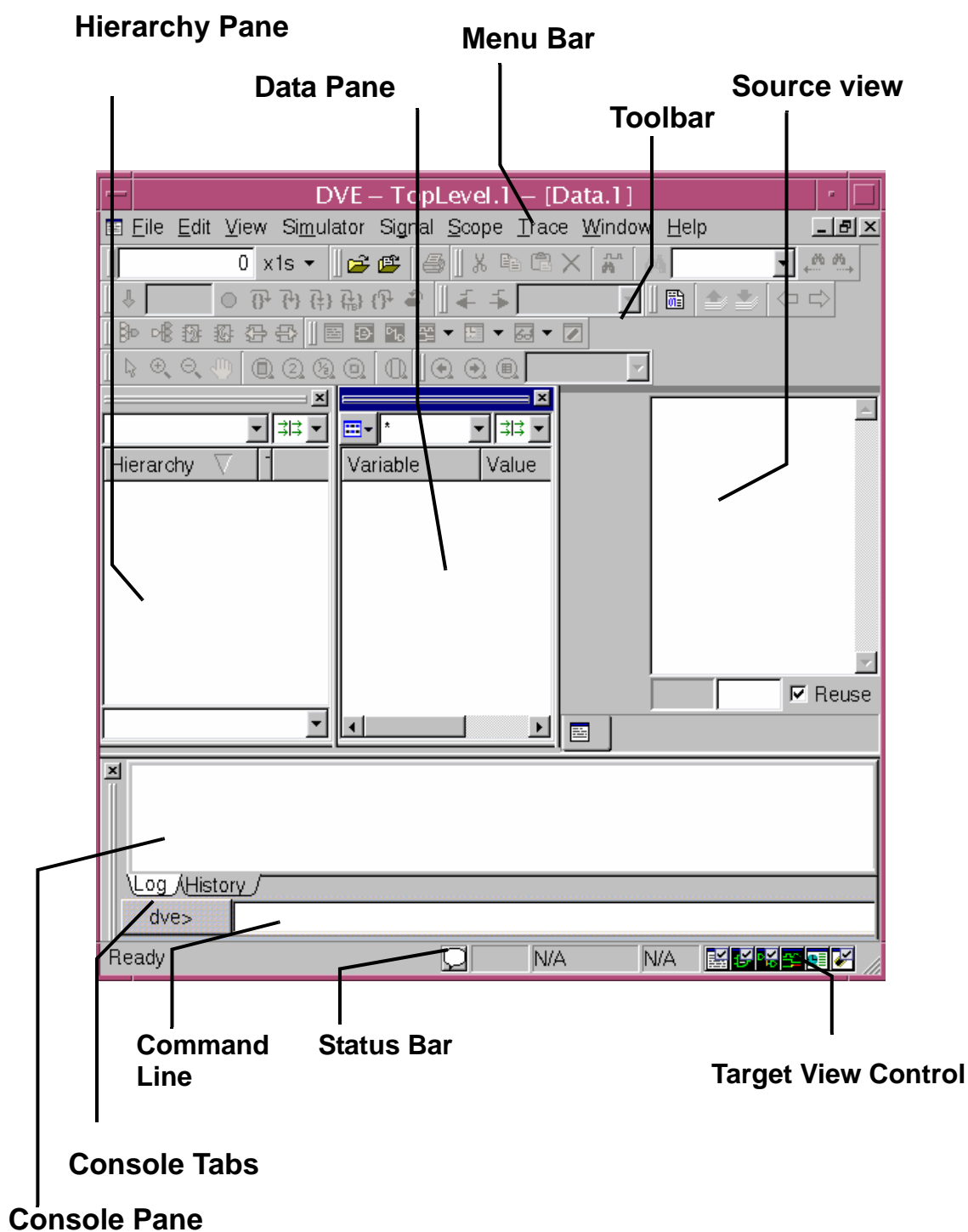
- [“Overview of DVE Window Configuration”](#)
- [“Managing DVE Panes and Views”](#)
- [“Using the Menu Bar and Toolbar”](#)
- [“Editing Preferences”](#)

Overview of DVE Window Configuration

DVE window model is based on the concept of the TopLevel window. A TopLevel window contains a frame, menus, toolbars, status bar, and pane targets. Any number of TopLevel windows are possible, however, at startup, the default number of windows is one.

A DVE TopLevel window is a frame which DVE uses for displaying design and debug data. The default DVE window displays the TopLevel window with the Hierarchy pane on the left, Data pane next to the Hierarchy pane, the Console pane at the bottom, and the Source view occupying the remaining space on the right. [Figure 2-1](#) shows the default TopLevel window.

Figure 2-1 Initial View of the DVE Top-level Frame



Note:

Watch pane appears only when you add variables or signals to it for monitoring purposes.

Managing DVE Panes and Views

A TopLevel window is a frame that displays panes and views.

- A pane can be displayed one time on each TopLevel window and it serves a specific debug purpose. Examples of panes are Hierarchy, Data, Watch, and the Console panes.

Panes can be docked on either side of a TopLevel window or remain floating in an area in the frame not occupied by docked panes (called the workspace).

- A view can have multiple instances per TopLevel window. Examples of views are Source, Wave, List, Memory, and Schematic.

DVE TopLevel window can contain any number of DVE views and panes. You can choose to display data in one or many DVE windows and panes by setting defaults, using the status bar window controls, or docking and undocking windows as you work.

Managing Target Views

You can set target views to create panes either as TopLevel window or in the existing frame. At the bottom right corner of each TopLevel window are target icons ([Figure 2-2](#)). These icons represent pane types.

Figure 2-2 View target icons



Targets a new Source view in a new TopLevel window.

Source



Targets a new Schematic view in a new TopLevel window.

Schematic



Targets a new Path Schematic view in a new TopLevel window.

Path Schematic



Targets a new Wave view in a new TopLevel window.

Wave



Targets a List view in a new TopLevel window.

List



Targets a new Memory view in a new TopLevel window.

Memory

Target icons can have the following two states:

- Targeted – In this state, the target icon has a dart in it, which means a new view will be created in the current frame.

- **Untargeted** – In this state, the target icon has no dart in it, which means a new TopLevel window will be created for the chosen view.

Check marks indicate that targeted windows are attached to the current window.



No check exists in this targeted Wave view icon

To open a pane in a new TopLevel window

1. Click the icon in the status bar to remove the check mark.
2. Click a corresponding window icon in the toolbar to open a window of that type.

It will not be attached to the current window and will open in a new TopLevel window.

Docking and Undocking Views and Panes

You can use the Windows menu to dock and undock windows and panes.

- Select **Windows > Dock in New Row**, then select the row position in which to dock the currently active window.
- Select **Windows > Dock in New Column**, then select the column position in which to dock the currently active window.
- Select **Undock** to detach the currently active window or pane.

To delete a window, click the **X** icon in the corner of the pane. This is the same for all dockable windows.

Dark blue color of dock handle (dock handle is the train track that connects to the X icon) indicates that this docked window is active. This is the same for all dockable windows. An action must occur such as a click to make the window active.

Dragging and Dropping Docked Windows

To drag and drop a docked window, click the dock handle and drag and drop the window to a new dock location or to a non-docked window.

Right-clicking on the dock handle invokes a small pop-up menu:

Undock	Undocks the active window.
Dock	Left – Docks the selected window to the left wall of the TopLevel window. Right – Docks the selected window to the right wall of the TopLevel window. Top – Docks the selected window to the top wall of the TopLevel window. Not recommended. Bottom – Docks the selected window to the bottom wall of the TopLevel window.

Using the Menu Bar and Toolbar

The menu bar and toolbar allow you to perform standard simulation analysis tasks, such as opening and closing a database, moving the waveform to display different simulation times, or viewing HDL source code. For more information about the menu bar and toolbar options, see [Appendix A, "Menu Bar and Toolbar Reference"](#).

Setting the Simulation Time

To set the simulation time display in the waveform

1. Select **View > Go To Time**.

The Go To Time dialog box appears.



2. Enter a value and click **Apply** or **OK**.

The waveform display moves to the specified simulation time.

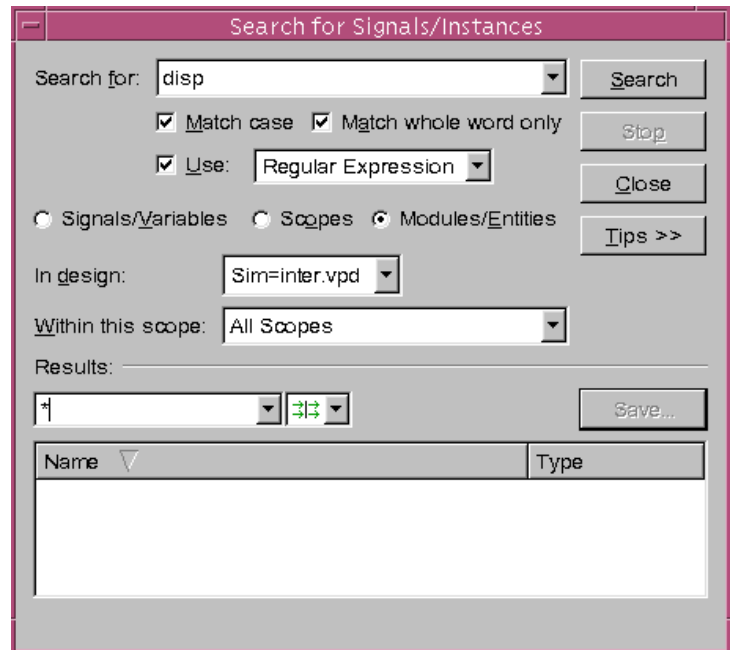
For a complete description about setting the simulation time and using the Waveform window, see [Chapter 5, "Using Wave Views"](#).

Searching Instances of Signal

To search instances, modules, variables, entities, scopes, and signals

1. Click the Search for Signals/Instances icon  on the toolbar.

The Search for Signals/Instances dialog box appears:



2. Enter the search criteria as follows:

Field Name	Description
Search for:	Specifies the signal name that you want to search.
Match case:	Searches specific to the specified text case. For example, if you enter <code>disp</code> as the signal name in the "Search for:" text box, and you select this check box, DVE would search only those signals that match the case <code>disp</code> .
Match whole word only:	Searches the signals containing the whole word. For example, if you enter <code>clk</code> <code>reset</code> in the "Search for:" text box, and select this check box, you will not find signals <code>clk</code> or <code>reset</code> .
Use:	Finds a signal using wildcard or Regular Expression.

Field Name	Description
Signal type:	Identifies signal, instances, scope, modules, or entities.
In design:	Specifies the design in which you want to search.
Within the scope:	Specifies the scope in which you want to search.

3. Click **Search**.

All the signals matching the specified criteria are displayed in the Results text area.

4. Enter the text string to filter items or select the filter type from the pull-down menu.

5. Click **Save**.

The results are saved as a text file.

Using Filters

You can enter the filter strings in three different styles:

- Explicit wildcard (Default)
- Implicit wildcard (*) at the beginning and end of filter string
- Regular expression strings

The logical OR operator "||" is supported in Explicit and Implicit wildcard strings. The Filter drop-down is also available in the Data pane, Wave Viewer, and Signal Search Results Dialog box.

To use the filter

1. Click **Edit > Preferences**.

The Application Preferences dialog box opens.

2. Click the **Global** category.
3. In the **View Filters** group box, select the following options, as appropriate:
 - Apply view filters immediately when typed
 - Case sensitive
 - Syntax - The following options comprise the Syntax drop-down list:
 - Wildcards - Input the wildcard pattern to filter items. The default string is “*”.
 - Simple - Input the regular expression pattern to filter items. The default string is “”.
 - Regular Expressions - Input the simple string to filter items. The default string is “.*”.

Editing Preferences

You can edit preferences to customize the display of DVE views and panes. For more information about the preferences option for all the panes and views in DVE, see the section entitled, [“Editing Preferences” on page A-22](#).

3

Using the Hierarchy and Data Panes

This chapter describes how to use the DVE Hierarchy and Data panes to show the static design structure in a tree view, navigate the design to view results in other DVE windows and panes, and display signal data. It includes the following topics:

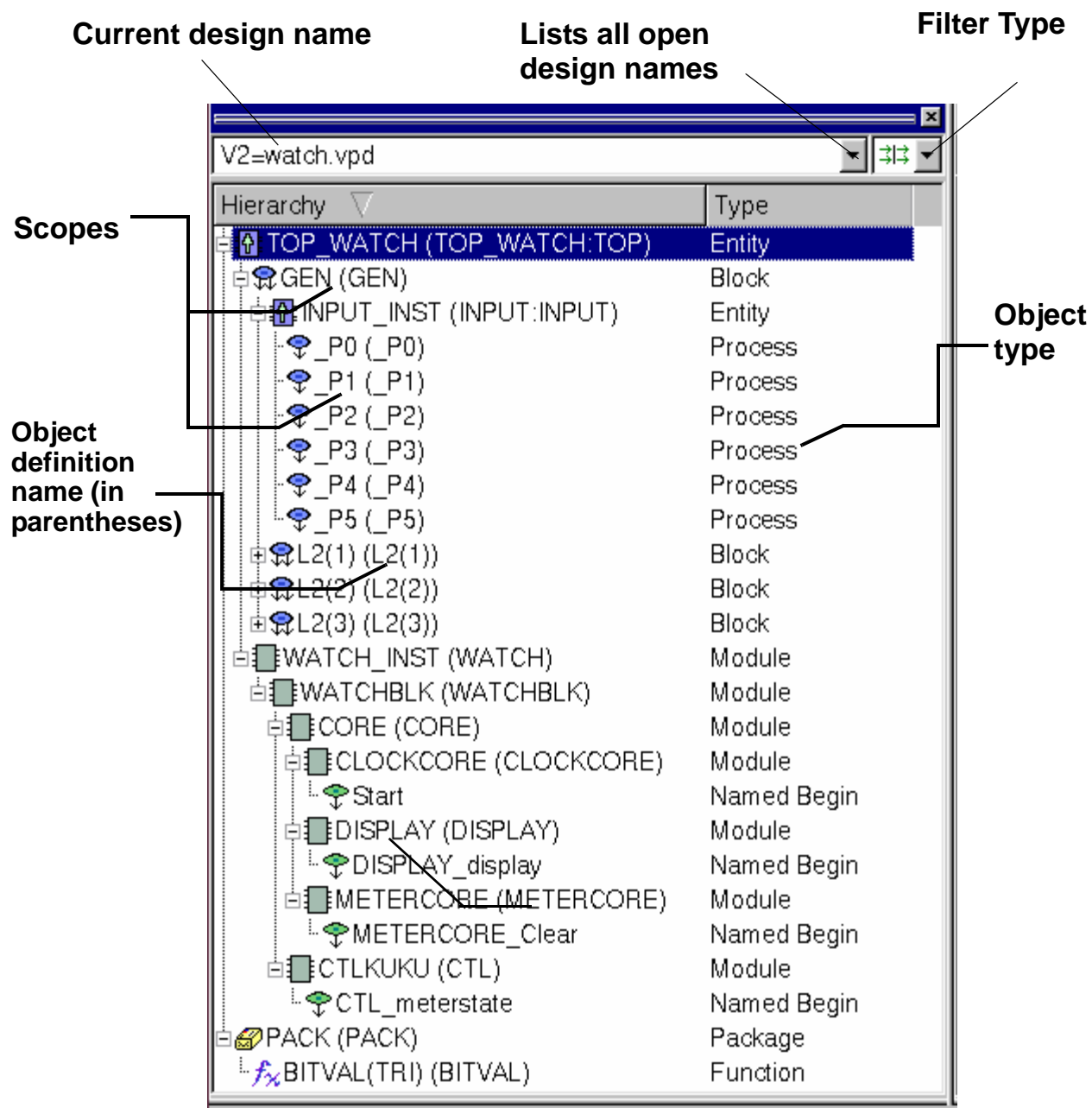
- [“The Hierarchy Pane”](#)
- [“The Data Pane”](#)

The Hierarchy Pane

The Hierarchy pane, shown in [Figure 3-1](#), is composed of two drop-down lists on top of a tree view.













- The drop-down list on the left is the design selection list, and contains a list of currently open designs, with the current design at the top.
- DVE uses the drop-down list on the right for filtering object types.
- The tree view is made up of two columns: Hierarchy and Type.
 - The Hierarchy column shows the static instance tree. The names in the instance tree are in the instance name (definition name) format. Top modules (or scopes) are at the top-level of the tree.
 - The Type column displays the type of hierarchical object.






Figure 3-1 Hierarchy Pane



Scope Types and Icons

DVE displays a wide variety of scope types in the Hierarchy pane. Each scope type is represented by a specific icon. The following table provides an overview of the various scope types and their corresponding icons:

Scope Type	Icon	Description
Design Root		The top of the design hierarchy. Expands the Design Root to see the design.
Verilog Module		Denotes an instance of a Verilog module. Verilog instances are listed as instance name (module name).
Verilog Task		Denotes a Verilog task.
Verilog Function		Denotes a Verilog task.
Verilog Named Begin		Denotes a named begin.
Verilog Named Fork		Denotes a named fork.
Assertion Unit		Denotes an assertion unit. In hierarchy, an assertion unit is listed below the instance to which it is bound. Assertions are listed as instance name (unit name).
VHDL Cycle Instance		Denotes an instance of a VHDL entity that was simulated in cycle mode. VHDL instances are listed as instance name (Entity:Architecture).
VHDL Event Instance		Denotes an instance of a VHDL Entity that was simulated in event mode. VHDL instances are listed as instance name (Entity:Architecture).
VHDL Package		Denotes a VHDL package.
VHDL Procedure		Denotes a VHDL procedure.
VHDL Function		Denotes a VHDL function.

Scope Type	Icon	Description
VHDL Process		Denotes a VHDL process.
VHDL Block		Denotes a VHDL block.
VHDL Generic		Denotes a VHDL generic.
SystemC Instance		Denotes an instance of a SystemC entity.
SystemC Process		Denotes a SystemC process.

Navigating Open Designs

In DVE, more than one design can be open, but only one of them can be active at any point of time. This active design is the "current design".

Designs are given designator strings, so that in cases where objects from more than one design are allowed (for example, in the Wave view), it is possible to relate object names to their designs. By default, the designators are V1, V2, V3, and so on.

For example, if a design A contains an object called `top.a` and design B also contains an object called `top.a`, these objects would be shown as `V1:top.a` and `V2:top.a`, by default. You can also choose your own design designators from the Open Database dialog box.

Filtering the Data in the Hierarchy Pane

You can view object types such as Tasks, Functions, Named Blocks, Packages, Processes, and Unnamed Processes in the Hierarchy pane.

To filter the data based on object types, click the Type filter list and select and clear the desired object types.

Expanding and Collapsing the Scope

If the scope has subscopes, a plus sign (+) appears to the left of the scope.

To expand and collapse the scope

1. Click the plus sign (+).

All the subscopes are displayed and the plus sign, +, sign turns to a minus sign, (-).

2. Click the minus sign (-) .

The expanded child scopes collapse. Note that all levels of expanded scopes beneath the scope that was clicked will collapse.

Rearranging Columns in the Hierarchy Pane

You can sort the Hierarchy column or rearrange the order in which the column headings appear in the Hierarchy pane.

To rearrange and sort the columns in the Hierarchy pane

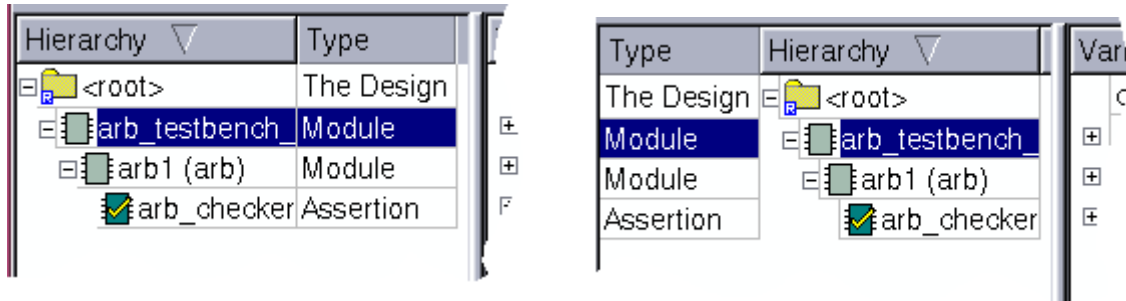
1. Click and hold the left mouse button on the column heading.
2. Drag the column to a new location and release the mouse button.

The column moves to a new location in the Hierarchy pane.

3. Click the arrow in the column heading.

The scopes are sorted in alphabetical order.

Figure 3-2 Moving a Column Heading



Populating Other Views and Panes

Use the Hierarchy pane to view data in other DVE windows and panes.

Displaying Variables in the Data Pane

To display variables in the Data pane, select an object in the Hierarchy pane.

Dragging and Dropping Scopes

You can drag and drop a selected object into any other DVE pane or window (such as the Source view, the Wave view, and the List view).

The following points should be noted while dragging the objects in various panes:

- Dropping a scope into the Source view displays the definition of that object in the Source view and selects the definition line.

- Dropping a scope into the Data pane causes the scope to be selected in the Hierarchy pane and displays the scope variables in the Data pane.

If the scope is already selected in the Hierarchy pane, dropping it into the Data pane has no effect because the variables of the scope selected in the Hierarchy pane are automatically displayed in the Data pane.

- Dropping a scope into the Wave view adds all the scopes signals to a new group or puts them under the insertion bar of the wave signal list.
- Dropping a scope into the Schematic view displays the design schematic for that scope.
- Dropping a scope into the Path Schematic view has no useful results.
- Dropping a scope into the Memory view is not allowed.
- Dropping a scope from elsewhere into the Hierarchy pane selects that scope, if it is available.
- If you drag a scope into the Hierarchy pane, but the object has not yet been loaded into the Hierarchy pane, use the **Edit > Search For Signal/Scopes** menu option.
- Dropping a scope into a text area, such as the DVE command line, drops the full hierarchical text. However, one exception is to drop a scope in the Find Dialog text entry area (either dialog or toolbar area). In this special case, just the leaf string is dropped. For example, dropping "top.c.b.a" results in just "a" in the Find text area.

If you select more than one hierarchy object (you can do this by pressing the Control key and clicking the left mouse button), the object closest to the linear top of the list is dropped. For example:

```
top
  top.a
    top.a.b
  top.b
    top.b.b
```

In this example, if you select, drag, and drop both `top.a.b` and `top.b` into a text area, DVE drops only `top.a.b`.

- Double-clicking on a Hierarchy object causes the object's definition to be displayed and highlighted in the Source view (based on the reuse policy of the Source view).

Using Context-Sensitive Menu

In any DVE window, right-click to display the context-sensitive menu (CSM), and then select a command. The Hierarchy pane CSM is as follows:

Command	Description
Copy	Copies the selected text.
Show Source	Displays the source code in the Source view for the selected scope; same as double-clicking.
Show Schematic	Displays the design schematic of the selected object in a new window.
Show Path Schematic	Displays the path schematic of the selected object in a new window. Note: This only displays the scopes ports.
Add to Waves	Adds all the scope's variables to the Wave view (opens one if none is currently opened).
Add to Lists	Adds all the scope's variables to the List view (opens one if none is currently opened.)

Command	Description
Expand By Levels	Allows expansion of multiple levels with a single action.
Expand All	Expands the entire hierarchy at once. There may be a delay getting the hierarchy from the simulation when working interactively.
Collapse Parent	Collapses the selected scope.
Collapse All	Collapses all expanded scopes.
Select by Levels	Allows you to select more than one level at a time.
Select All	Selects all levels that are visible (does not implicitly expand).
Add Dump	Opens the Dump Values dialog box for specifying dump parameters.
Dump	Appends signals and scopes or objects to the current dump file.

Dumping Signal Values

To dump signal values

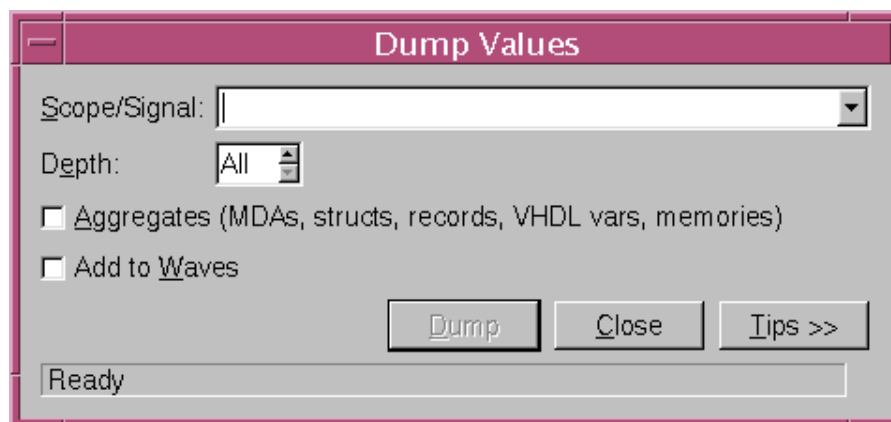
1. Select the scope in the Hierarchy pane, right-click and select **Add Dump**.

or

Select **Simulator > Add Dump**.

The Dump Values dialog box appears.

Figure 3-3 Dump Values Dialog Box

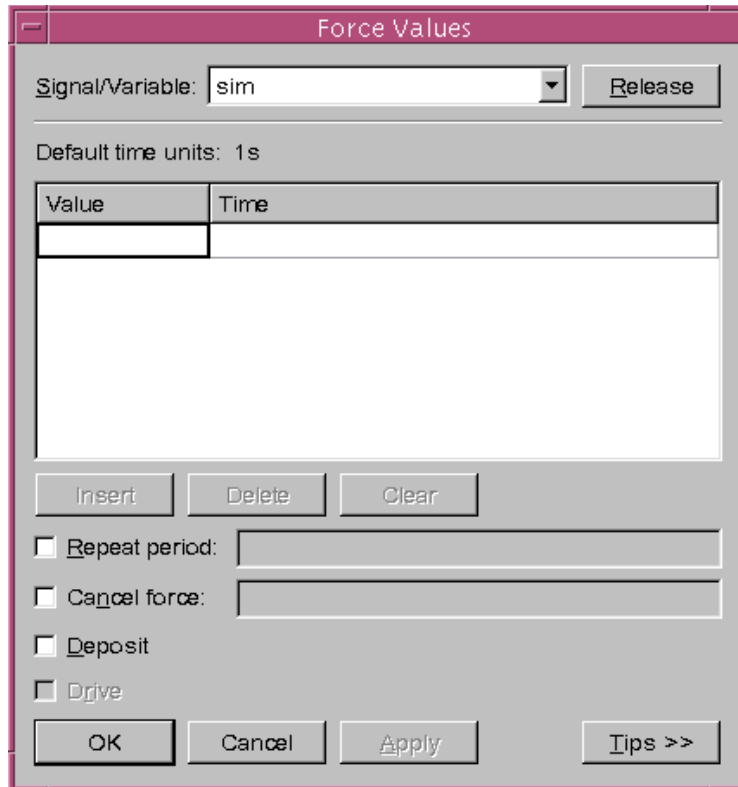


2. Select the Scope/Signal that is to be dumped.
3. Select the **Depth** to specify the level in the hierarchy for which the objects are to be dumped.
4. Select **Aggregates** to dump all complex data types.
5. Select **Add to Waves** to add the scopes and signals to the Wave view.
6. Click **Dump** to save the specified objects.

Forcing Signal Values

You can force a value onto a signal or variable by specifying force criteria in the Force Values dialog box.

Figure 3-4 Force Values Dialog Box



The dialog box is titled "Force Values". It features a "Signal/Variable:" dropdown menu with "sim" selected and a "Release" button. Below this, it states "Default time units: 1s". A table with two columns, "Value" and "Time", is present, with the first row highlighted. Under the table are buttons for "Insert", "Delete", and "Clear". Further down are four checkboxes: "Repeat period:", "Cancel force:", "Deposit", and "Drive", each followed by an input field. At the bottom are "OK", "Cancel", "Apply", and "Tips >>" buttons.

Value	Time

To force signal values

1. Select **Simulator > Add Force**.

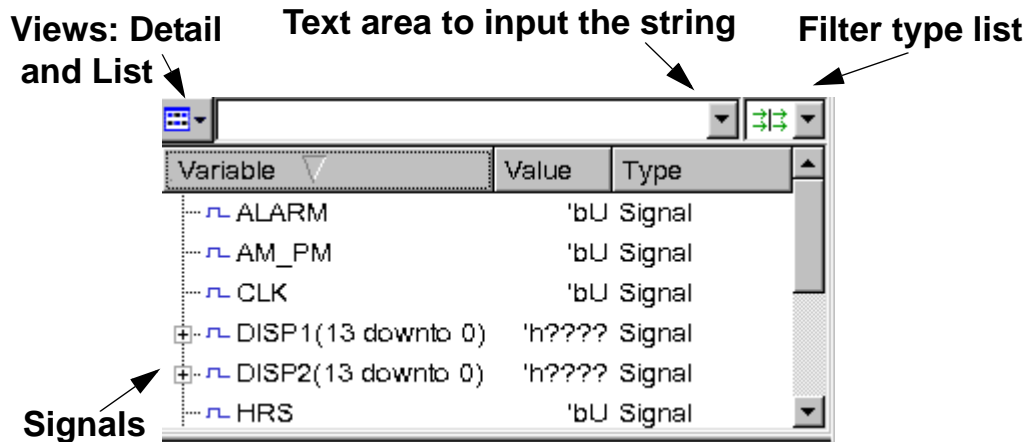
The Add Force dialog box opens.

2. In the Signal/Variable field, specify the signal/variable whose value is to be forced.
3. In the Table, specify a timed sequence of values by a set of value/time pairs.
 - The unit of time in time expressions is optional and defaults to the time precision of the tool.

- The @ sign is optional and indicates that the following time expression is relative to the beginning of the simulation. If it is omitted, the time expression is considered as relative to the current simulation time.
4. Click **Add** or **Delete** to add or delete the rows in the table.
 5. Use **Repeat period** to specify a repeat delay, after which the sequence of forced values need to be restarted.
 6. Use the **Cancel force** section to specify a cancellation time for the force command.
 7. Click one of the following:
 - OK** to apply your force specification and close the dialog box.
 - Apply** to apply your force specification and have the dialog box remain open.
 - Cancel** to close and not apply the specification.


The Data Pane

DVE displays simulation analysis data corresponding to the contents of the scope you select in the Hierarchy pane.



Viewing Signals and Values

To view signals and their values

1. Click the arrow  next to the object in the Hierarchy pane.

The values at the current simulation time of the selected scope are displayed in the Data pane.

2. Click the arrow in the **Variable** column.

The signals get sorted in alphabetical order.

3. Click the down arrow to display the type filtering pull-down menu.

4. Select or clear the check boxes against each filter type.

The signals are filtered per your selection.

5. Select a signal in the Data pane, then select **Source > Show Source**.

The source code of the selected signal is displayed in the Source view.

Viewing Interfaces as Ports

You can view Interface/Modport in the Data pane when it is passed as port. You need to select the module name in the Hierarchy pane to view the port in the Data pane. You can add the interface/modport port to the Wave view, List View, or Watch view.

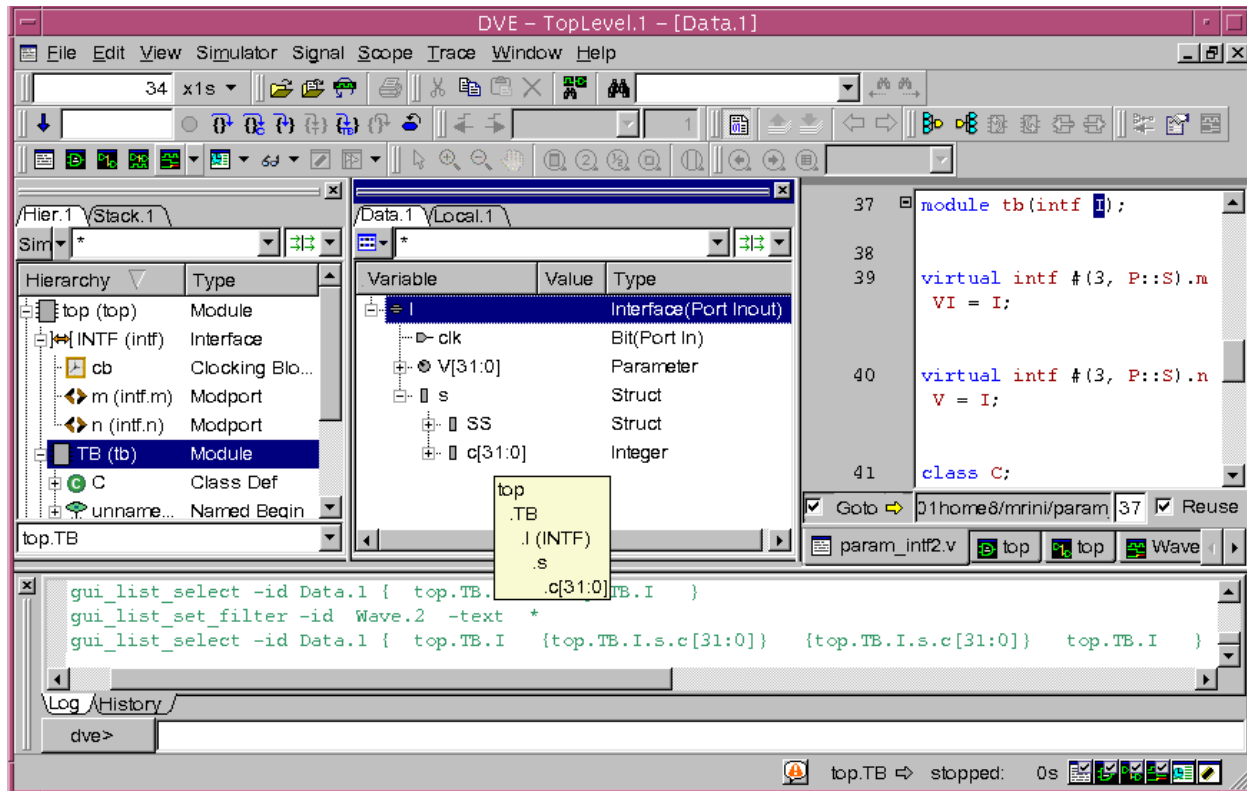
To view the interface port in Data pane

1. Load the database in DVE.

The module is displayed in the Hierarchy pane.

2. Select the module.

The interface/modport port and its type is displayed in the Data pane. The tooltip shows the interface/modport used.



3. Click the “+” button under the Variable column in the Data pane to expand the interface/modport port.

The signals under the interface/modport port are displayed.

4. Right-click the interface/modport port in the Data pane and select **Show Source**.

The source of interface/modport is shown in the Source viewer. You can also drag and drop the interface/modport from the Data pane to the Source viewer.

5. Use the Text filter or Type filter drop-down and select the Interface/Modport port filter to filter the signals.

6. Select the interface/modport port in the Data pane and select **Signal > Show Definition** from the menu or right-click the signal and select **Show Definition**.

The definition is shown in the Hierarchy pane, signals of interface/modport port in the Data pane, and the definition location is shown in the Source view. You can also drag and drop the interface/modport port from the Data pane to the Wave view.

Note:

- Interface array port is not displayed in the Data pane.
- Driver/Load, schematic/path schematic operations do not support interface port and signals of interface port. A warning message is displayed when you perform these operations.
- Follow signal doesn't work for interface port and signals of interface port.
- Modport clocking port is not shown in the Data pane.

Viewing \$unit Signals

\$unit is the name of the scope that encompasses a compilation unit. Its purpose is to allow the unambiguous reference to declarations in the compilation unit scope. This is done via the same class scope resolution operator used to access package items. For more information about compilation units, see the chapter “Hierarchy” in the *IEEE P1800-2005 SystemVerilog LRM*.

You can view the \$unit signals in the Data pane.

To view the \$unit signals

1. Load the design in DVE.

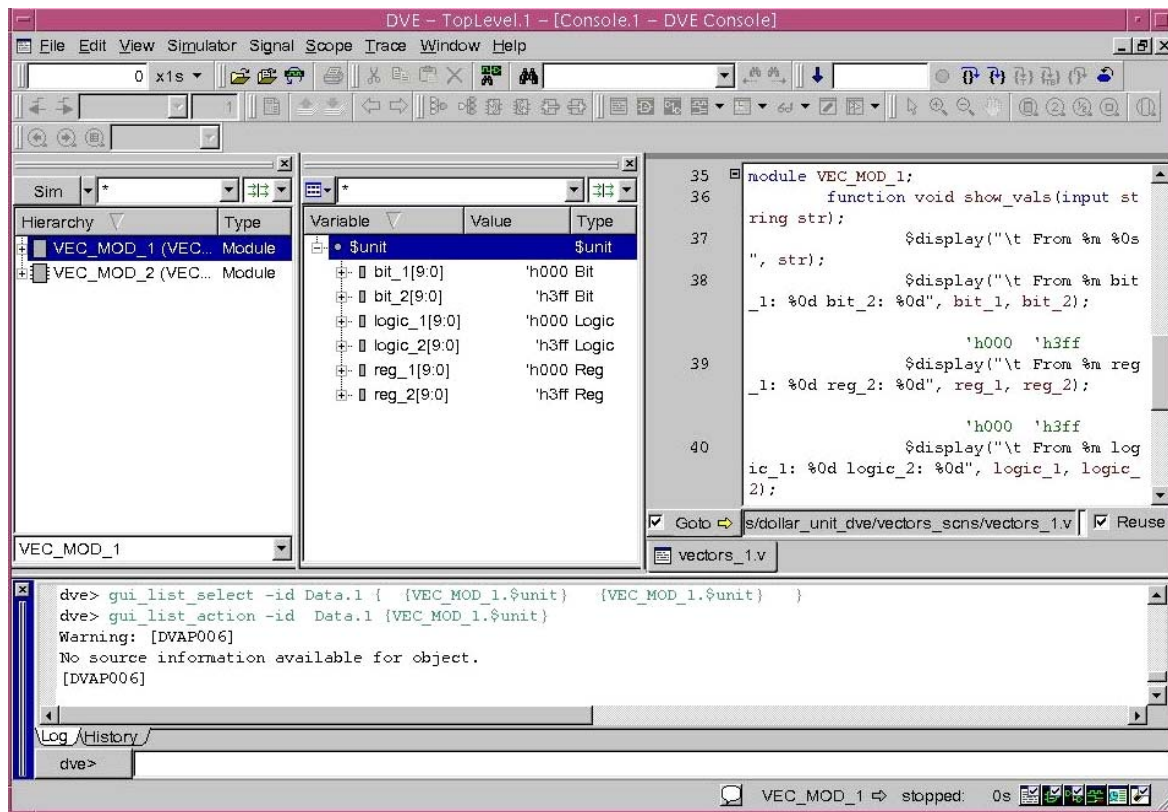
The design appears in the Hierarchy pane.

2. Select a scope in the Hierarchy pane.

The \$unit folder is visible in the Data pane with the Type field \$unit.

3. Expand the \$unit folder.

The signals under the \$unit folder are visible.



4. Filter or sort the signals under the \$unit folder, if required.
5. Add and view the \$unit signals in Waveform, Watch, and List window, as required.

Note:

- \$unit is not visible in the Hierarchy pane because it is not a global scope.
- The task, function, class definition defined in \$unit are not visible in the hierarchy pane or data pane. You can view the task, function, or class if you put them in a global package.
- Drivers or loads, schematic, path schematic, or back trace schematic are not supported for \$unit signals.

The Watch Pane

You can use the Watch pane to monitor the status of a specific signal, a group of signals, or an object regardless of the active thread. You can drag and drop the object or signal from the Hierarchy and Data panes into the Watch pane to view its behavior.

The Watch pane displays the selected item, its value, type, and the scope in which it belongs.

The Watch pane, by default, contains three tabs labeled Watch 1 through Watch 3. There is no limit to the number of tabs you can add. Using the check box in the scope column, you can tie the variable to a given thread throughout simulation or tie the variable to the currently selected thread.

To open the Watch pane

1. Select an object or signal from the Hierarchy or Data pane.
2. Right-click and select **Add to watches**.

The Watch pane is displayed with the selected signals.

3. To add a Watch tab, go to the menu **View > Watch > Add New Page**. You can also delete the watch tabs.

4

Using Source Views

The Source view displays the HDL, any foreign language (C, C++, SystemC or OV) or assertion source code of your design. You can open as many Source views as you need to perform your analysis by selecting **Window > New > View > Source View**. You can also set the number of Source views that DVE should display in the TopLevel window.

This chapter includes the following topics:

- [“Loading Source Code”](#)
- [“Using the Mouse in the Source View”](#)
- [“Working with the Source Code”](#)
- [“Navigating the Design from the Source View”](#)
- [“Navigating Code in Interactive Simulation”](#)

- [“Setting Breakpoints in Interactive Simulation”](#)
- [“Annotating Values”](#)

Loading Source Code

This section includes the following topics:

- [“Loading a Source View from the Hierarchy Pane”](#)
- [“Loading a Source view from the Assertion View”](#)
- [“Displaying Source Code from a File”](#)

Loading a Source View from the Hierarchy Pane

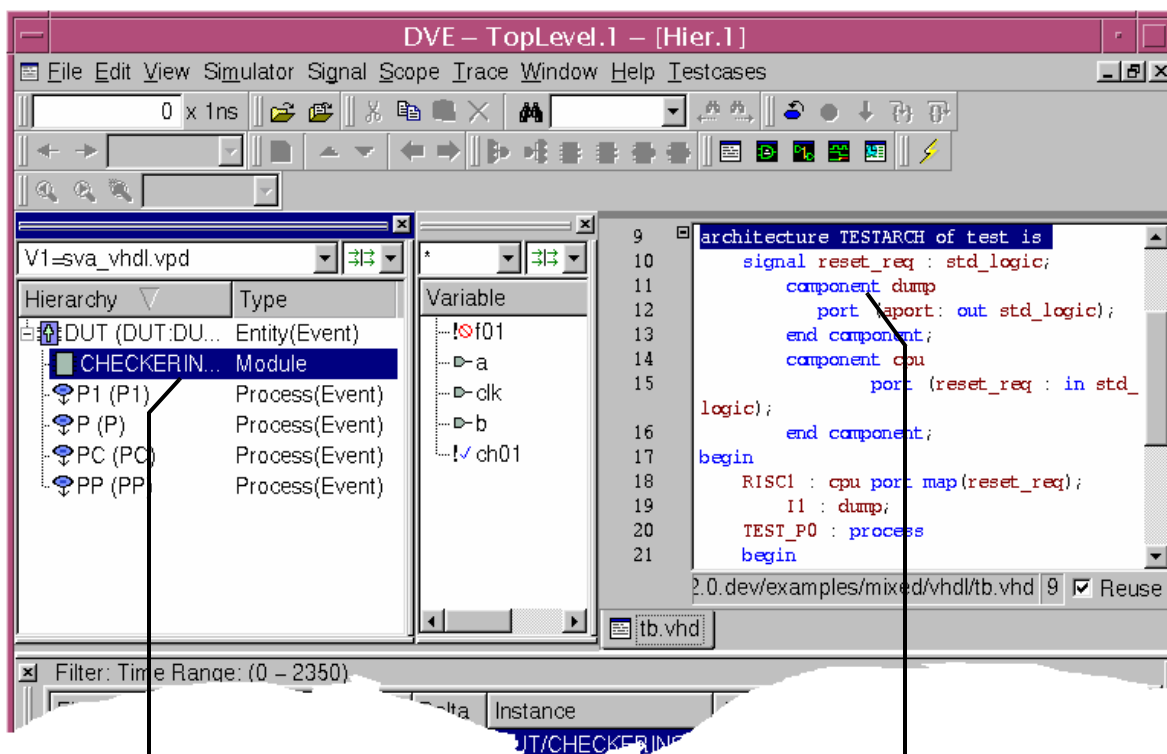
Ensure that a database is currently loaded in the Hierarchy pane.

To load the Source view from the Hierarchy pane

1. In the Hierarchy pane, perform one of the following:
 - Select a scope, then select **Scope > Show Source**.
 - Select a scope, right-click and then select **Show Source**.
 - Double-click on a scope icon.
 - Drag and drop scope from the Hierarchy pane to the Source view.

The Source view loads the data corresponding to the selected scope.

Figure 4-1 Loading the Source View



Hierarchy pane

Corresponding data in the
Source view

Loading a Source view from the Assertion View

Ensure that the Assertion view is loaded with data.

If your design contains assertions, the Assertion view loads results when you open the simulation database.

To load assertion code into a Source view via the Assertion view, perform one of the following:

- Select an assertion in either tab, then select **Scope > Show Source**.
- In the Assertion Failure Summary tab or the Assertions tab, double-click the variable or assertion you want to display in the Source view.
- In the Assertion Failure Summary tab or the Assertion tab, drag and drop the item to the Source view.
- Select **File > Open File**, then select an assertion file.

DVE loads and displays the source file.

Displaying Source Code from a File

You can open a source file in the existing Source view, or you can open a new window by performing the following steps:

1. Select **File > Open File**.

The Open Source File dialog box appears.

2. Select the name of the HDL file you want to display from the browser, and then click **Open**.

DVE loads and displays the selected HDL source file.

Using the Mouse in the Source View

The following table describes the different mouse actions in the Source view:



Mouse Action	Command Operations
Left-click	Clears the current selection and selects a signal or an instance.
Drag-left	Selects area for multiple selection.
Click on the line number	Selects the whole line.
Double-click on a signal name	Traces the signal's drivers.
Double-click on an instance	Pushes down into the instance's definition module.
Double-click on a module name	Displays the upper hierarchy and locates the module's instantiation.
Double-click on an architecture	Jumps to the entity definition of selected <code>entity_name</code> or jumps to an instance definition of the entity.
Double-click on an entity (after double-clicking on an architecture)	Jumps to the architecture that was previously double-clicked.
Right-click on a signal name or anywhere in the Source view	Displays a CSM or Source view menu.
Position the mouse cursor on any signal name.	Displays ToolTip with the current value.

Working with the Source Code

This section describes how to use the Source view to examine the source code while debugging it. It allows you to expand and collapse required portions of the code, display line attributes for specific lines, and edit the source code using a text editor.

Expanding and Collapsing Source Code View

To expand or collapse the source code view

1. Click  in the Line Attribute area, or right-click and select **Expand Source** to view code that is folded.
2. Click  in the Line Attribute area, or right-click and select **Collapse Source** to hide code.

Editing Source Code

To edit source code

1. Select the text editor by setting the `$EDITOR` environment variable.

```
%>setenv EDITOR vi [OR]
```
2. Select **Edit > Preferences**, select **Source view**, choose the editor you prefer and save from the editor pull-down menu.
3. In the source code area, right-click and select **Edit Source** or **Edit Parent** to open the source code in the default editor and edit the same.

Selecting and Copying Text to the Clipboard

You can select some or all text displayed in a Source view, and copy it to your clipboard.

To select all text or copy text in a Source view

1. Drag your mouse across the text to select a portion of text in a Source view.

DVE highlights the selected text.

2. Select the text in the Source view.
3. Right-click and select **Copy** from the CSM.

You can paste this text in any text area or in the source editor.

Navigating the Design from the Source View

Use the Source view to navigate through the design and view results in other DVE windows by dragging and dropping signals, scopes, and objects into other views.

To view the source code in other views

1. Select the code in the Source view.

The text is highlighted.

2. Right-click and select the desired view from the CSM.

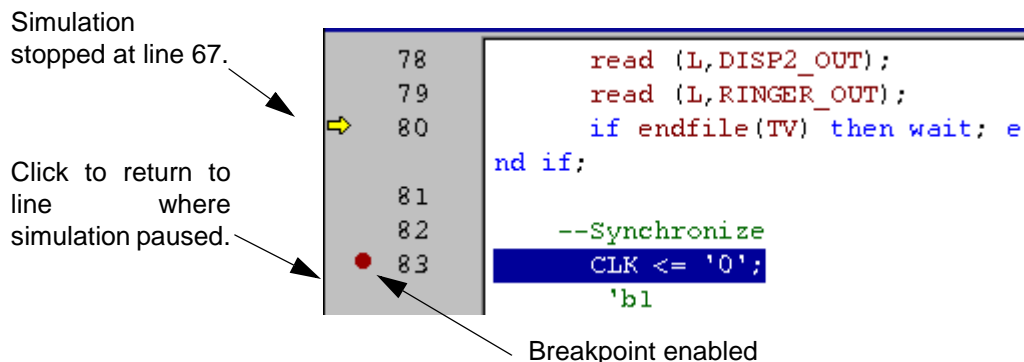
You can view the source code in the Wave view, List view, Group them, or add them to the Watch view.

Navigating Code in Interactive Simulation

Use the line attribute area to toggle line numbering and control line breakpoints when running interactive simulation. To display line attributes, right-click in the line attribute area, then select **Line Number** to toggle line numbering.

When you run a simulation interactively, the line where the simulation stopped is marked by a yellow arrow in the Source view. However, you can search and review any code in the design during a pause in the simulation. You can return to the line where the simulation paused by clicking the yellow arrow at the bottom of the Source view as shown in the following illustration.

Figure 4-2 Navigating an interactive simulation



Setting Breakpoints in Interactive Simulation

You can set breakpoints to stop the simulation. Note the following points regarding breakpoints:

- Line breakpoints execute each time a specified line is reached during simulation (see the section *Displaying Line Attributes and Managing Breakpoints* from the dialog box for more information) about line breakpoints. You can also specify an instance to have the tool stop only at the line in the specified instance.
- Time breakpoints stop at a specified absolute or relative time in the simulation.
- Signal breakpoints trigger when a specified signal rises, falls, or changes.
- Assertion breakpoints stop at a specified assertion event.
- Task/Function breakpoints stop at the specified task or function.
- Dynamic breakpoints are not supported.

To set and delete a breakpoint

1. Click in the attributes area of the Source view next to an executable line.

A solid red circle indicates that a line breakpoint is set.

Note:

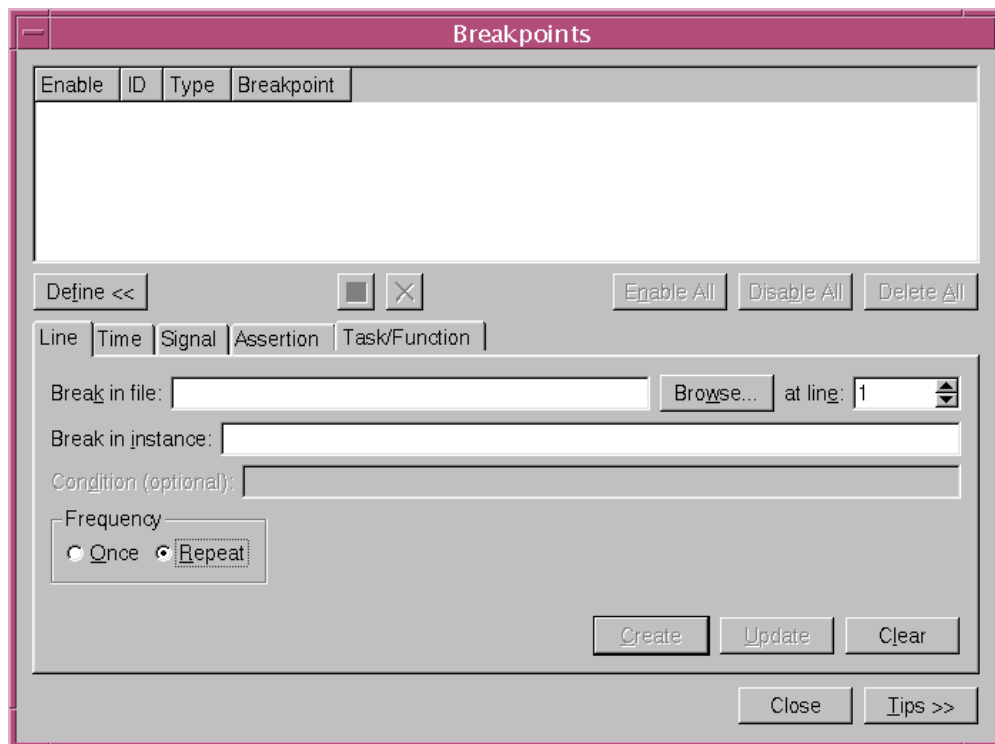
A line breakpoint can only be set on an executable line. If a line is not executable, no breakpoint will be set when you right-click next to it.

2. Right-click in the attributes area of the Source view, then select **Set Breakpoint**.

A plus sign (+) appears inside the solid red circle.

3. Right-click on the red circle and select **Properties**.

The Breakpoints dialog box appears.



4. Enter the condition in the Condition field.

5. Click **Update**.

The breakpoint is set to stop simulation on the condition.





6. Select the solid red breakpoint circle to disable it.

The solid red circle changes to an empty red circle.

7. Right-click on an enabled or disabled breakpoint, then select **Delete Breakpoint** or **Delete All Breakpoints**.

The red circle disappears indicating that the breakpoint is deleted.

The following table describes the breakpoint icons and ways to control them:

Breakpoint Icon	Description
	Denotes a breakpoint.
	Denotes a line breakpoint was set on this line, and it is enabled. Line stepping and simulators will stop on execution of this line.
	Denotes a disabled breakpoint.
	Denotes a line breakpoint was set on this line, and it is disabled. Line stepping and simulators will not stop on execution of this line.

Managing Breakpoints

You can manage all types of breakpoints in an interactive simulation from the Breakpoints dialog box.

To create breakpoints using the Breakpoints dialog box

1. Right-click in the Source view line attribute area and select **Set Breakpoints**.

The Breakpoints dialog box appears.

2. Click **Define** to display the breakpoint creation tabs.
3. Select the Line tab and enter the following information:
 - Break in file - enter the file name or browse to the file where you want to create the breakpoint.

- At line - enter the line number for the breakpoint.
 - Break in instance - enter the instance where the breakpoint will fire.
4. Select the **Time** tab and enter the following information:
 - Select **Absolute** or **Relative** time reference, then enter the time to set the breakpoint.
 5. Select the **Signal** tab and enter the following information:
 - Enter the desired signal in the Break on signal text.
 - Select Any, Rising, or Falling Edge to define the breakpoint event.
 6. Select the **Assertion** tab and enter the following information:
 - Enter the full path to the Assert in the Break on Assertion text field.
 - Select an event type to trigger the breakpoint from any, start, end, failure, or success.
 7. Select the **Task/Function** tab and enter the following information:
 - Enter the full path to the task or function in the Break in Task/Function field.
 8. (Optional) Enter a condition for VHDL objects to be met for the breakpoint to fire.

Note:

Condition is not supported for Verilog objects.


9. Select the frequency. Select **Once** if you want to fire the breakpoint once, otherwise, select **Repeat**.

10. Provide a name for the breakpoint in the Name field.
11. Define Tcl commands to execute when breakpoint triggers in the Command field.
12. Enter the skip time before stopping in the Skip field. Select the Continue check box to prevent breakpoint to stop. Selecting the Quiet check box will not print any error message when breakpoint triggers.
13. Click **Create**.

The breakpoint is created and appears in the breakpoint list box.

Annotating Values

To enable value annotation for variables/signals in the Source view

1. Click the Annotate Values Icon  in the toolbar.
2. Select **Scope > Annotate Values**.
3. In the Source view, right-click and select **Annotate Values**.

The annotated values display in the Source view.

```
9
10 module DISPLAY (hour_clock, min_clock, sec_clock, hour_
meter, min_meter,
0 5'h00 6'h2c 6'h01 5'h00
6'h00
11 sec_meter, rst, clk, hour, min, sec, mo
de, lcd);
6'h00 St0 *t0 *h00 *2c *01 3'
h2 St1
12
13 // input ports
14 input [4:0] hour_clock;
5'h00
15 wire [4:0] hour_clock;
5'h00
16 input [5:0] min_clock;
6'h2c
17 wire [5:0] min_clock;
6'h2c
18 input [5:0] sec_clock;
6'h01
```

Goto ➡ ./watch.v 10 ☒ Reuse

If there is not enough space to show the values, the value is shown as * (asterisk character). You can see the exact value when you hover your mouse on the variable.

5

Using Wave Views

The Wave view displays waveforms for signals, traced assertions, and signal comparison.

This chapter includes the following topics:

- [“Viewing Waveform Information”](#)
- [“Using the Signal Pane”](#)
- [“Using User-defined Radices”](#)
- [“Using the Waveform Pane”](#)
- [“Printing Waveforms”](#)

For information about using the Wave view to view and debug assertions, see [Chapter 8, “Working with Assertions and Cover Properties”](#).

Viewing Waveform Information

To view waveform information in the Wave view, set the target window and choose the waveform you want to view. You can customize how DVE displays the waveform by changing the settings in the Wave view.

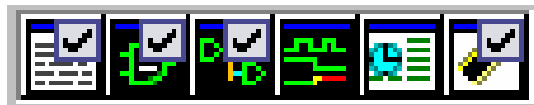
Set the Target View

The default setting is to open a waveform in a new TopLevel window. You can accept the default or use the target window controls in the status bar to display a waveform in the current TopLevel window.

The target windows control the creation of a new TopLevel window:

- Untargeted (default) – when the Waves icon has no check mark in it, a new TopLevel window is created.

Figure 5-1 Target Window Controls




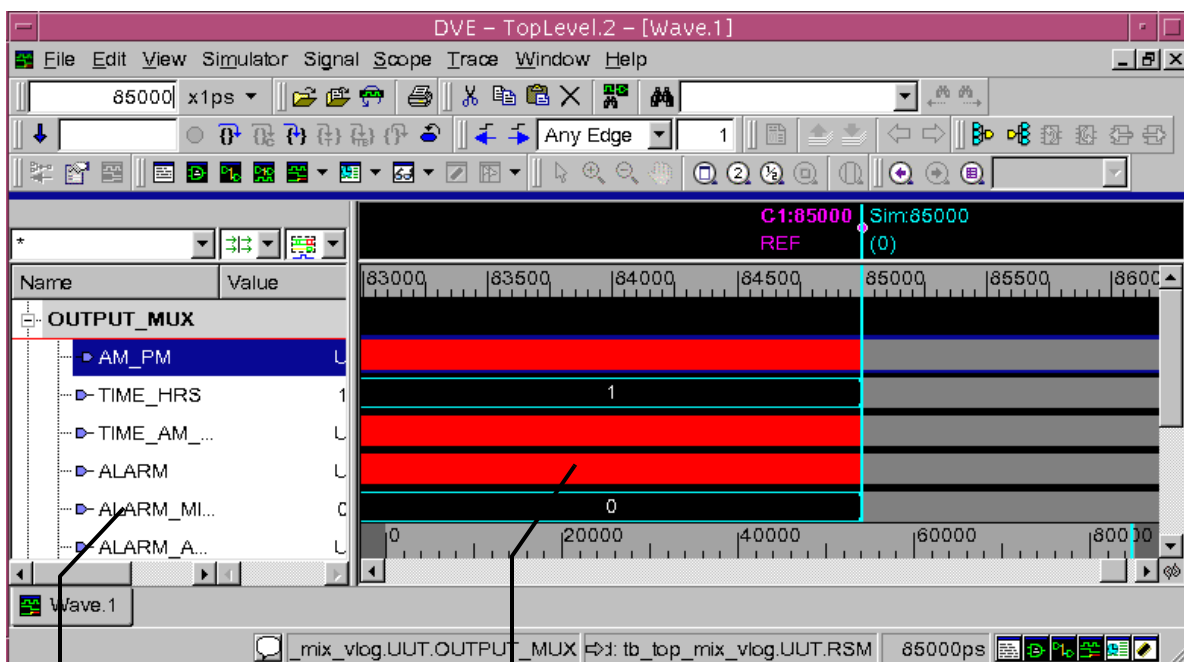
- Targeted – When the icon has a check mark in it, a pane is created in the current TopLevel window frame.

If you open a waveform in the current TopLevel window, you can change the display to a TopLevel window by selecting **Windows > Undock** from the menu bar.

Viewing a Waveform

To view waveform information for signals in the Wave view

1. Select a scope or object from the Hierarchy pane, Data pane, Source view, List view, Schematic view, or Assertion window.
2. Click the **Add to Waves** icon in the toolbar .



Signal Pane

Waveform Pane

3. Add the selected signals to the new waveform window or to the recently used waveform window.

Simply clicking on this icon will add signals to the recently used Wave view.

DVE supports display of Nanosim signals dumped to a VPD file. The Wave view displays the signals with units and resolution appended to the values. [Figure 5-2](#) shows the Wave view display of Nanosim data.

Data type

Analog unit

Vertical scale

Using Wave Views

Using the Signal Pane

The Signal pane displays signals in groups:

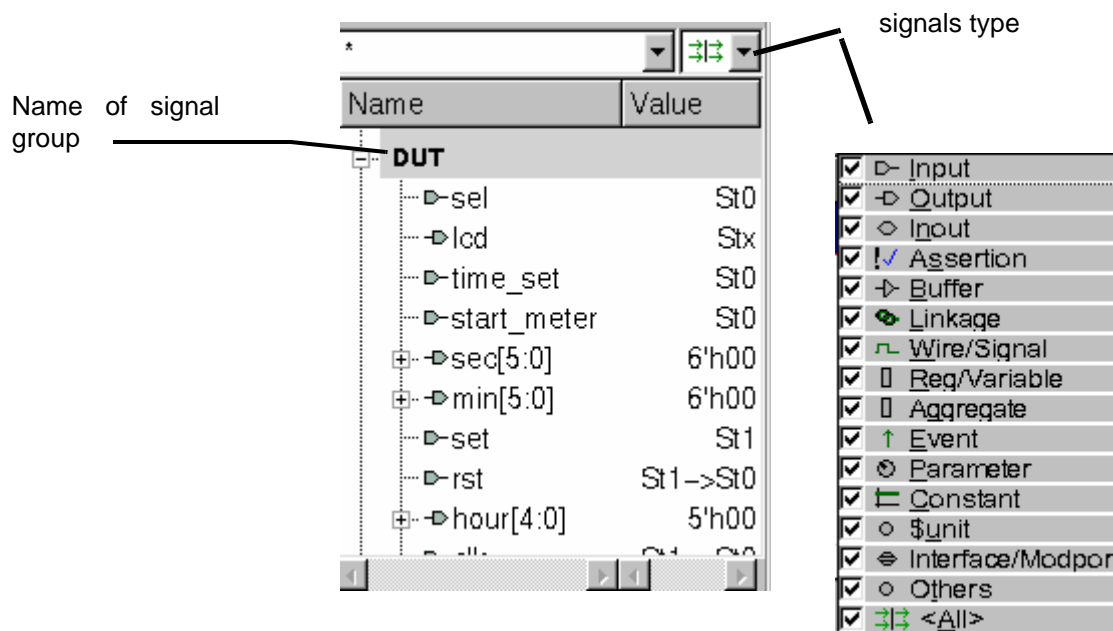
- Scalar signals have their value displayed in binary radix.
- Vector signals have their values displayed in hexadecimal radix.
- Integers, real numbers, and times are displayed in the floating point radix.

The components of the Signal pane are as follows:


- The header allows filtering of signals displayed in the Wave view.
- The **Name** column displays signal names.
- The **Value** column displays the value of signals at the simulation time selected by the C1 cursor (which is also the value in the TopLevel Window Time field).

See [Figure 5-3](#) for an example of the Signal pane.

Figure 5-3 The Signal Pane



Expanding Verilog Vectors, Integers, Time, and Real Numbers

To expand the Vector signals to their individual bits, click the plus icon  to the left of the signal name.

After you expand the display, each bit is added to the Signal pane and waveforms for these bits are added to the Wave view.

DVE represents integers in 32 bits, so you can expand an integer in the Signal pane to display separate waveforms for each of these bits. Similarly, DVE represents the time data type with 64 bits, and you can expand a time to display a waveform for each of these 64 bits.

You cannot expand a real data type.

You can also expand assertions. Upon expanding an assertion, its children will include the assertion clock and the signals and events (or sequences and properties for SVA) that make up the assertion.

Renaming Signal Groups

To rename a signal group

1. Click the signal in the Signal pane.

The signal group is selected.

2. Click the signal name again and enter the new name.

The signal is renamed.

Creating Multiple Groups when Adding Multiple Scopes

When you add the scopes to the waves, lists, or groups from the Hierarchy pane, the signal groups will be created based on their respective scopes.

If you select **Display signal group exclusively** in the Application Preference dialog box, and add multiple scopes to Wave view, multiple scopes will be created, but only the last group will be displayed.

Filtering Signals

To filter signals

1. Click the drop-down menu in the upper right of the Signal pane.

The signal types are displayed.

2. Select or clear a signal, as desired.

The signals are filtered based on your selection.

Adding Signal Dividers

A divider, inserted into a Signal Group, displays in every instance of that signal group when opened in Wave views. Dividers are saved in the session TCL file and are restored when the session is opened.

To separate signals in the Wave view, click **Signal > Insert Divider**. Dividers are added between signals.

There is no limit to number of dividers you can add between signals.

Customizing Duplicate Signal Display

When displaying duplicate signals, you can customize the display of an instance of a signal without affecting the display of any duplicates.

To customize the signal display

1. Select a signal, then toggle **Signal > Default Properties** off.
2. Select **Signal > Properties** and make any changes to the signal scheme or color.

Changes are made to the selected signal without affecting the display of duplicate signals.

Note:

- If you do not toggle Default Properties off, the changes will become the default and duplicate signal display will also change.
- If a signal group is in two Wave views, changing a signal will change the signal in the other Wave view if it is the same instance.

Using User-defined Radices

This section describes how to create, edit, import, and export user-defined radices.

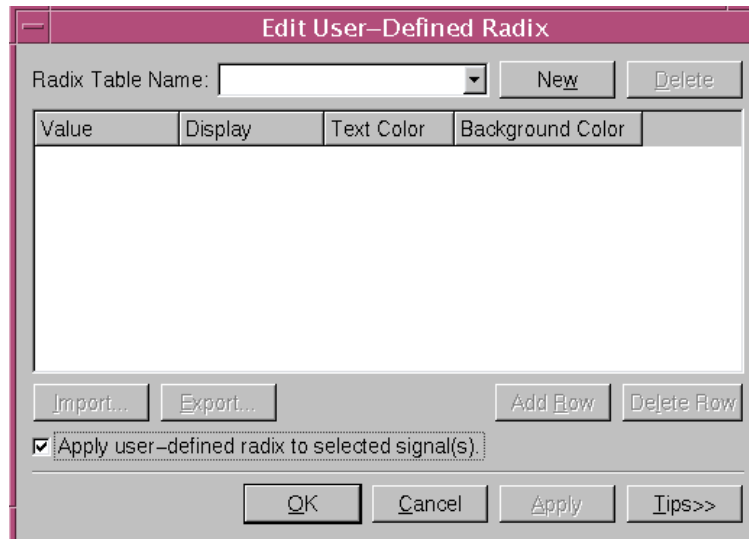
Managing User-defined Radix

You can define a custom mnemonic mapping from values to strings for display in the Wave view.

To create, delete, import, and export a user-defined radix

1. Select **Signal > Set Radix > User Defined > Edit**.

The Edit User-defined Radix dialog box opens.



2. Click **New**, enter a radix name, then hit the **Enter** key on your keyboard.

All buttons on the Edit User-defined Radix get enabled.

3. Click **Add Row** to activate a row for the user-defined radix and perform the following steps:
 - Select the text and background colors for each row entry.
 - Select the radix, click a cell in the Value and Display column, then enter the values.

The radix is edited.

4. Select a row, then click **Delete Row**.

The row is deleted.

5. Select a radix from the Radix Table Name drop-down and click the **Delete** button.

The radix is deleted.

6. Click **Import**, then browse and select the desired radix.

The radix is imported.

7. Click **Export**, select the radix, then enter a radix name.

The radix is exported.

8. Select the Apply user-defined radix to selected signal(s) checkbox.

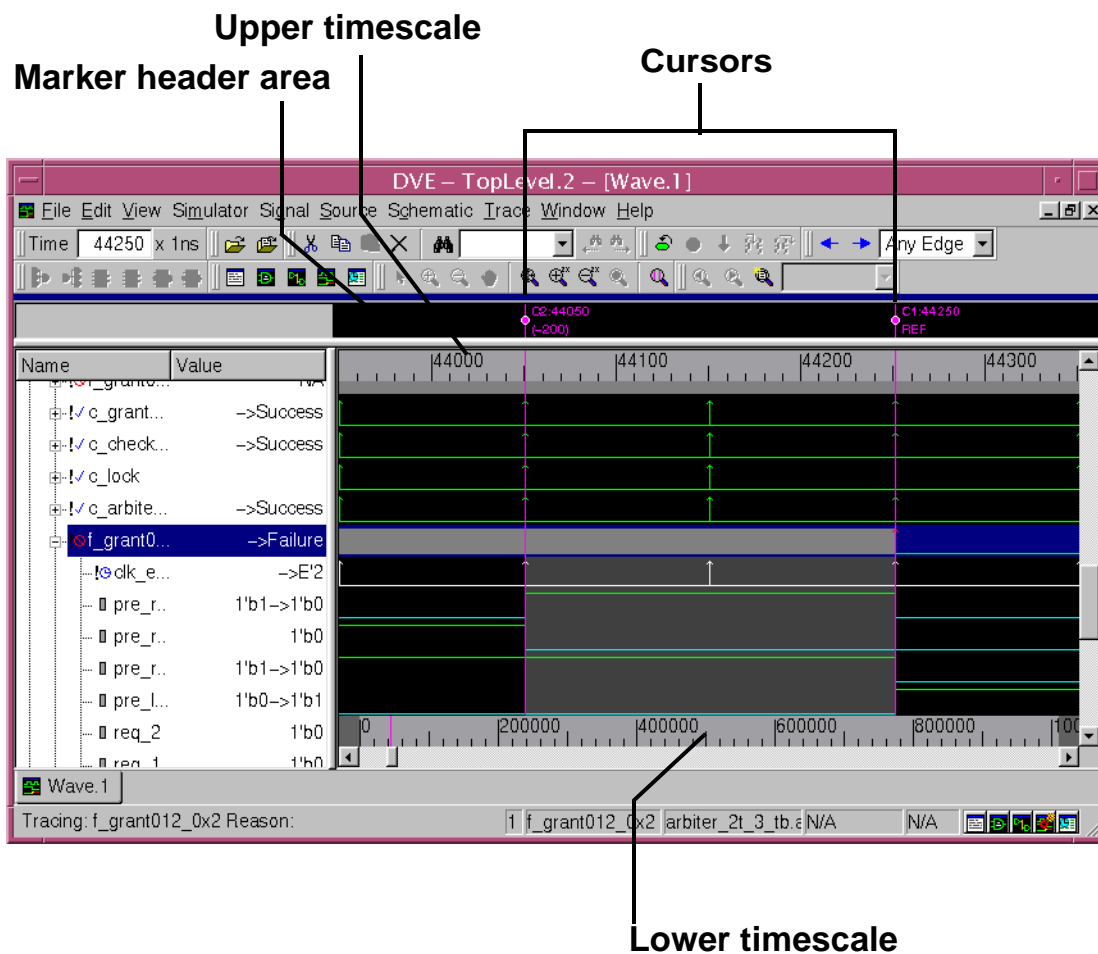
The user-defined radix is applied to the selected signal in the Wave view.

9. Click **OK** or **Apply** to save the user-defined radix.

Using the Waveform Pane

The Wave view displays the value transitions of signals and the real and vacuous successes and failures of assertions.

Figure 5-4 The Wave View



Cursors and markers are explained in “[Cursors and Markers](#)”.

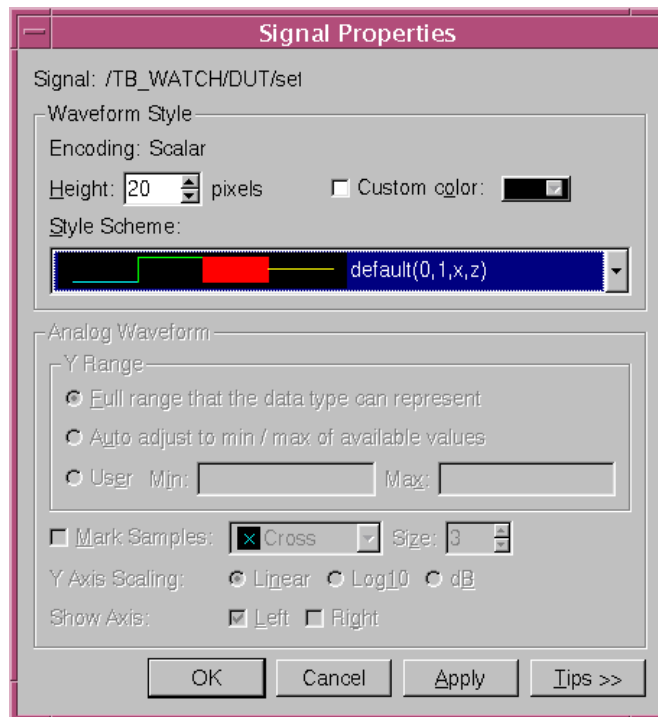
The Wave view has an upper and a lower timescale. The upper timescale displays the range of simulation times currently on display in the Wave view. The lower timescale displays the range of simulation times throughout the entire simulation.

Customizing Waveforms Display

To customize the display of waveforms

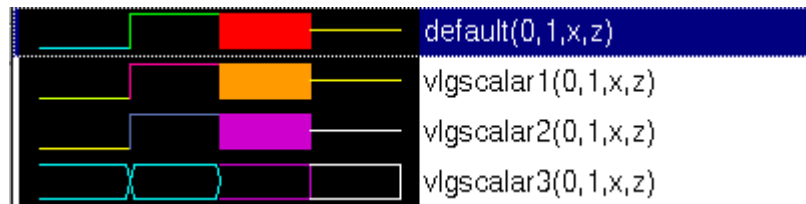
1. Select a signal in the Wave view and select **Signal > Properties**.

The Signal Properties dialog box opens:

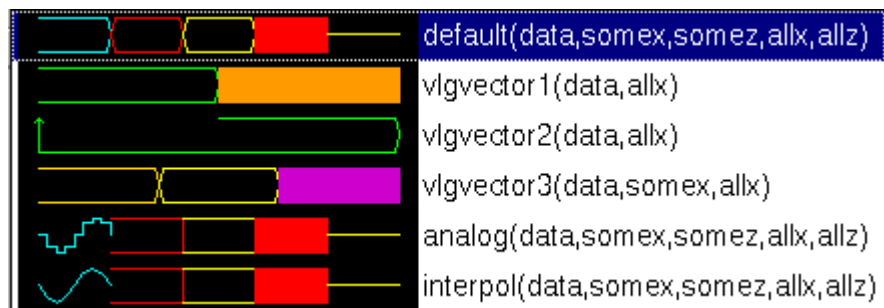


2. In the Waveform Style section, set a height and custom color for the waveform.
3. Set the Style Scheme as follows:

- For a scalar waveform, click the arrow and select from a scalar scheme as shown below:

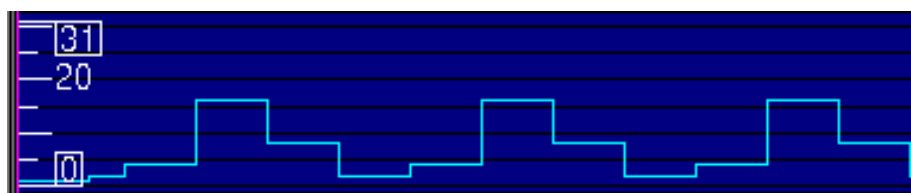


- For an analog waveform, click the arrow and select from a vector scheme, an analog scheme, or an interpolated scheme as shown below:

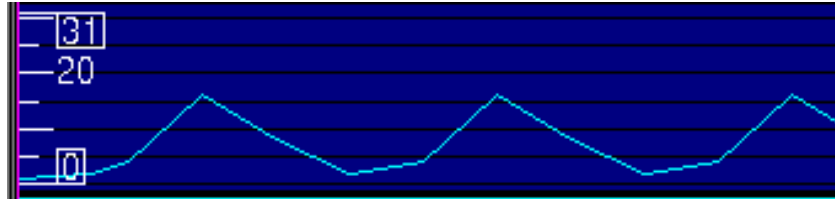


The default display is a vector scheme. Variant vector schemes alter the color and the values are displayed.

- The **analog** option displays an analog waveform as a stairstep scheme, that stays at the value until the next reported value change.



- The **interpol** option displays an analog waveform interpolated between each reported value change.



Note:

You cannot represent same signal in different "Drawing Style Scheme" (Digital and Analog).

4. If you are displaying the analog option, set the Y range values as follows:
 - Display the full range the data type can represent
 - Auto adjust the display to the minimum and maximum of available values
 - Display a user-defined minimum and maximum range
5. Select a Custom Resolution.
6. Select the Mark Samples check box and select the marker style and size.
7. Select Y-axis scaling from **Linear**, **Log_10**, or decibel (**dB**).
8. Set the Axis display.
9. Set the Controls as left or right.
10. Click **OK** to apply the settings and close the dialog box, **Apply** to apply changes and keep the box open, or **Cancel** to close the dialog box and disregard changes.

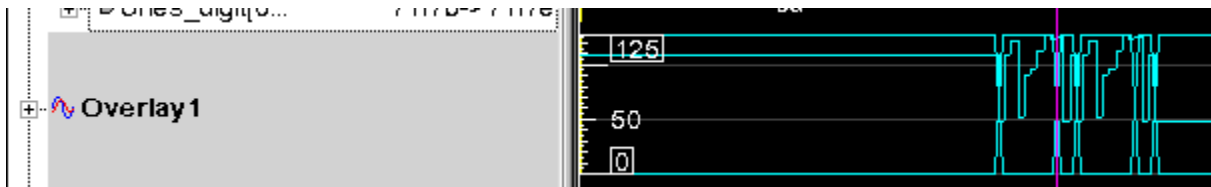
Overlapping Analog Signals

You can combine analog signal waveforms to visualize relationships between signals.

To overlap analog signals, do any of the following steps:

1. Drag and drop one or several analog signals onto an analog signal.

DVE creates a overlaying group as shown below:



2. Right-click the signal and select **Analog Overlay** from the menu.

The signals are overlapped.

3. Select a signal group with two or more analog signals, then right-click and select **Analog Overlay** from the CSM.

DVE overlay all signals in the group.

To restore the overlaid group back to a regular signal group, right-click the signal and select **Unoverlay** from the CSM.

Deleting Signal Group

When you delete a signal from the Wave view, it will be deleted globally. If the same signal is present in few other views, and you want to delete it, a warning message is displayed. You can either select to delete or hide the signal.

Once you select to delete the signal, it would be deleted globally from all views. If you select to hide the signal, it will be hidden in the current view.

If the signal groups are deleted, save session will not have the deleted signal groups. If you hide the signal, it will be hidden when you are saving or reloading the session.

Cursors and Markers

In the graphical display, you can insert markers and cursors.

Using Cursors

To insert cursors

1. Click the left mouse button to deposit cursor C1 in the graphical display.

The C1 cursors default position is at time 0.

2. Click somewhere else in the graphical display.

Cursor C1 moves to the new location.

3. Click the middle mouse button to deposit cursor C2 in the graphical display.

Similar to cursor C1, middle-click somewhere else in the graphical display and cursor C2 moves to this new location.

4. Place the mouse cursor on the round cursor handle in the cursor area, hold down the left mouse button, and drag the cursor to the desired location.

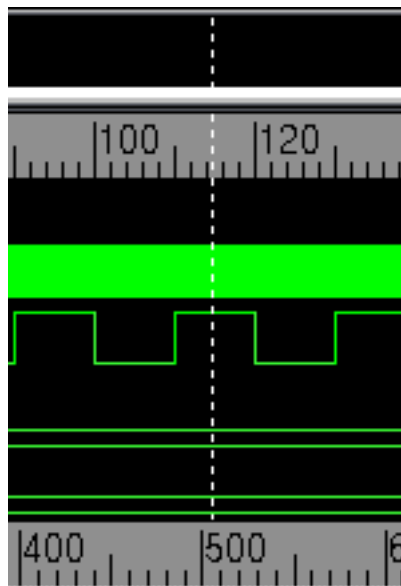
Creating Markers

Markers differ from cursors in the way you insert and move them. Like cursors, markers display the delta between the reference cursor (C1) and the marker. The Markers dialog box allows you to create, move, hide, delete markers, set the reference marker, and scroll the graphical display until it reveals a marker.

To create a marker

1. Right-click in the Waveform pane, and select **Create Markers** from the CSM.

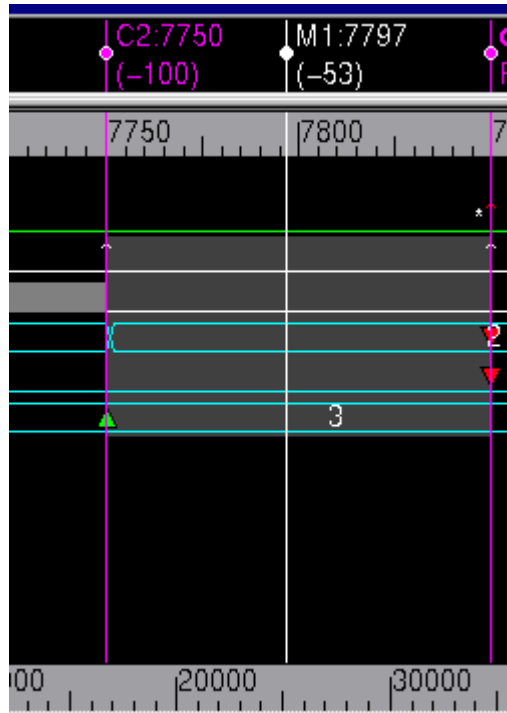
This inserts a dotted line on your mouse cursor in the graphical display.



The dotted line tracks the mouse cursor as you move the mouse in the waveform or marker header area.

2. Position the marker in the graphical display, then click to position the marker.

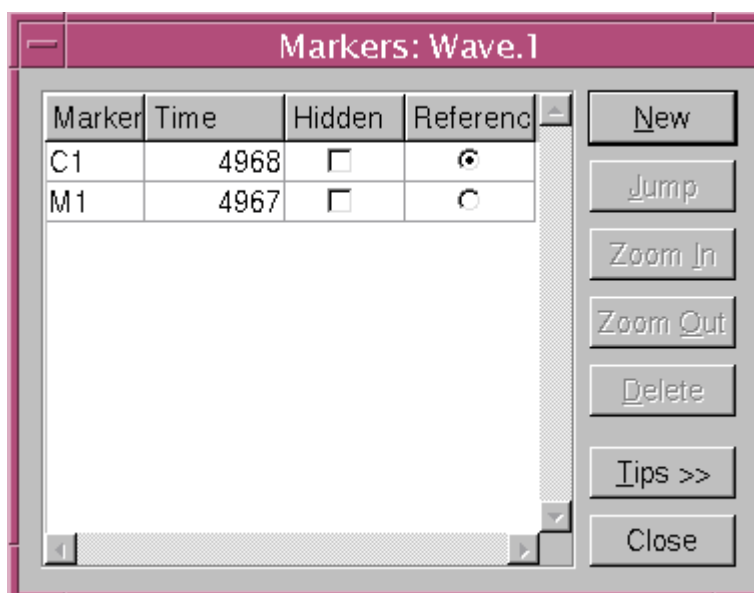
The marker annotation displays marker position and the delta between the marker and cursor



As you insert markers, DVE names them M1, M2, M3 and so forth.

3. Right-click a signal and select Markers.

The Markers dialog box opens:



4. Select the marker in the reference column.

The chosen marker is set as the reference marker.

5. Click the **Tips** button to expand the dialog box to display help for the Markers dialog box.

Zooming In and Out

You can zoom in to get a close-up view of your signal or zoom out to view the signal at its original size.

To choose zoom settings

1. Right-click in the waveform pane and select Zoom.
2. Select the zoom options as desired.

Note:

For more information about the zoom options, see the section entitled, [“View Menu” on page A-6](#).

Drag Zooming

You can drag the zoom to view specific transitions in a selected region of the timescale.

To drag zoom

1. Select any point in either of the timescales or in the waveform display area.
2. Hold down the left mouse button and drag the mouse to another time in the timescale.

The selected region is highlighted in blue.

3. Release the mouse button.

The signal is zoomed to the selected timescale.

Visualizing X at all Zoom Levels

You can visualize the X value at all zoom levels.

To visualize X value

1. Select a signal in the Signal pane with many value changes.
2. Zoom out the signal until the waveforms are condensed to yellow bar.

3. Right-click the signal and select **Highlight X Values** or select **Signal > Highlight X Values** from the menu.

The waveform is refreshed in the Wave view and X values are displayed as red color lines for all the zoom levels.

Expanding and Contracting Wave Signals

You can expand and contract the height of wave signals.

To expand and contract wave signals

1. Select **View > Increase Row Height**.

The signal is expanded.

2. Select **View > Decrease Row Height**.

The signal is contracted.

You can also right-click, select **Properties** and use the Signal Properties dialog box to increase or decrease row height.

Searching Signals

When searching signals, if you select the signals, only the selected signals are searched. If no signals are selected, all the signals are searched.

To search for signals

1. Select a signal in the Wave view.
2. Set the search constraint.

3. Enter the number of seeks.
4. Click the Search forward and Search backward arrows (see below) in the toolbar.



The C1 cursor moves from its current location to the next or previous location.

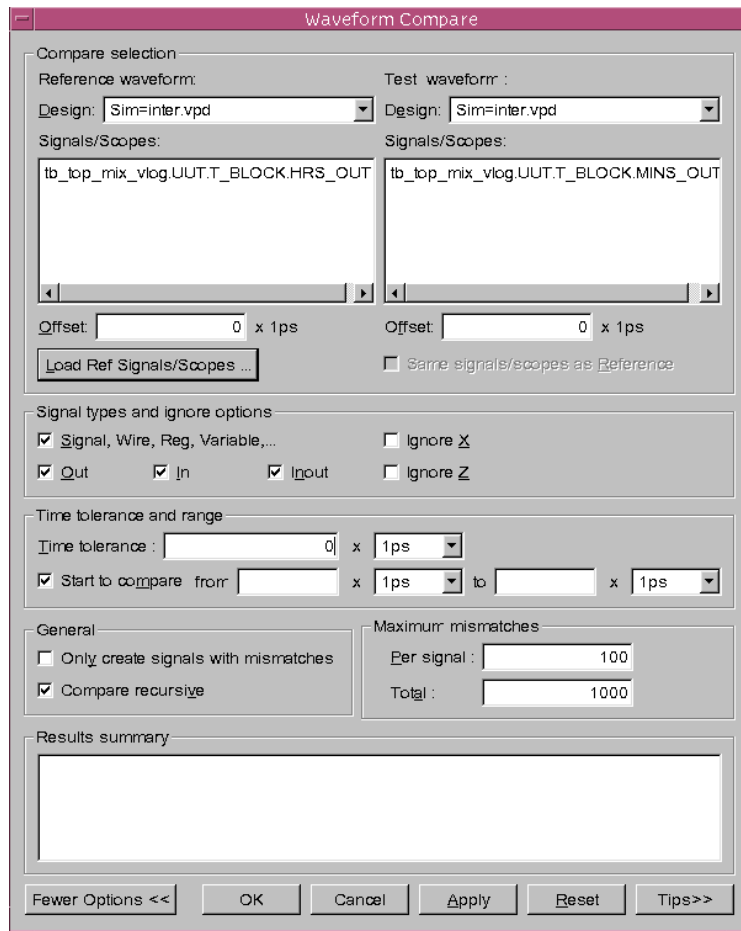
Comparing Signals, Scopes, and Groups

You can compare individual signals with the same bit numbers, scopes (for comparing variable children), buses, or groups of signals from one or two designs.

To view a comparison

1. Select one or two signals, signal groups, scopes, or buses from the Signal pane of the Wave view.
2. Right-click and select **Compare**.

The Waveform Compare dialog box opens.



3. Click **Load Reference Signals/Scopes** and select the text file with the signals and scopes to reference.

Note:

If you are comparing two designs from root, then the reference waveform region and test waveform region can be empty.

4. Click the **More Options** button.

The dialog box is expanded and additional options are displayed.

5. In the Signal types and ignore options section, select the signal types to compare and select ignore options.

For example, if you select Ignore X and if the reference signal value is X, there is always a match, whatever the values of the Test Signal.

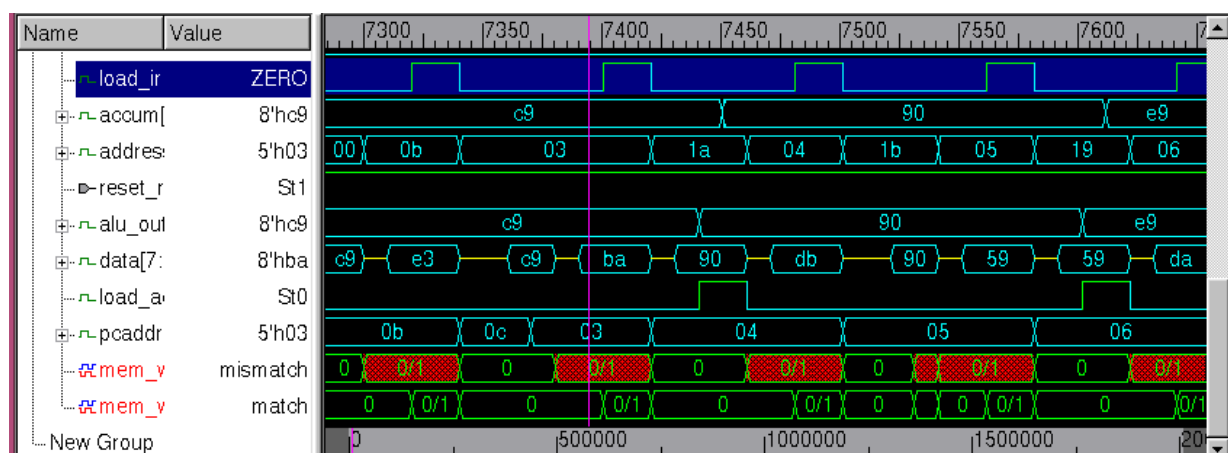
6. Enter a Time Tolerance to filter out mismatch values that have time ranges smaller than the tolerance range.
7. In the General section, select to compare recursively or to only create signals with mismatches.
8. Enter mismatch settings for maximum mismatches per signal and maximum total mismatches to report.
9. Click **Apply** to start the comparison and keep the dialog box open.

Or

Click **OK** to start the comparison and close the dialog box (you can open it at any time from the Signal pane CSM).

Results are displayed in the current Wave view.

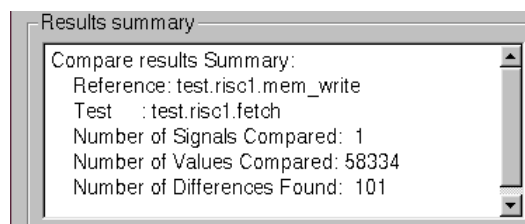
Figure 5-6 Compared Signal Groups in the Wave View



10. Select a result in the Wave view, right-click and select **Show Compare Info**.

The comparison results are displayed in the Waveform Compare dialog box.

Figure 5-7 Waveform Compare Summary Report



You can change the options, then compare them again.

Creating a Bus

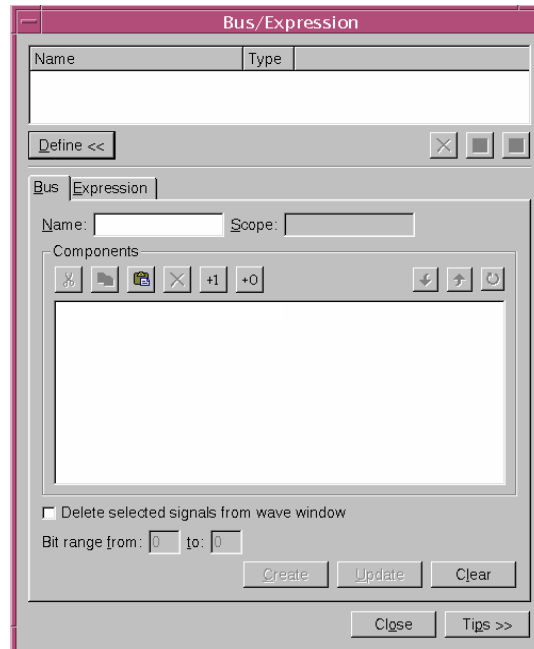
You can select few signals or components and group them to create buses. You can then view the behavior of the bus in the Wave or List view. You use the Bus Builder function to create and edit buses using the signals from the opened designs. You can drag and drop components from the Hierarchy pane or Data pane to create buses.

After you create a bus, you can select the bus in the Bus/Expressions dialog box and add it to the Wave or List view. By default, it will reside in the highest level signal group common to its components.

To create a bus

1. Select a few signals in the Wave view.
2. Right-click and select **Set Bus**.

The Bus/Expressions dialog box opens.



3. Click the **Define** button.

The Bus/Expressions dialog box is expanded.

4. Enter a name for the new bus.

You can give any legal name to the buses for the language (for example, Verilog or VHDL).

5. Enter a scope name to create the bus under a user-specified scope.
6. (Optional) Add constant +1 or constant +0 signal to the bus using the following icons:



7. Click **Create**.

The bus is created. The bit range is shown from 0 to N, where N is the number of components in the bus. Vectors and structures are expanded to their bits. For example, if “top.risc.pc[3..0]” is added to the list, it is added as four items.

Modifying Bus Components

You can edit an existing bus or modify the components and their order in a new bus using the Bus Builder toolbar.

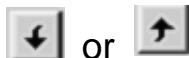
To modify bus components

1. Select the bus in the Wave view, then right-click and select **Set Bus**.
2. Select the components in the component list, then click the icon in the Bus Builder toolbar (see below):



The components are deleted.

3. Select one or more components in the component list, then click one of the following icons in the Bus Builder toolbar:



The components are moved up or down in the list.

4. Select two or more components from the component list, then click the following icon in the Bus Builder toolbar:



The order of components relative to each other is reversed.

5. Click **Update**.

The bus is updated.

Editing the Bus Bit Range

You can edit the bus bit range by simply changing either the MSB (most significant bit) or the LSB (least significant bit) of the bus.

To edit the bus bit range

1. Select the bus in the Data pane or from the Wave view signal pane and select **Signal > Set Bus** from the menu.

The Bus/Expression dialog box appears. You can also drag and drop the signal in the Bus/Expression dialog.

2. Double-click the signal and edit the range.

For example, define a signal as `a[0:7]`, change the bit range to `a[7:6]`, and click on create/update. A bus will be created/updated with the selected 2 bits as `a[7]`, `a[6]`.

If you enter a wrong range, a warning message is displayed and the text color of the signal becomes red. You need to enter the correct format to save the bit range.

3. Select the signal and click the “+” icon to expand.

The signal is expanded in the same order as specified in the name. For example, `a[1:3]` signal will expand to `a[1]`, `a[2]`, `a[3]`.

You can perform all the toolbar operations on the expanded signals.

Creating an Expression

You can create expression for signals in the Wave view to view the value change when that expression occurs. You use the Expressions tab in the Bus/Expressions dialog box to create and modify expressions.

Note:

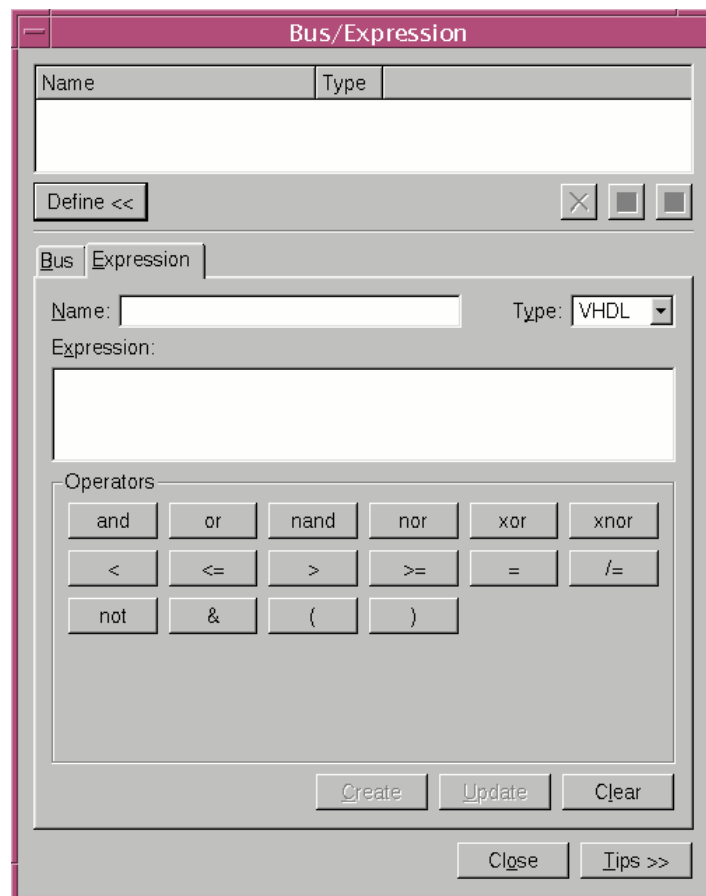
Complex SystemVerilog data types are not supported.

To create an expression

1. Select a signal in the Wave view, right-click and select **Set Expressions**.

The Bus/Expressions dialog box opens.

Figure 5-8 Expression Tab



2. Enter a name for the expression.
3. Select an expression type.
4. Insert the operators into the expression.
5. Click **Create**.

The expression is created.

6. Select the expression and click the Add to Wave or Add to List icons.

The expression is added in the Wave or List view. You can now run the simulation and view the values of signal when the expression occurs.

Viewing Bus Values

To view bus values in waveforms (with transitions on edges), position the cursor on the waveform.

A ToolTip is shown at a transition displaying the transition values.

Creating and Updating Counters

You can create counter signal to count the value transitions for expression or signal. Counter is treated as a special expression and you can create or update counter in the same way as you create an expression in the Bus/Expression dialog box.

The counter is supported both in post processing mode and interactive mode. In interactive mode, with simulation going on, the count result will be updated as other signals in the Wave view.

To create and update a counter

1. Select a signal in the Data pane and select **Signal > Set Expressions** from the menu.

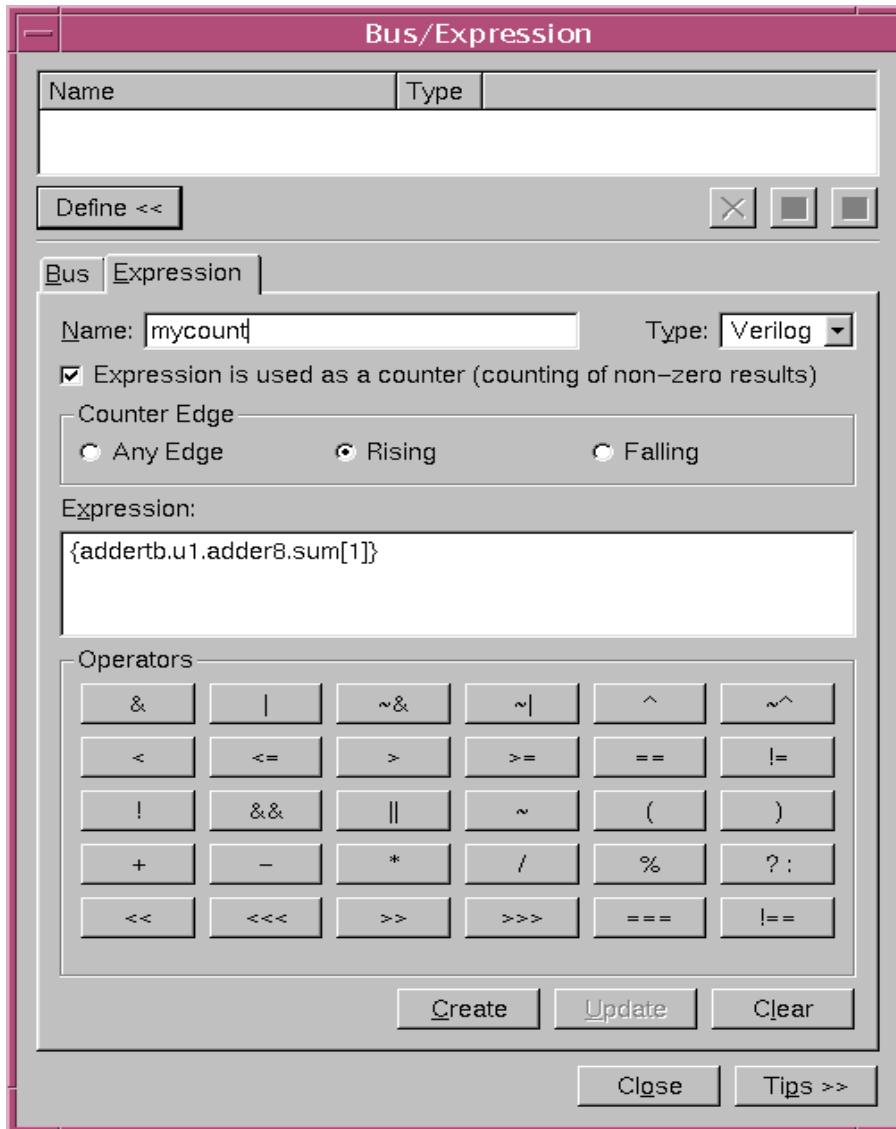
The Bus/Expressions dialog box is displayed.

2. Click the **Expression** tab, then drag and drop the signal that you want to create for expression counter from the Wave view or Source view under the Expression area.
3. Type a counter name in the Name field.

For example, EXP:mycount.

4. Select the checkbox **Expression is used as a counter (counting for non-zero results)** to create a signal that represents the value transitions of the expression.

The Counter Edge radio buttons get enabled.



5. Select any of the following **Counter Edge** as desired.

- Any Edge - This option is for any expression/signal. Whenever the value of the expression/signal changes, counter signal counts this transaction.
- Rising - (Default) This option is only for bit type signals. Whenever the expression/signal changes from low to high, counter signal counts the transaction.
- Falling - This option is only for bit type signals. Whenever the expression/signal changes from high to low, counter signal counts the transaction.

6. Click **Create**.

The newly created expression counter is displayed in the Bus/Expressions dialog box. You can also view the count results in the Wave view.

7. Drag and drop the counter in the Bus/Expression dialog box to update it.

For example, you can update the counter edge from “Rising” to “Any Value”.

8. Click the **Update** button.

The counter is updated. You can view the count results in the Wave view.

Shifting Signals

You shift a signal by creating a new signal based on a time shifted signal.

To shift a signal

1. Select a signal in the Wave view.

2. Select **Signal > Shift Time**.

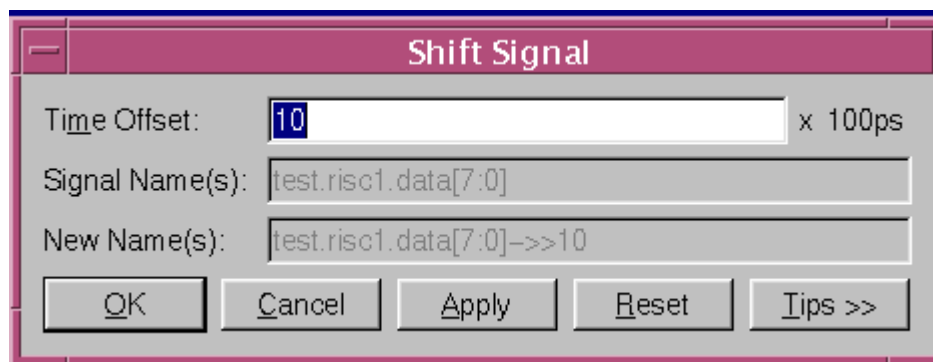
The Shift Signal dialog box opens.

3. Enter a positive Time Offset.

The signal is shifted to the right.

4. Enter a negative number.

The signal is shifted to the left in the Waveform pane.



The signal displays with the original signal name followed by the time offset. In the previous figure, it is `test1.risc.daata(7:0)->>10`.

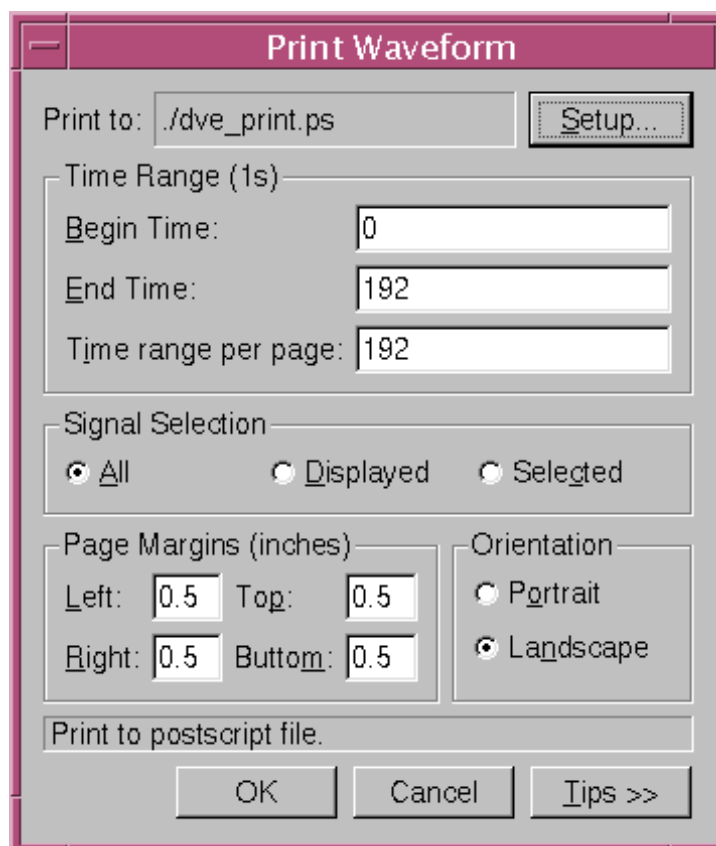
Printing Waveforms

You can print waveforms to a file or printer from an active Wave view selecting time range and signals to print.

To print waveform

1. From an active Wave view, select **File > Print**.

The Print Waveform dialog box opens.



2. Click the set up button to set printing options:
 - Printer or print to file
 - Print in color or grayscale
 - Print orientation and paper size
 - Print options such as range and number of copies

3. Select the total time range to print and the time range to print per page.
4. Select whether to print **All**, **Displayed**, or **Selected** signals.
5. Select the page margins.
6. Choose **Landscape** or **Portrait** orientation.
7. Click **OK**.

The waveform is printed.

6

Using the List View

The List view displays simulation results in tabular format. For Verilog, the List view displays nets and register variables. For VHDL, it displays signals and process variables.

This chapter includes the following topics:

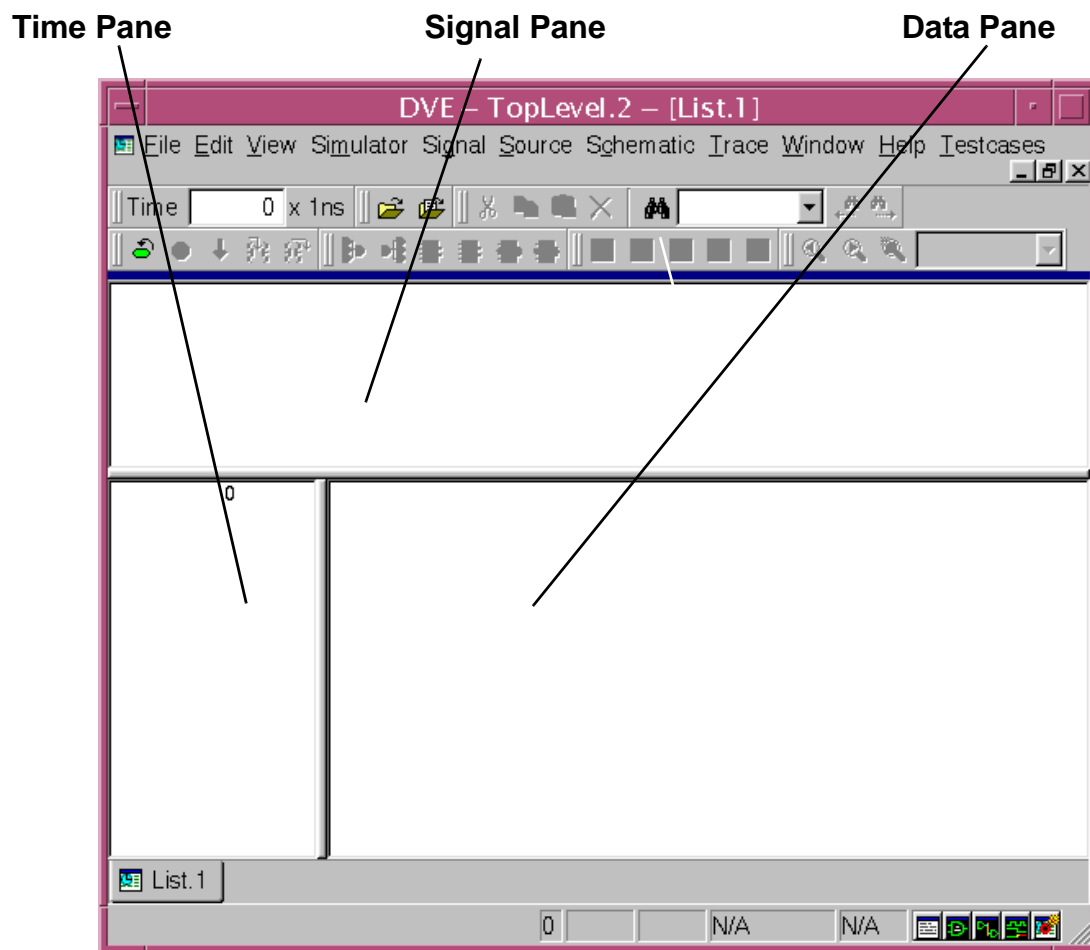
- “The List View”
- “Displaying Data”
- “Navigating Simulation Data”
- “Setting Signal Properties”
- “Comparing Signals”
- “Saving a List Format”

The List View

The List view contains three panes:

- The Signal pane displays signal names as headers above the simulation data.
- The Data pane displays simulation results in tabular format.
- The Time pane displays simulation time value.

Figure 6-1 The List View



Displaying Data

The List view displays results similar to the Wave view, but it displays the results in tabular form. You can display data in the List view in the same way as you display in the Wave view. You can perform the following tasks in the List view:

- Open a database

- Open a session
- View simulation data in tabular format
- Create markers
- Set signal properties
- Associate signals with any database
- Compare signals
- Save signal values

Dragging and Dropping Signals into the List view

To populate the List view with data from other DVE windows drag and drop into the List view, a scope or assertion of interest from the Hierarchy pane, the Source view, the Wave view, or from either tab of the Assertion window.

The data is displayed in the default format.

Opening a Database

To open a database

1. Click the following icon in the toolbar:



The List view opens.

2. Choose **File > Open Database.**

The Open Database dialog box opens.

3. Browse and select the .vcd, .vpd, or .dump database file.

The data is displayed in the default format.

Loading a Session

To load a session

1. Click the following icon in the toolbar:



The List view opens.

2. Select **File > Load Session**.

The Load Session dialog box opens.

3. Select the .vcd, .vpd, or .dump database file.

The data is displayed in the default format.

Navigating Simulation Data

This section includes the following topics:

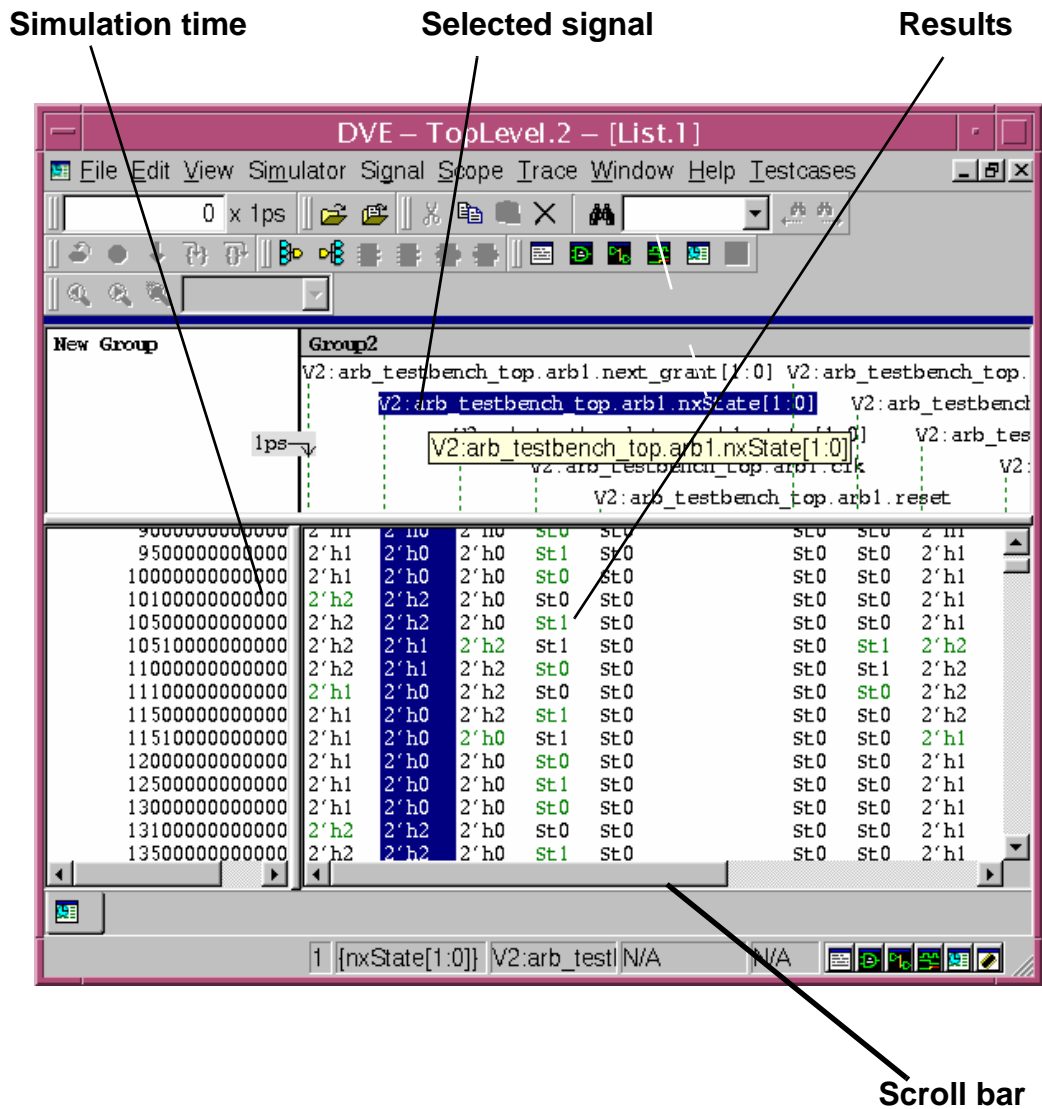
- Viewing Data in the List view
- Using Markers

Viewing Data in the List view

To view simulation data in the List view

- Use the bottom scroll bar to move left and right and view signals and their values.
- Use the right scroll bar to move up and down through simulation time.

- Select a signal in the signal pane to highlight the signal values.



Using Markers

You can create markers in the List view to speed up navigation.

To create and delete markers

1. Select a time unit in the Time pane, right-click and select **Markers**.
The Markers.List dialog box opens.
2. Click **New** to create a new marker in the list table.
3. Select the Time cell for the new marker and enter the time at which to set the marker.
4. Click **Hidden** if you do not want to display the marker in the Data pane, then click **Return**.
5. Repeat steps 2 through 4 to create additional markers.
6. Select a marker in the Markers.List dialog box, then click **Jump**.
The selected marker is displayed in the Time pane.
7. Select a marker and click **Delete** in the Marker.List dialog box.
The marker is deleted from the list.

Setting Signal Properties

To customize signal display, you set signal properties for individual signals.

To set the signal properties

1. Select a signal in the Signal pane.
2. Select **Signal > Signal Properties**.
The Signal Properties dialog box opens.

3. Enter the number of characters for the selected signal value column width.
4. Select whether a signal value change triggers a new line of values in the Data pane or not.
5. Click **Apply** to make the change and keep the dialog box open.

Or

Click **OK** to apply the changes and close the dialog box.

Comparing Signals

You can compare signals in the List view similar to the way you compare signals in the Wave view.

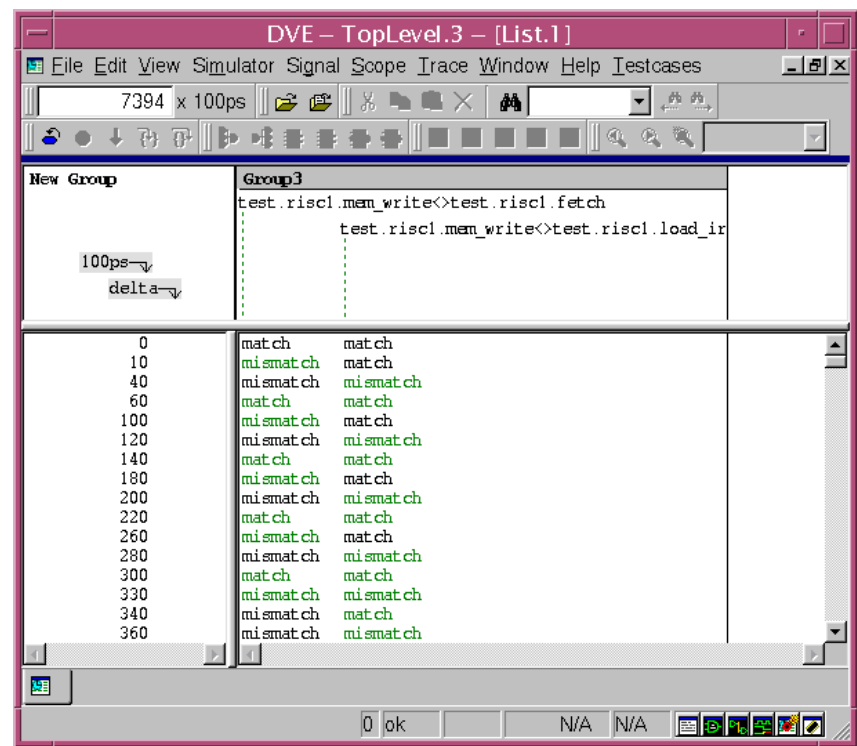
To compare signals

1. Select signals from the Signal pane.
2. Right-click and select **Compare**.

The Waveform Compare dialog box opens.

3. Select the options, as appropriate.
4. Click **OK**.

The comparison results are displayed in the current List view.



For more information about the Waveform Compare dialog box, see the topic [“Comparing Signals, Scopes, and Groups”](#) on page 5-24.

Saving a List Format

After you customize the display in the List view, you can save the format for future use.

To save the list format

1. Select **File > Save Values**.

The Dump Text List dialog box opens.

2. Select the format, tabular or event based.
3. Enter a filename with a `.tcl` extension.
4. Select **Save**.

The list format is saved.

7

Using Schematics

This chapter includes the following topics:

- “Overview”
- “Opening a Design Schematic View”
- “Opening a Path Schematic View”
- “Finding Signals in a Schematic View”
- “Following a Signal Across Boundaries”
- “Tracing X Values in a Design”
- “Printing Schematics”

Overview

Schematic views provide a compact, easy-to-read graphical representation of a design. You can view a design, scope, signal, or group of selected signals and select ports to expand connectivity in relevant areas. You can explore the design behavior by analyzing the annotated values for ports and nets.

There are two types of schematic views in DVE:

- Design - a design schematic shows the hierarchical contents of the design or a selected instance and allows you to traverse the hierarchy of the design.
- Path - a path schematic is a subset of the design schematic that displays where signals cross hierarchy levels. Use the path schematic to follow a signal through the hierarchy and display portal logic (signal effects at ports).

Note:

Your design must be compiled in the same version of VCS that you are currently pointing to in your session.

Viewing Schematic

When viewing the schematic, use the scroll bars to move up and down and left and right in the displayed graphics. You can also use toolbar and menu commands to select parts of the design to zoom in, copy, drag and drop into another DVE window, move one level up to a parent or definition. You can also add signals from the Schematic view to the Wave view, List view, or Source view.

To customize the schematic display

1. Set the maximum number of cells in the schematic.
2. Change the text style and size displayed on your schematic.
3. Change the visibility and colors of cells, hierarchical crossings, nets, buses, ports, pins, and rippers.

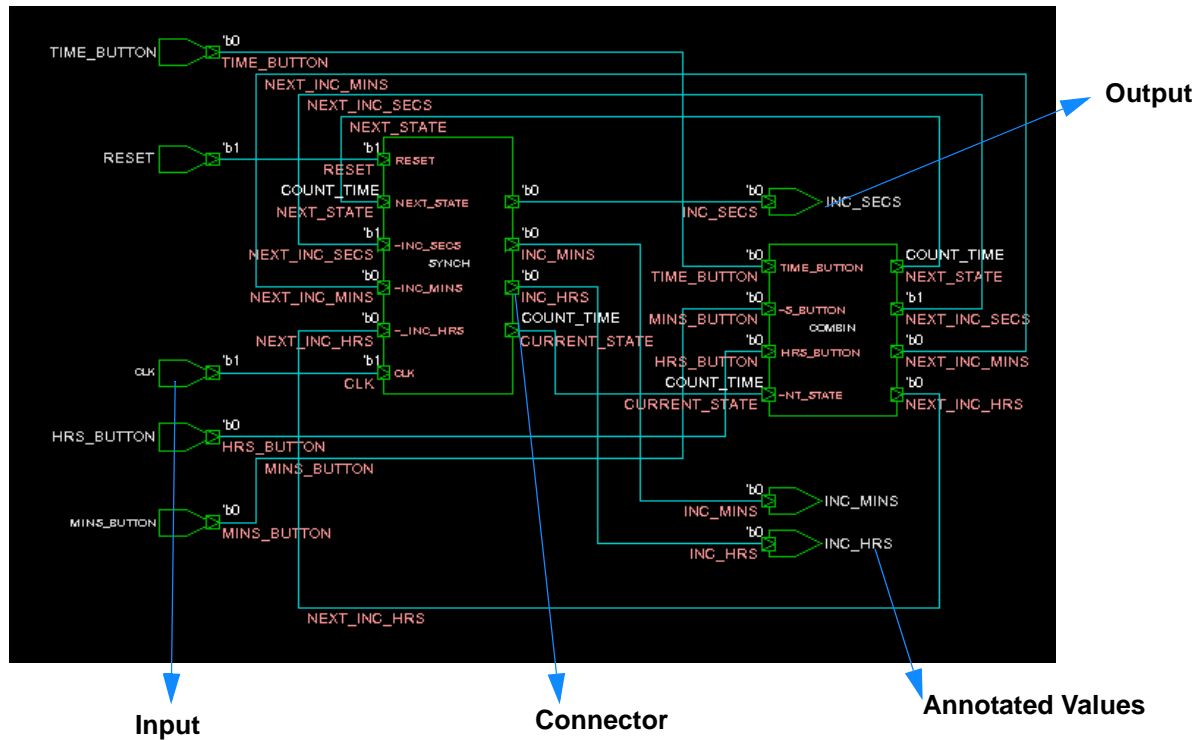
Opening a Design Schematic View

To view a schematic of an open design in DVE, you must generate your simulation on the same platform where you run DVE and use the `-debug` command-line option to compile.

To open a design schematic

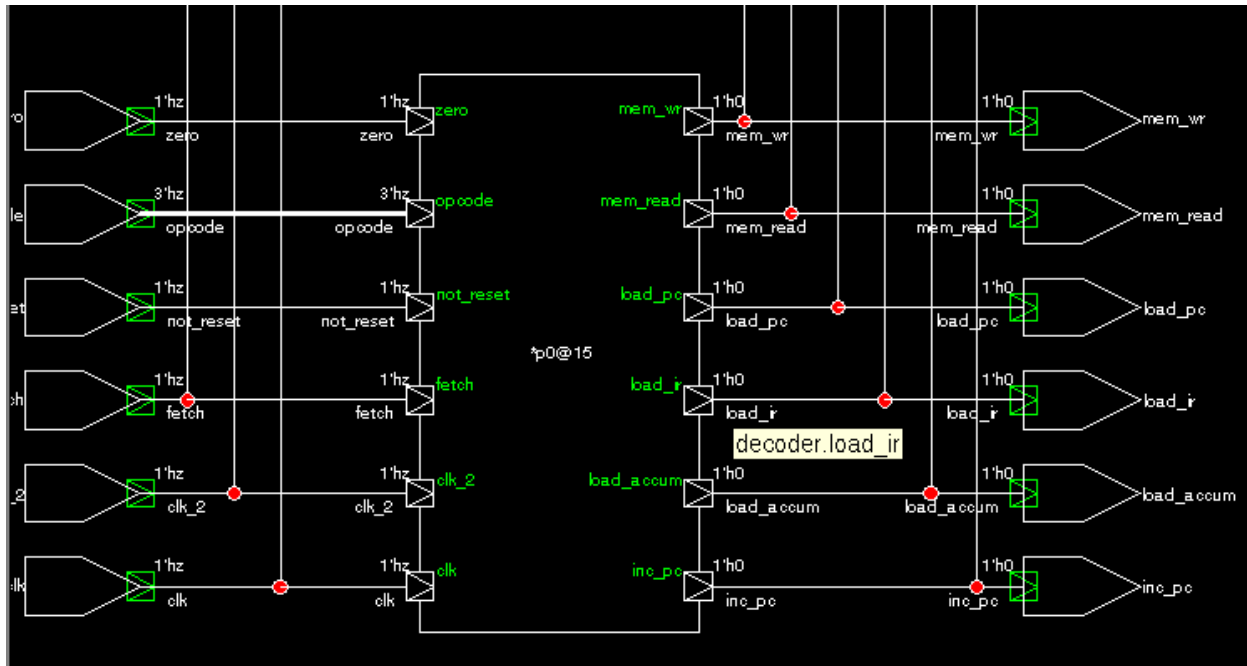
1. Select an instance from the Hierarchy pane, right-click and select **Show Schematic**.

The Schematic view displays the connectivity in the selected instance.



Annotating Values

To view the annotated values of a signal, select **Schematic > Annotate Values**.



Note:

If you hold the cursor on a signal, a ToolTip identifies the signal as shown in the previous diagram.

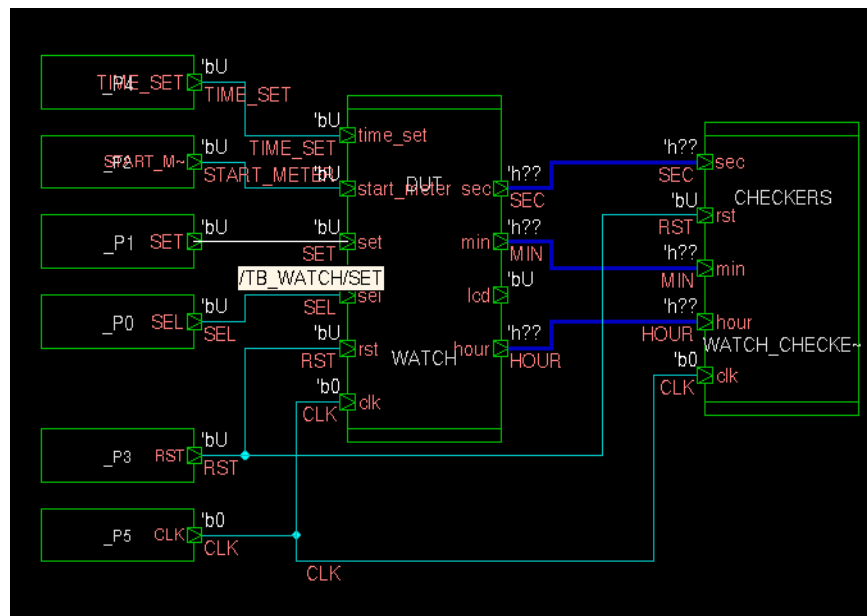
Opening a Path Schematic View

A path schematic is a subset of the design schematic displaying connections that cross hierarchical boundary.

To open a path schematic view

1. Open a design schematic view of an instance containing the hierarchical crossings of interest.
2. When you have identified the instance to display, click on the instance to select it.

The color change indicates that it is selected.



Note:

You can also drag the selection cursor over multiple objects to select multiple items.

3. Right-click and select **Show Path Schematic** or click the following icon in the toolbar to view a path schematic in a new window:



The path schematic for the selection is displayed:



Displaying Connections in a Path Schematic

With a path schematic displayed, you can add the logic fanin to, or logic fanout from, specified objects in the schematic across specified levels or the entire design.

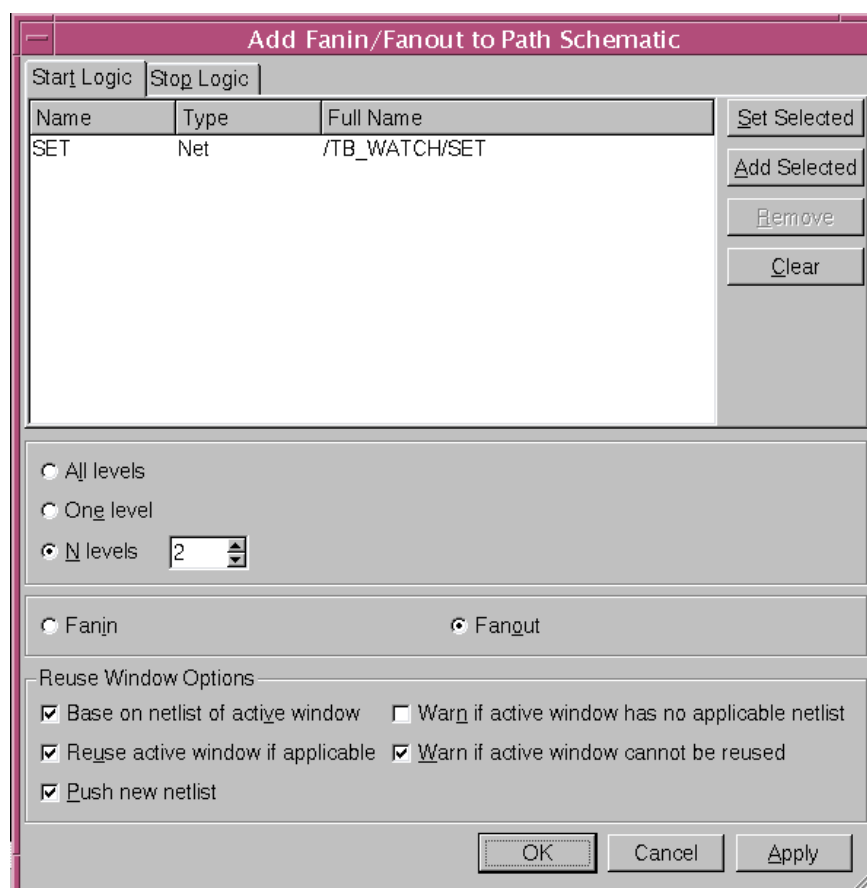
To display connections

1. Select an object in the path schematic.

The change in color confirms the selection.

2. Select **Scope > Add Fanin/Fanout**.

The Fanin/Fanout to Path Schematic dialog box opens.

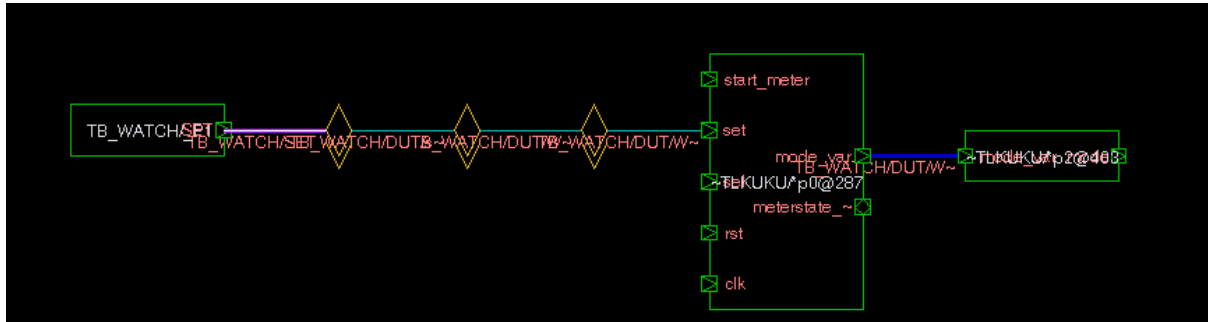


3. Click **Set Selected** to add the selected objects to the list box.

You can optionally select more objects and use the **Add Selected** button to add them to the list.

4. Set the other options, such as the number of logic levels to be added and the Reuse Windows options.
5. Click **OK**.

The schematic is updated with additional fanin or fanout logic. You can also view the signal values.



Finding Signals in a Schematic View

Manually Tracing Signals

You can select one or more signals to trace in a Schematic or Path Schematic view. With this option, the selected signals are highlighted based on specified line colors you select.

To highlight signals

1. Select **Trace > Highlight**, then select **Set Current Color**.
2. Select a color for the highlight.
3. Click **OK**.

The signal is highlighted with the selected color.

Searching for Signals

You can use the Find toolbar option to locate signals.

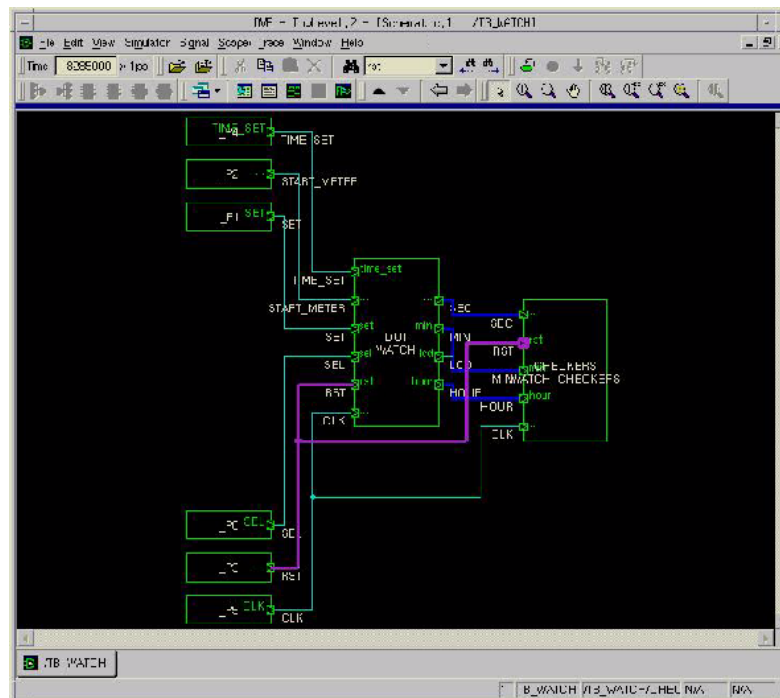
To locate signals

1. Enter the signal name in the Find toolbar box in the Schematic view, then click the **Find Next** toolbar button.

The signal becomes highlighted in the schematic.

2. With the signal selected, click the **Trace Drivers** toolbar button, or select the **Trace-Drivers** menu item.

The signal is highlighted in purple.

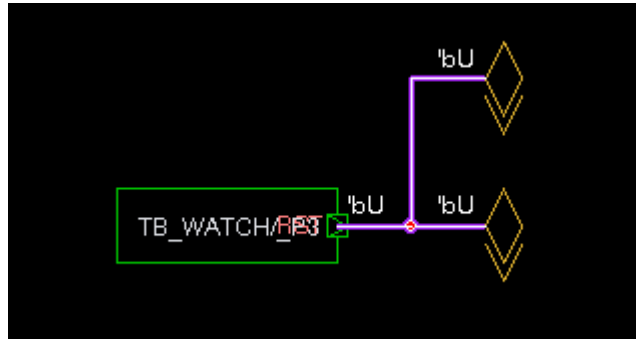


Following a Signal Across Boundaries

You can select a signal and follow it across hierarchical boundaries in the Path Schematic view.

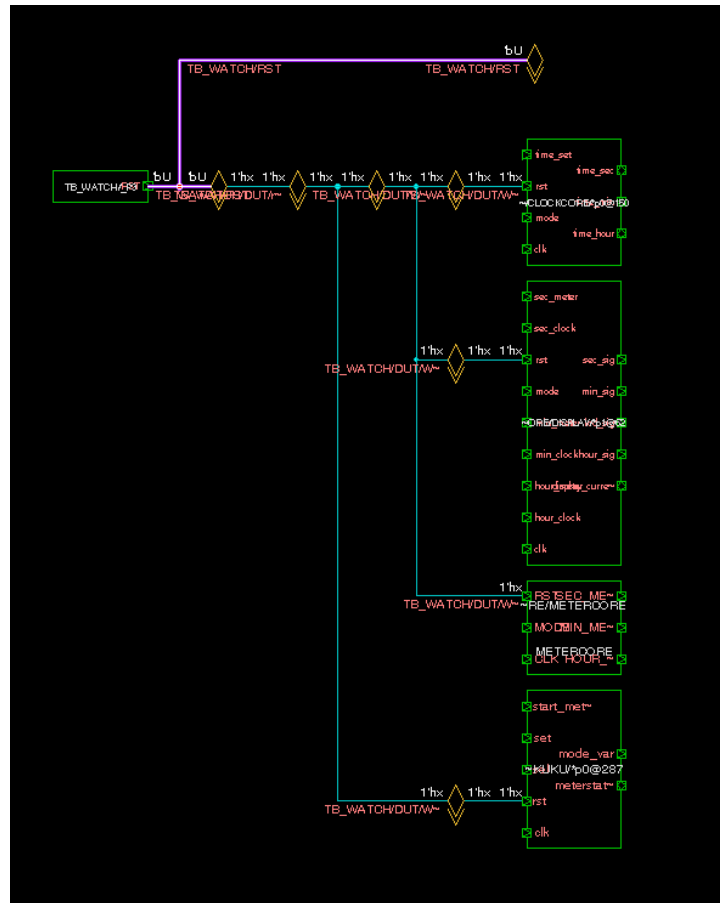
To follow a signal

1. Select a signal or signals, right-click and select **Show Path Schematic**.
2. Select a signal in the Path Schematic view.



3. Right-click and select **Expand Path** from the CSM.

The signal is highlighted in the path view.



Tracing X Values in a Design

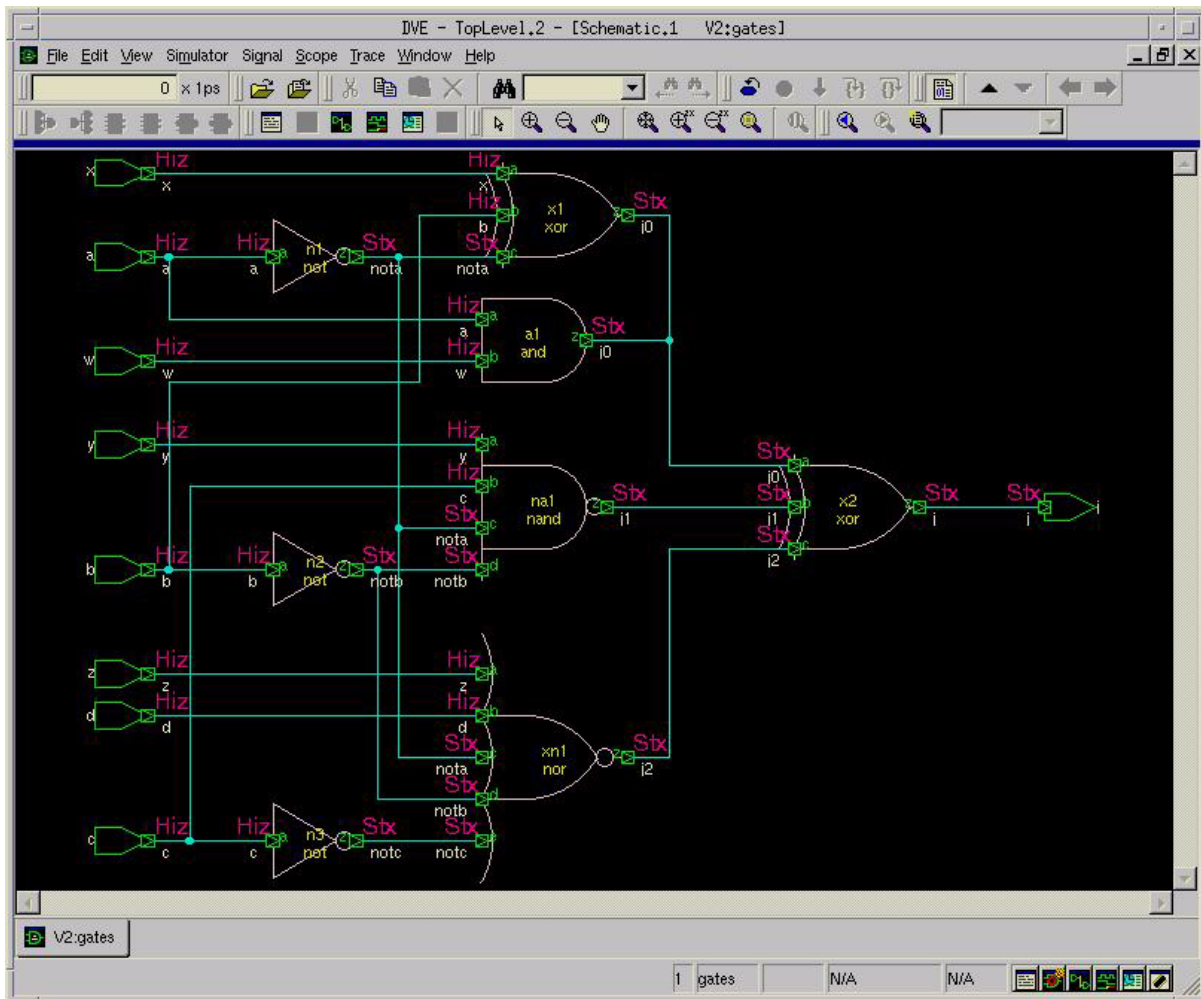
You can trace an X value through a design, for example, across gates, to identify the signal that caused the X value.

To trace an X value

1. Select an instance from the Hierarchy pane, then right-click and select **Show Schematic**.

By default, a new Schematic view opens in the main window as a tabbed window.

2. Zoom schematic to fit window.
3. Click the **Annotate Values** toolbar button in the Schematic view.

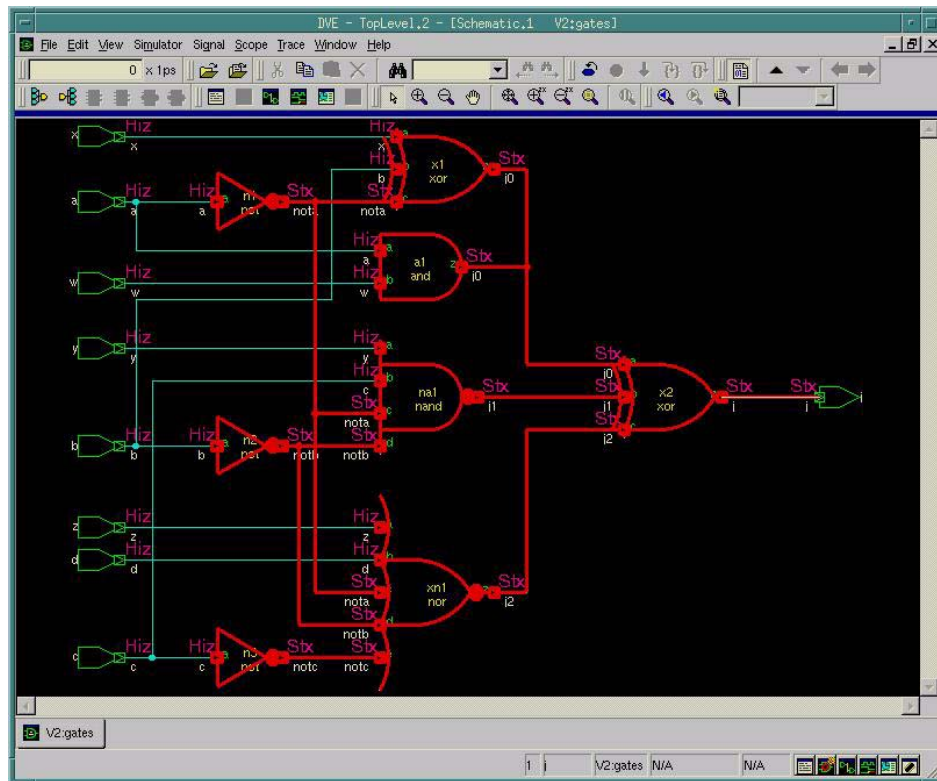


4. Select **Schematic view** category.
5. Select the signal with an X value.

The signal wire turns white when selected.

6. Select **Trace > Trace X**.

The X value is traced to its source signals making it easy to identify the signals that caused the X value.



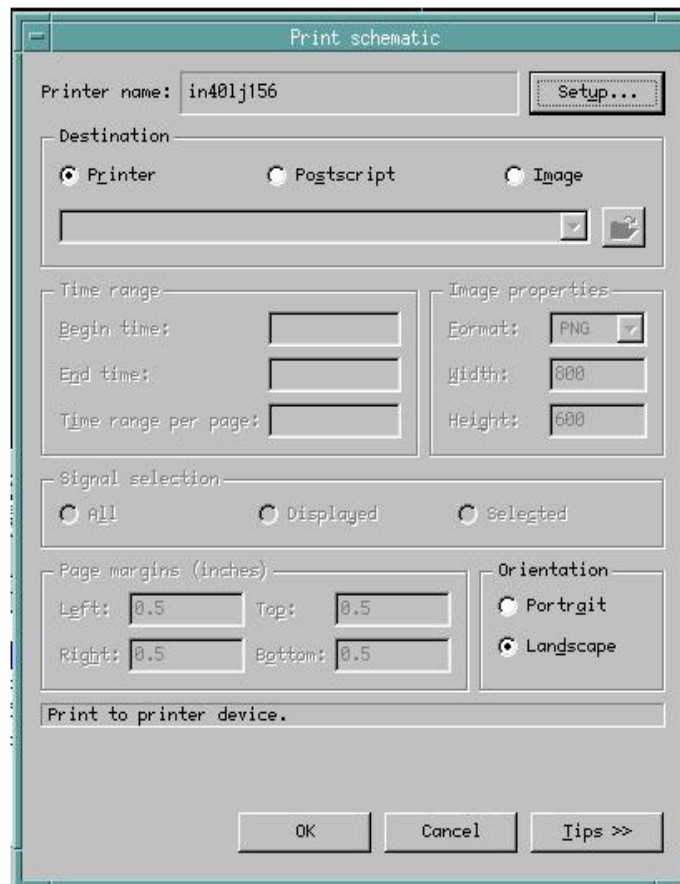
Printing Schematics

You can print schematics to a file or printer from an active Schematic or Path Schematic view selecting time range and signals to print.

To print schematics

1. From an active Schematic or Path Schematic view, select **File > Print**.

The Print Schematic dialog box opens.



2. Click the Setup button to set printing options:
 - Printer or print to file
 - Print in color or grayscale
 - Print orientation and paper size
 - Print options such as range and number of copies
3. Select whether to print **All**, **Displayed**, or **Selected** signals.
4. Select the page margins.

5. Choose **Landscape** or **Portrait** orientation.
6. Click **OK**.

The schematic is printed.

8

Working with Assertions and Cover Properties

This chapter describes viewing and working with assertion results. It includes the following topics:

- [“Compiling SystemVerilog Assertions”](#)
- [“Viewing Assertion Results”](#)
- [“Debugging Assertions”](#)

Compiling SystemVerilog Assertions

Use the `-assert dve` flag on the VCS command line when compiling SystemVerilog assertions (SVA) for debugging with DVE.

Note:

The link step can take a long time if you use a Solaris linker prior to version 5.8.

To avoid linking delays when using DVE to debug designs compiled on Solaris, perform either of the following:

- Make sure your Solaris C compiler is version 5.8 or above. To check your compiler version, enter the following on the command line:

```
ld -V
```

The system returns your linker version, for example:

```
ld: Software Generation Utilities - Solaris Link Editors:  
5.8-1.283
```

- Use the gcc C compiler when compiling your design. For example:

```
vcs -assert dve -debug_pp -sverilog a.v -ld gcc
```

Viewing Assertion Results

The Assertion Pane displays SVA and OVA assertion and cover property results:

- DVE displays assertion results by instance include start and end times of assertion events, the delta, the offending string in assertion failures, and totals for failures, real successes, vacuous successes, incompletes and attempts. Successful assertions are displayed in green, vacuous successes in brown, and failed assertions in red.

Figure 8-1 Assertion Results

Name	Instance	Start	End	Delta	Reason	Failures	Incompletes	Successes	Attempts
HOUR_24_CHECKER	/TB_WATCH/CHECKERS	0	0	0		0	0	10000	10000
MIN_60_CHECKER	/TB_WATCH/CHECKERS	0	0	0		0	0	10000	10000
RESET_CHECKER	/TB_WATCH/CHECKERS	8295000	8295000	0	(((hour == 5'b0) && (min == ...	15	0	19985	20000
SEC_60_CHECKER	/TB_WATCH/CHECKERS	9410000	9410000	10000	(sec == 6'b0)	137	0	9863	10000

Note:

Use the VCS -assert DVE command-line switch with the -debug_pp flag to enable SVA tracing in DVE. If you do not enable SVA tracing, assertion value changes will still be dumped into the VPD file and be visible in the Wave view, but assertion attempts cannot be traced.

- Cover properties results, shown in Figure 8-2, display instance include start, end time, and the delta, and the total number of matches, mismatches, incompletes, and attempts.

Figure 8-2 Cover Properties

Name	Instance	Start	End	Delta	Matches	Mismatches	Incompletes	Attempts
HOUR_24_COVER	/TB_WATCH/CHECKERS	0	0	0	10000	0	0	10000
MIN_60_COVER	/TB_WATCH/CHECKERS	0	0	0	10000	0	0	10000
RESET_COVER	/TB_WATCH/CHECKERS	0	0	0	19985	0	0	19985
SEC_60_COVER	/TB_WATCH/CHECKERS	0	0	0	9863	0	0	9863

Setting Display Criteria

You use the Assertion pane navigation bar to display results based on selected criteria, as shown in Figure 8-3.

Figure 8-3 Assertion and Cover Properties

Assertions Window

Click to display assertion or cover properties results.

Specify display criteria by starting and ending time.

Select a condition for the results display.

Name	Instance	Start	End	Delta	Reason	Failures	Incompletes	Successes	Attempts
HOUR_24_CHECKER	/TB_WATCH/CHECKERS0	0	0	0		0	0	10000	10000
MIN_60_CHECKER	/TB_WATCH/CHECKERS0	0	0	0		0	0	10000	10000
RESET_CHECKER	/TB_WATCH/CHECKERS	8295000	8295000	0	((hour == 5'b0) && (min =...	15	0	19985	20000
SEC_60_CHECKER	/TB_WATCH/CHECKERS	9400000	9410000	10000	(sec == 6'b0)	137	0	9863	10000

Cover properties Window

Name	Instance	Start	End	Delta	Matches	Mismatches	Incompletes	Attempts
HOUR_24_COVER	/TB_WATCH/CHECKERS0	0	0	0	10000	0	0	10000
MIN_60_COVER	/TB_WATCH/CHECKERS0	0	0	0	10000	0	0	10000
RESET_COVER	/TB_WATCH/CHECKERS0	0	0	0	19985	0	0	19985
SEC_60_COVER	/TB_WATCH/CHECKERS0	0	0	0	9863	0	0	9863

To set display criteria by starting ending time

1. Click the arrow in the display control bar and select **all**, **starting**, **ending**, or **starting and ending**.

The display criteria selections become active.

2. Select **at time** or **from**, then select **begin**, **current**, or **end** to specify the window.

3. Specify the condition as follows:

- For assertions, select failures, incompletes, successes, vacuous successes, or all.
- For cover properties select **uncovered**, **covered** or **all**. Note that the default is to display uncovered.

Debugging Assertions

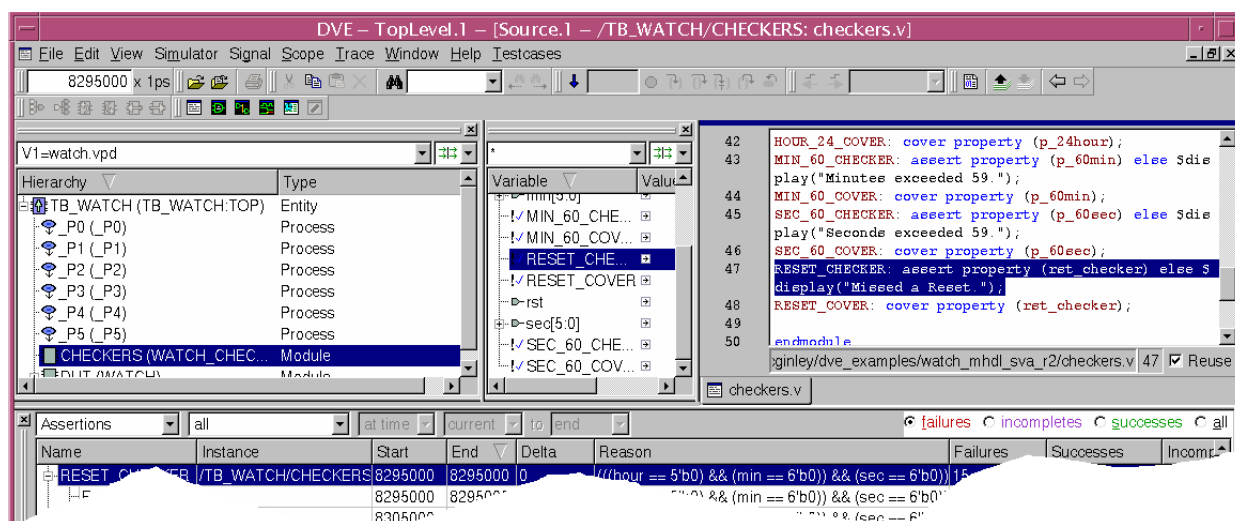
When you open a design that contains assertions, DVE displays the Assertion Pane even if all the assertions pass. The default is to display failed assertions. To display the first 10 failures and successes, click the "+" next to an assertion of interest. [Figure 8-4](#) illustrates the attempt list.

Figure 8-4 Assertion Attempt List

Name	Instance	Start	End	Delta	Reason	Failures	Successes	Incompletes
RESET_CHECKER /TB_WATCH/CHECKERS		8295000	8295000	0	((hour == 5'b0) && (min == 6'b0)) && (sec == 6'b0)	15	19985	0
Failure1		8295000	8295000	0	((hour == 5'b0) && (min == 6'b0)) && (sec == 6'b0)			
Failure2		8305000	8305000	0	((hour == 5'b0) && (min == 6'b0)) && (sec == 6'b0)			
Failure3		8315000	8315000	0	((hour == 5'b0) && (min == 6'b0)) && (sec == 6'b0)			
Failure4		8320000	8320000	0	((hour == 5'b0) && (min == 6'b0)) && (sec == 6'b0)			
Failure5		8330000	8330000	0	((hour == 5'b0) && (min == 6'b0)) && (sec == 6'b0)			
Failure6		8335000	8335000	0	((hour == 5'b0) && (min == 6'b0)) && (sec == 6'b0)			
Failure7		8345000	8345000	0	((hour == 5'b0) && (min == 6'b0)) && (sec == 6'b0)			
Failure8		8350000	8350000	0	((hour == 5'b0) && (min == 6'b0)) && (sec == 6'b0)			
Failure9		8360000	8360000	0	((hour == 5'b0) && (min == 6'b0)) && (sec == 6'b0)			
Failure10		8365000	8365000	0	((hour == 5'b0) && (min == 6'b0)) && (sec == 6'b0)			
Success1		0	0	0				
Success2		5000	5000	0				
Success3		10000	10000	0				
Success4		15000	15000	0				
Success5		20000	20000	0				

The Assertion Pane is interconnected with other DVE windows and panes. To display an assertion in the Hierarchy and Variable panes, the Source view and the Wave view, double click an assertion instance in the Assertion Pane. [Figure 8-5](#) shows the panes.

Figure 8-5 Populated Panes



The following occurs:

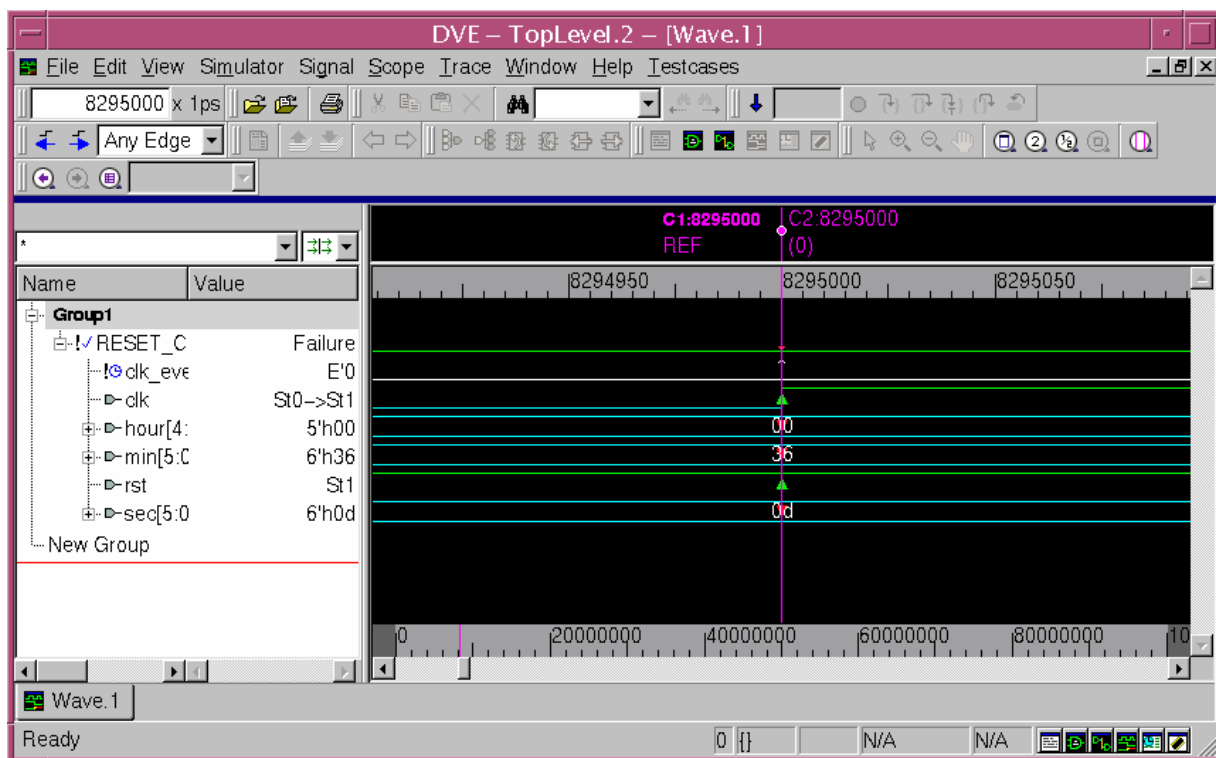
- The Hierarchy pane displays the associated unit or HDL scope that contains the assertion.
- The Variable pane displays the HDL variables corresponding to the unit or scope. This is not specific to the assertion (i.e., it may contain more signals that are used in the assertion).
- Up to three Source views may be displayed:
 1. HDL source
 2. Bind source
 3. OVA definition source
- The Wave view opens and displays the selected assertion centered on the failure. The cursors mark the start and end time of the selected assertion with the area between the cursors grayed.

A green circle indicates a signal value at a specific time that contributed to a successful sub-expression in the assertion. A red circle indicates a signal value at the time which caused a sub-expression to fail. A sub-expression failing may result in the overall assertion failing.

Viewing an Assertion in the Wave view

Figure 8-6 shows an assertion with no delta between the start and end time.

Figure 8-6 Assertion in the Wave view



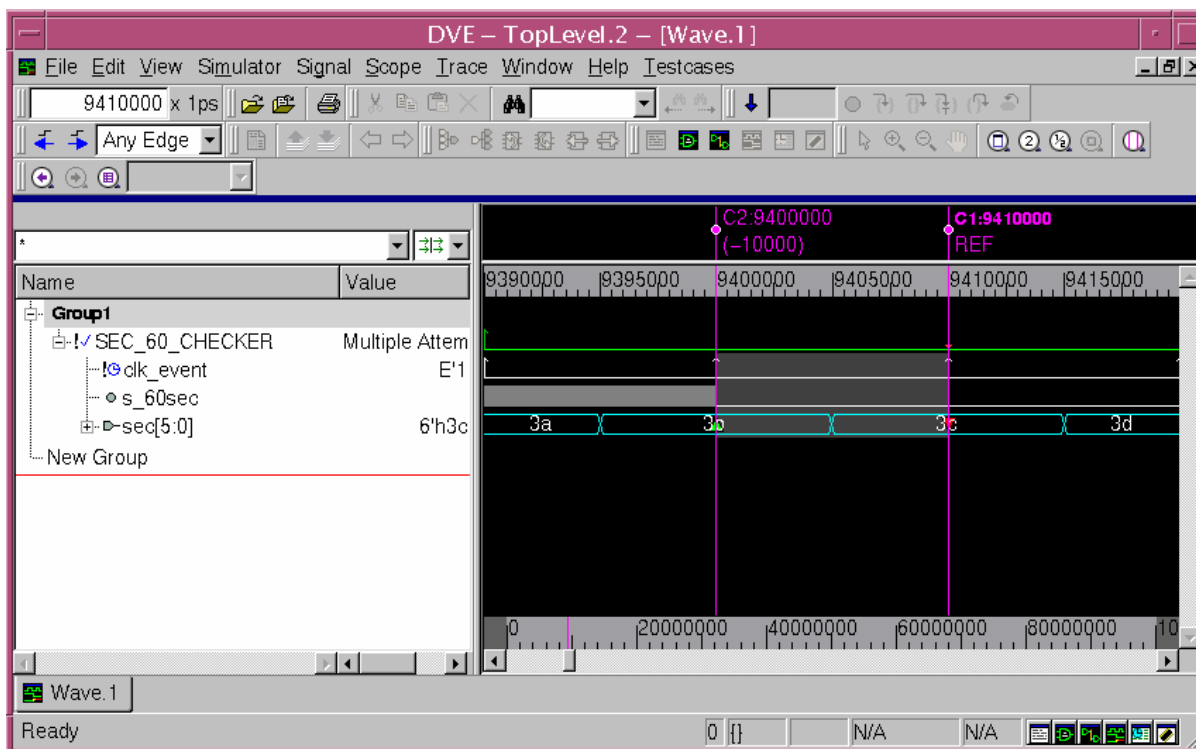
The following occurs in Figure 8-7:

- The C1 and C2 cursors are automatically placed at the start and end of the assertion. In this assertion, there is no time component (sequence), so C1 and C2 are at the same time.
- In Signal Group 1, the assertion `RESET_CHECKER` is listed first in the tree view. This is the assertion result signal. The waveform consists of red and green arrows. Green arrows indicate where the assertion was determined to be a success, and red arrows indicate where it failed; the red arrow illustrates the first failure.
- `RESET_CHECKER` is expanded into the following components:
 - The first component is `clk_event`. Each clock event shows you when the assertion fired and the clock ticks that happen for sequences.
 - The rest of the signals are the those that contributed to the success or failure. These signals are `clk`, `rst`, `hour`, `min`, and `sec`.
 - The green dots on the waveform indicate that the signal was OK at that clock tick. The red dots indicate that the signal contributed to the failure of the assertion at that clock tick.

Viewing an Assertion Failure Time Delta

When there is a delta for an assertion failure, the Wave view opens and displays the selected assertions centered on the failure. The cursors mark the start and end time of the selected assertion with the area between the cursors grayed. A green circle within a signal indicates a successful signal or value, while a red inverted circle and the C1 cursor indicates a failure.

Figure 8-7 Assertion Failure Time Delta in the Wave view



- When you hold the mouse cursor over a green or red circle, an InfoTip appears, displaying details on the impact of the signal.
 - If a green arrow exists, the InfoTip tells why this signal contributed to a success so far.
 - If red arrow exists, the InfoTip tells why this signal contributed to a failure.
- A white arrows indicates assertion clock events in the window.
- If you hold the mouse cursor over the attempt failure, an InfoTip appears, displaying details on the failure. For each attempt ending at this time, the InfoTip contains the following:
 - Start time

- End time
- Delta
- Instance
- Offending/Reason

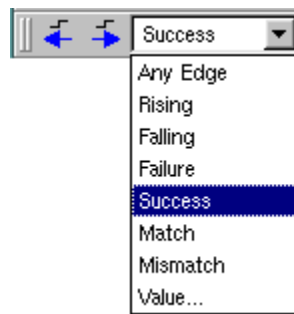
Navigating Assertions in the Wave view

Use the Search toolbar item to move to the next or previous assertion success or failure in the Wave view.

To navigate assertions in the Wave view

1. With an assertion selected, select **Success** or **Failure** from the Search pull-down menu (see [Figure 8-9](#)).

Figure 8-8 Navigate Assertion Successes and Failures



2. Click the back or forward arrow to move to the previous or next success or failure of the selected assertion.

Note:

The Match and Mismatch items in the Search pull-down menu are used for signal comparisons, not for assertion cover property match and mismatch display in the Wave view.

Local variables are inserted into the signal list in the hierarchy pane as shown by the local variable count.



Navigating Source Code

To display code related to an assertion attempt

1. Double-click an assertion attempt in either Assertion View tab (see [Figure 8-9](#)).

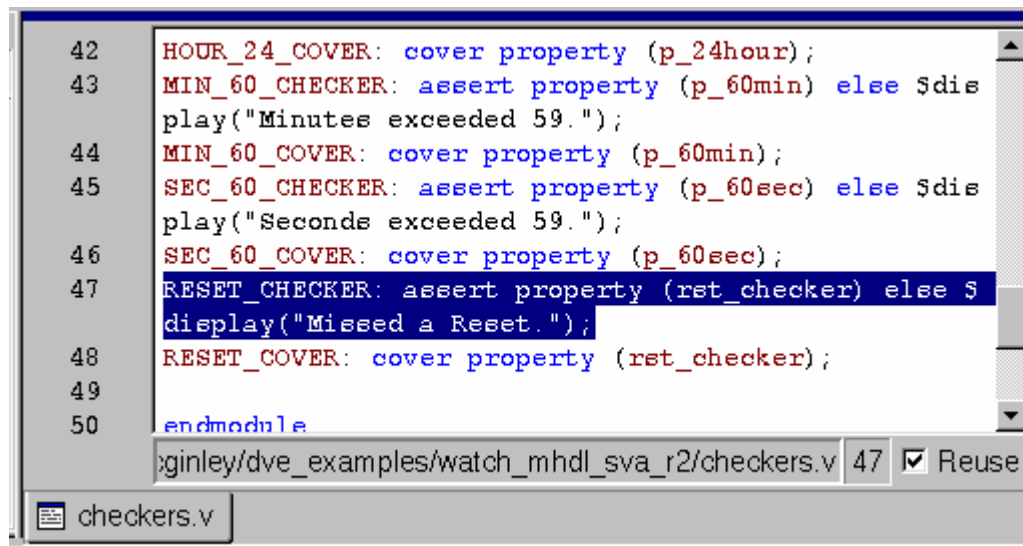
Figure 8-9 Select an Assertion Attempt

Name	Instance	Start	End	Delta	Reason
RESET_CHECKER	/TB_WATCH/CHECKERS	8295000	8295000	0	(((hour == 5'b0) && (min == 6'b0)) && (sec == 6'b1
Failure1		8295000	8295000	0	(((hour == 5'b0) && (min == 6'b0)) && (sec == 6'b1
Failure2		8295000	8295000	0	(((hour == 5'b0) && (min == 6'b0)) && (sec == 6'b1

Code is displayed as follows:

- The Source view displays the HDL code where the assertion is inlined or bound.
- A Source view displays the assertion code with the assertion highlighted (see [Figure 8-10](#)).

Figure 8-10 Assertion Source Code



```
42  HOUR_24_COVER: cover property (p_24hour);
43  MIN_60_CHECKER: assert property (p_60min) else $display("Minutes exceeded 59.");
44  MIN_60_COVER: cover property (p_60min);
45  SEC_60_CHECKER: assert property (p_60sec) else $display("Seconds exceeded 59.");
46  SEC_60_COVER: cover property (p_60sec);
47  RESET_CHECKER: assert property (rst_checker) else $display("Missed a Reset.");
48  RESET_COVER: cover property (rst_checker);
49
50  endmodule
```

pginley/dve_examples/watch_mhdl_sva_r2/checkers.v 47 ☒ Reuse

checkers.v

2. To edit the assertion in your default text editor, select **Edit > Edit Source**.

9

Tracing Drivers and Loads

This chapter describes how to trace drivers and loads of signals in your design. It includes the following topics:

- “Overview of Tracing Functionality”
- “Trace Drivers”
- “Trace Loads”

Overview of Tracing Functionality

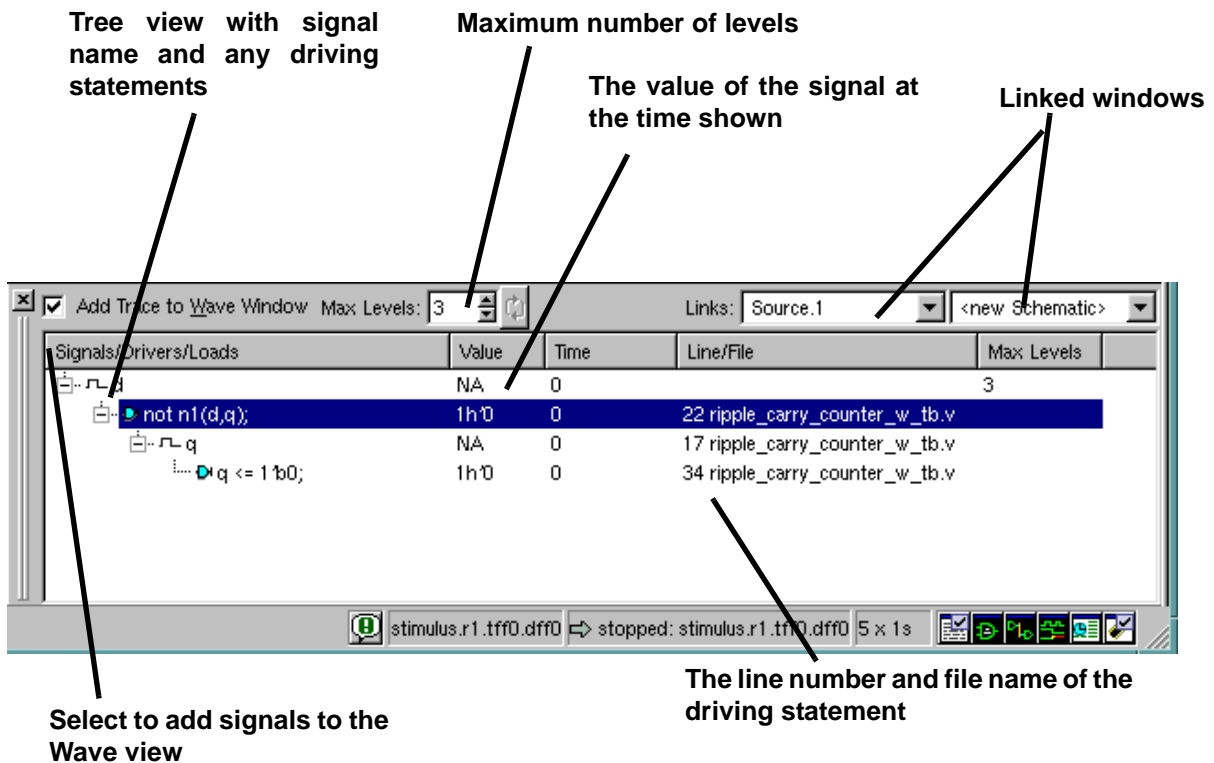
Use DVE to trace drivers and loads and follow connections that affect a signal value.

- The driver of a signal is the driver that contributed to the value of the signal at a given time.

- Displaying all drivers lists all the drivers that can possibly contribute to a signal value.
- A signal's load(s) are the input port(s), I/O port(s), and statements that read the signal's value.

Figure 9-1 shows the Drivers and Loads pane.

Figure 9-1 Drivers and Loads pane



For tracing drivers and loads, DVE includes the following functionality:

- Multiple driver panes are allowed as long as there are multiple top-level windows to contain them.
- Driver panes are independent from top level to top level.
- Driver panes can be deleted by clicking on the X icon.

- Driver panes can be undocked or docked.

Driver panes can be linked to, at most, one Source view in the same top-level frame and one Schematic view. The Links: combo boxes, at the top of the pane show the current linked windows. By default, the first open Source view is linked to the driver pane and no schematic (indicated by <new Schematic>).

Linking a Source and Schematic view means those windows can be wired up to track the selection. That means that selecting an object in the drivers pane will cause that object to be selected and shown in the linked Source or Schematic views. Checking the **Add to Waves** check box will add any signals shown in the driver pane to the Wave view.

Unchecking the **Add to Waves** check box does not delete anything from the Wave view, but prevents additional signals in the drivers pane from being added to the Wave view.

Supported Functionality

- All Verilog types, constructs, control path.
- Verilog gate and UDPs.
- VHDL but only down to the process statement. All drivers within a process are determined to be active.
- SystemVerilog datatypes.

Trace Drivers

To trace drivers

1. Select a signal in a DVE window or pane.

For example, Hierarchy pane, Data pane, Wave view, Source view, List view, or Wave view.

2. Click the drivers toolbar icon:



3. Select a signal in any pane, then select **Trace > Trace Drivers**.
4. In the Wave view, double-click on a waveform.

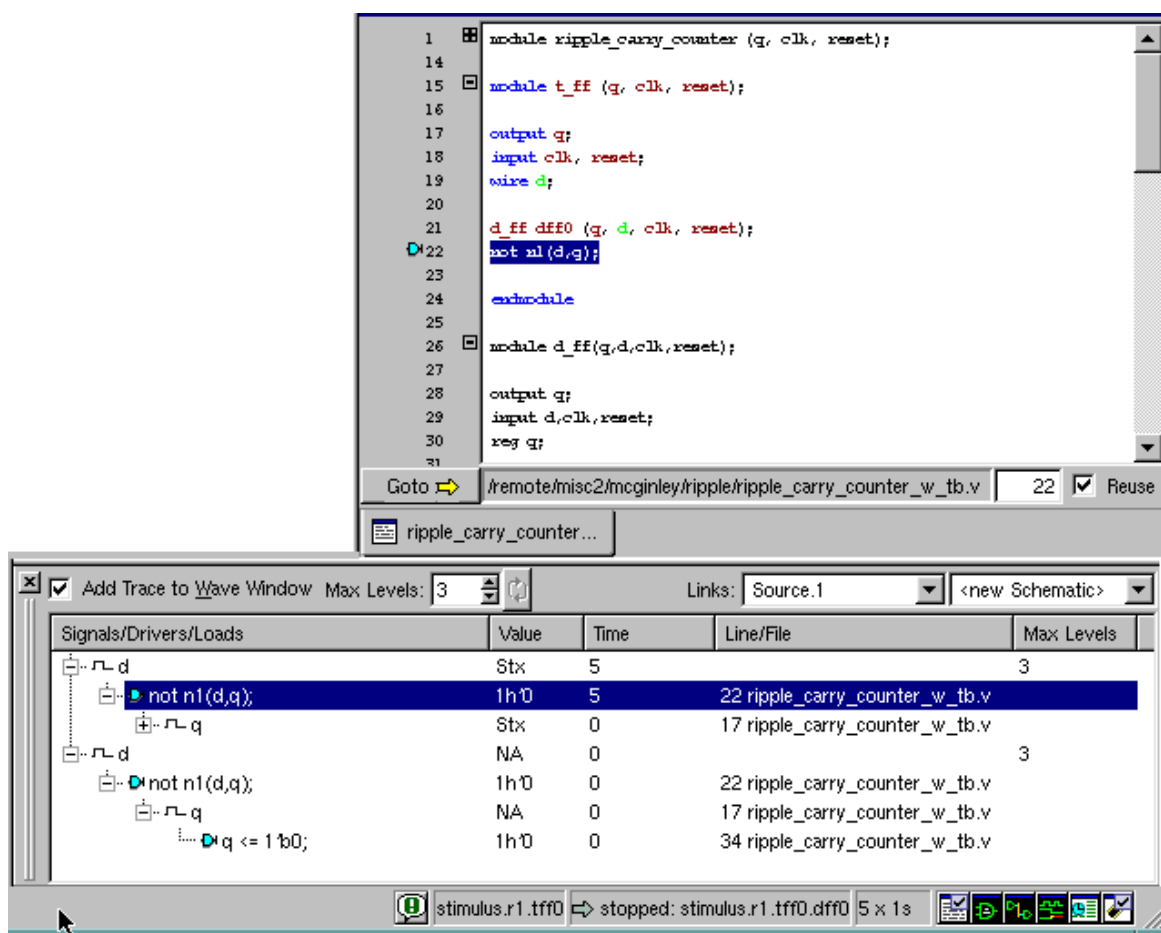
For example, on a transition from 0 -> 1 or 1 -> 0.

When a driver is traced, a new driver pane will be created if none exists in the current top level. If a driver pane exists, the driver information will be added to the top of the list. The first driver will be highlighted in the Source view and annotated with a blue node in the gutter, as shown in [Figure 9-2](#).

Note:

Only one driver pane is allowed per top-level frame.

Figure 9-2 Select a Transition in the Wave view to Display Driver



Trace Loads

To trace loads

1. Select a signal in a DVE window or pane.

For example, the Hierarchy pane, Data Pane, Wave view, Source view, List view, or Wave view.

2. Click the Trace Loads toolbar icon:



When a load is traced, a new driver pane will be created if none exists in the current top-level. If a driver pane exists, the load information is added to the top of the list. The first load will be highlighted in the Source view and annotated with a blue node in the gutter

Note:

Only one driver pane is allowed per top-level frame.

10

DVE Testbench Debugger

This chapter describes the DVE (Discovery Visual Environment) Testbench Debugger. It includes the following topics:

- [“Overview”](#)
- [“Enabling Testbench for Debugging”](#)
- [“Using the Testbench Debugger GUI”](#)
- [“Understanding the Testbench Debugger GUI”](#)

Overview

The DVE integrated testbench graphical debugger provides a common interface for debugging HDL and Testbench code simultaneously and is seamlessly integrated with the current DVE HDL debug windows.

In interactive mode, the Testbench Debugger provides you with visibility into the testbench-related dynamic constructs and their values during simulation. This is done by using the proven visualization of the Testbench GUI's stack pane, local pane, and watch pane combined with DVE's Source view and its intuitive look and feel.

Using the salient features of the new testbench debugger, you can analyze, understand, and debug the behavior of your complicated verification environment faster. You will be able to perform a comprehensive analysis using the seamless design and verification environment.

The testbench debugging interface enables you to perform the following:

- Navigate HDL or Testbench source code in a single DVE Source view
- View HDL and Testbench scopes in DVE's Hierarchy pane
- Analyze HDL and TB signals together in the Watch pane
- Run HDL and Testbench-related UCLI commands all from a single application

Enabling Testbench for Debugging

To enable the debugging capabilities for the testbench, you must specify the `-debug_all` switch along with your compilation command.

Note:

If you separately compile your design and testbench (NTB-OV separate compile flow), ensure that you use the `-debug_all` switch when compiling both your design and the testbench.

Using the Testbench Debugger GUI

You can start the Testbench Debugger from the command line and then perform your simulation run from the user interface.

- From the command prompt, enter the following:

```
%> simv -gui
```

In this example, `simv` is the executable.

Note:

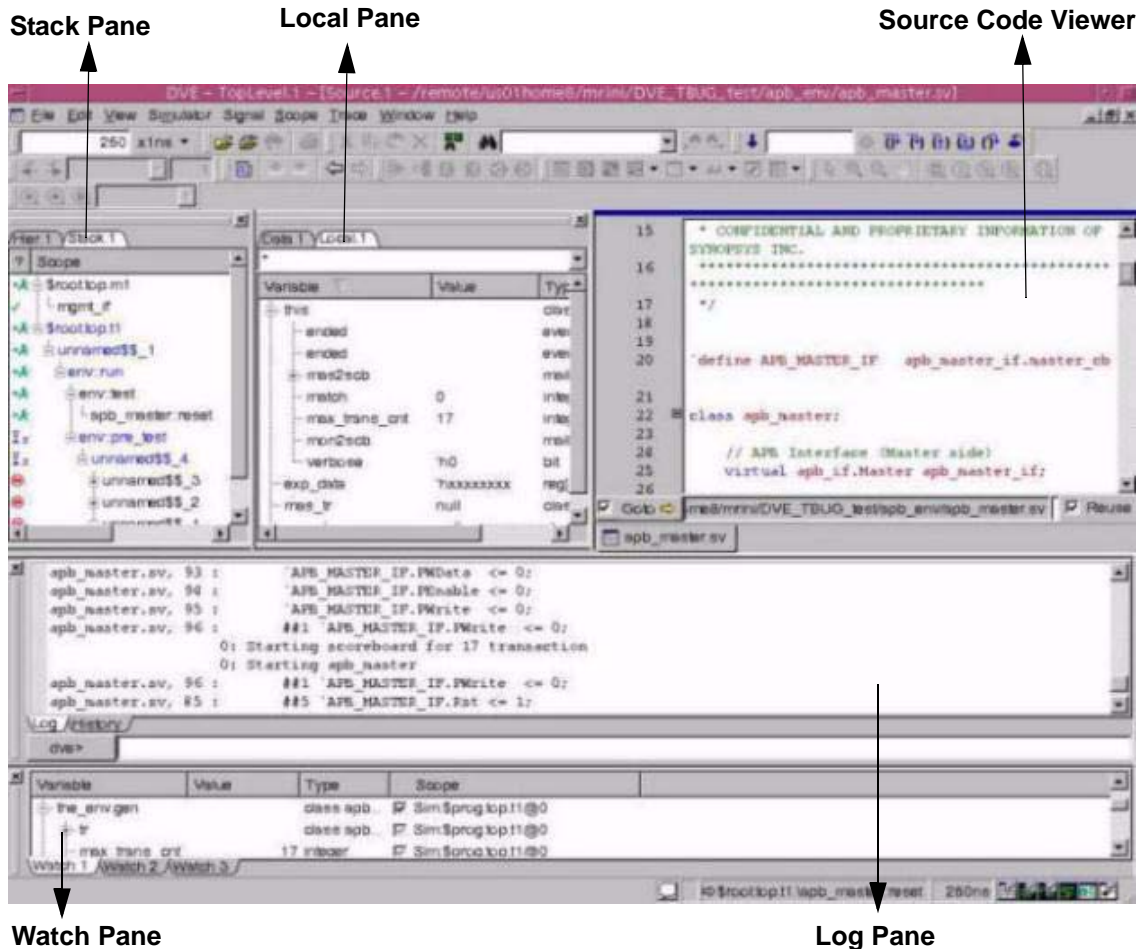
If you use the `-debug_all` switch when compiling your design that contains testbench code, the DVE automatically provides debugging options for your testbench program. However, you can disable these settings in the Preferences dialog box by clearing the option "Enable testbench debugging for interactive design" in the Edit menu. Select a transition in the Wave view to display the driver.

Understanding the Testbench Debugger GUI

This section describes the DVE Testbench Debugger GUI.

Testbench Debugger Panes

The top-level window of DVE contains three additional panes. The three additional panes are – Stack pane, Local pane, and Watch pane as illustrated in the following diagram:



Note:

The Watch pane appears only when you add variables or signals to it for monitoring purposes.

- Stack Pane - Displays the testbench dynamic hierarchy tree along with all the testbench threads and their status. This pane is highlighted when the Local tab is selected.
- Local Pane - Displays all the testbench variables and dynamic objects with their current values based on the currently selected scope within the call stack. The testbench variables and dynamic objects will change when you select different testbench scopes in the Stack.
- Watch Pane - Enables you to monitor the status of your variables during simulation.

Stack Pane

This pane shares a tabbed view with the hierarchy pane. Select the Stack tab to display the Stack pane and view the status of various threads. This view is cross-linked with the Source and Local panes. Double-clicking on objects in this pane synchronizes the display in the source and local panes. The hierarchy tab displays only the static objects in your design, whereas the Stack tab displays the dynamic threads created during runtime.

Note:

The Stack pane appears empty when you invoke the Testbench Debugger at time 0. The dynamic objects are displayed as and when they are created in the testbench during simulation.

The following figure illustrates the Stack tab and the Hierarchy tab:

Hier.1		Stack.1	
?	Scope	File : Line	Thread
✓	mgmt_if	mem.v : 58	8
→✖	\$root.top.t1	test_00_debug.sv : 19	0
→✖	unnamed\$\$_1	test_00_debug.sv : 31	4
→✖	env::test	env.sv : 90	4
→✖	apb_master::reset	apb_master.sv : 85	4
⏸z	env::pre_test	env.sv : 78	4
⏸z	unnamed\$\$_4	env.sv : 81	4
✓	unnamed\$\$_3	env.sv : 84	7
⊖	unnamed\$\$_2	env.sv : 83	6
⊖	unnamed\$\$_1	env.sv : 82	5

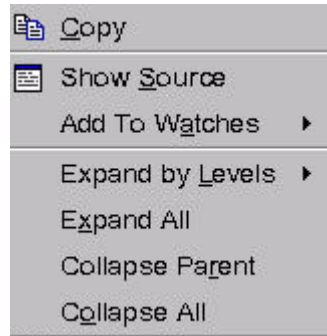
The above illustration shows the current active threads and their status. The status of a thread could be – Running, Stopped, or Suspended. The following table illustrates the conventions denoted by these icons:

→✖	Thread is running.
⊖	Thread is stopped.
⏸z	Thread is suspended.

The thread column displays the unique id of the thread. It can be the same if function calls in the stack belong to the same thread.

Using the Stack Pane Context Menu

The Stack pane context menu provides various options. You can quickly access and start using these options through the context menu. To invoke the context menu, right-click from the Stack pane. The following menu options appear:

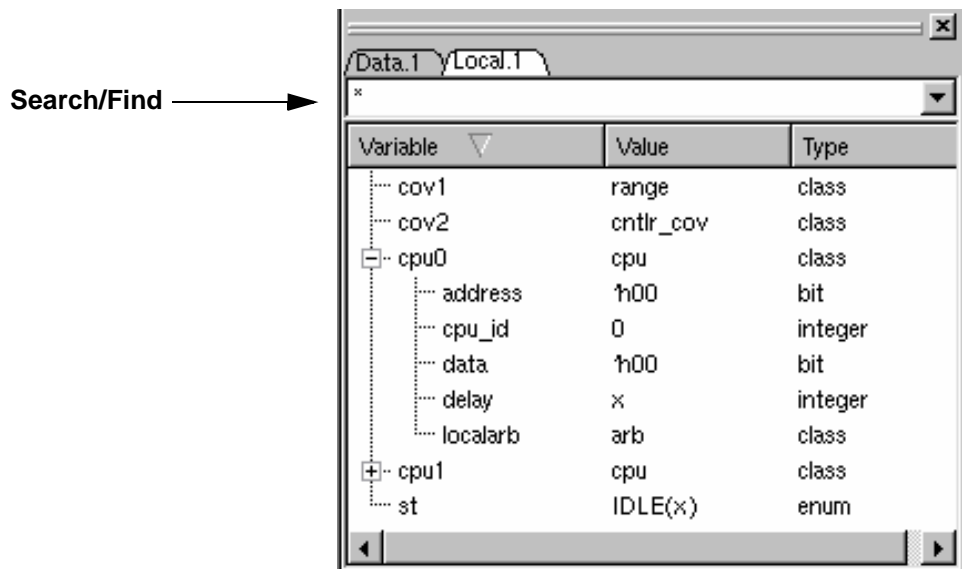


The following table explains the menu options:

Option	Description
Copy	Takes copy of the object.
Show Source	Displays source code of your testbench program.
Add To Watches	Adds signals to monitor in the Watch pane.
Expand All	Expands the tree.
Collapse All	Collapses the tree.
Select All	Takes copy of all the objects.

Local Pane

The local pane shares a tabbed view with the 'Data' tab. The local pane displays variables in a selected scope in the stack pane. This view is tied to the stack pane and the default view shows variables of the current active thread. This pane also has a Filter feature that you can use to search or find variables.



Note:

The Local pane displays the variables when you select an object in the Stack pane.

Watch Pane

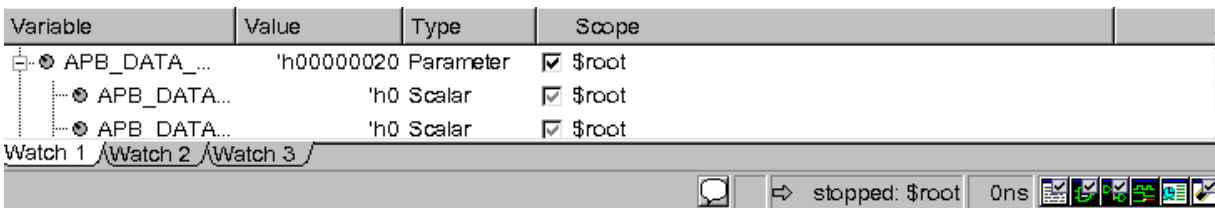
Occasionally, you may need to monitor the status of testbench and HDL variables throughout the simulation regardless of the active thread. You can select all the variables and objects to watch their behavior in this pane. The Watch pane displays the selected item, its value and the type for tracking, regardless of the active context.

By default, the Watch pane contains three tabs labeled Watch 1, Watch 2, and Watch 3. You can add as many tabs as you desire. Use the Watch panes to monitor values of variables regardless of the current context. You can add variables from the Source or Local window or by performing a drag-and-drop.

To add a Watch tab, go to the menu **View > Watch > Add New Page**. You can also delete the watch tabs.

For more information about adding signals to the Watch pane, see [“Signal Menu”](#).

The following figure illustrates a typical Watch pane:



This figure illustrates the variable, its value, type, and the scope. Using the check box in the Scope column, you can tie the variable to a given thread throughout simulation or tie the variable to the currently selected thread in the call stack. This feature is available for all object types, including the design signals.

For example, add a variable called 'x' in the Watch pane and select the check box to tie it to a given thread. This variable is displayed throughout the simulation from the same dynamic instance of the scope (active thread), irrespective of the thread being alive or not. By default, this check box is selected.

Clearing the check box evaluates the variable in the currently active thread in the call stack. For example, add a variable 'x' from the active thread, 'main', during the beginning of simulation. Assume the

active thread changes to some other thread at a later point of time. The variable 'x' in the Watch pane now refers to the same variable in the dynamic instance of the scope (active thread), but not from the active thread, 'main'.

Testbench Debugger Menu Options

A few menu options specific to debugging your testbench programs are available. These options are added to different menus in the main DVE GUI such as View menu, Simulator menu, and the Window menu.

The following section explains the new options in different menus.

View Menu

The View menu allows you to customize your viewing options in the Watch pane. It has a new menu option, Watch. Click **View > Watch** and select the relevant option, as described in the following table:

Option	Allows you to...
Add New Page	Add a new tab to your Watch pane.
Delete Current Page	Delete the Watch tab from the Watch pane.
Rename Current Page	Rename your Watch tab.
Edit Variable	Cut, copy, and paste the variable names in Watch tab.

Signal Menu

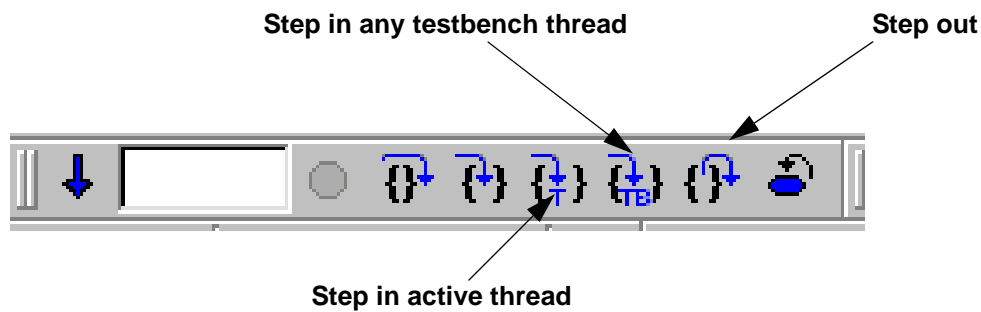
This menu allows you to add signals to the Watch tab. To add a signal for monitoring, select **Signal > Add to Watches > Watch**.

Simulator Menu

After you start simulation, you can use menu commands to run and control the simulation. Use the following commands to control the simulation:

Command	Description
Step In Testbench	Stops at the next executable line in the testbench for Native Testbench (NTB), OpenVera, and SystemVerilog testbenches.
Step Out	Steps to the next executable line outside of the current function or task.
Step in ActiveThread	Stops at the next executable line in the current active thread.

Alternatively, you can use all of the simulator menu options from the toolbar by clicking on the following icons in the interface:



Note:

Move the cursor over the icon to display the ToolTip.

Window Menu

This menu allows you to select or clear the following panes to debug your testbench programs. To enable the panes, select **Window > Pane** and then select the relevant option, as described in the following table:

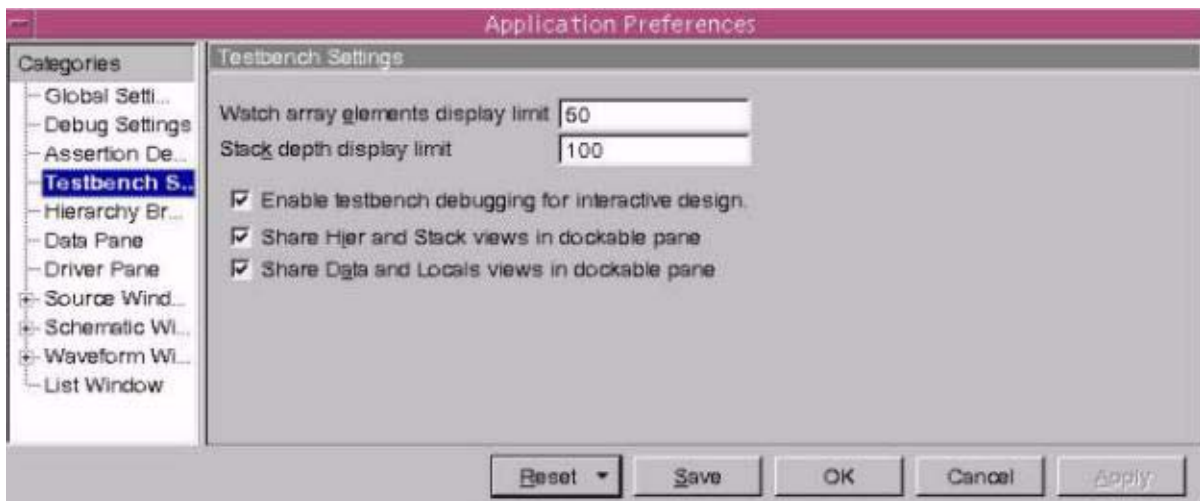
Option	Allows you to add a...
Stack	Stack pane to your DVE top-level window.
Local	Local pane to your DVE top-level window.
Watch	Watch pane to your DVE top-level window.

Setting Testbench Debugger GUI Preferences

You can set your preferences to customize the interactive Testbench debugger using the Application Preferences dialog.

To set the testbench debugger GUI preferences

1. Select **Edit > Preferences**, then select Testbench Settings to display the Testbench Preferences screen as follows:



2. Set the following preferences, as appropriate:
 - Watch array elements display limit - Sets the maximum number of elements that the GUI can display for a large array when it is expanded in the Local pane and Watch pane.
 - Stack depth display limit - Sets the maximum levels of stacks for Cbug.
3. Use the check boxes to perform the following:
 - Enable testbench debugging for interactive design - By default, this option is enabled. Clear this check box to disable the Testbench Debugger panes in the DVE main window. This will disable "step in testbench" and the programs will not be dumped and shown in the Hierarchy pane. However, you can still open the testbench source files manually and set line breakpoints, but you cannot see the testbench signals.
 - Share the hierarchy and stack views in a dockable pane - By default, this option is enabled to make the Stack pane available to you in a dockable pane. Clearing this check box opens the Stack pane in a separate window.
 - Share delta and local views in a dockable pane - By default, this option is enabled to make the Local pane available to you in a dockable pane. Clearing this check box opens the Local pane in a separate window.

11

Using the C, C++, and SystemC Debugger

This chapter describes debugging VCS and VCS MX designs that include C, C++, and SystemC modules with DVE. It contains the following sections.

- “Getting Started”
- “Commands Supported by the C Debugger”
- “Common Design Hierarchy”
- “Interaction with the Simulator”
- “Configuring CBug”
- “Supported platforms”
- “Example: A Simple Timer”

Getting Started

This section describes how to get started using DVE to debug designs that include C, C++, and SystemC modules.

Using a Specific gdb Version

Debugging of C, C++ and SystemC source files relies upon a gdb installation with specific patches. This gdb is shipped as part of the VCS image and is used per default when CBug is attached. No manual setup nor installation of gdb is needed.

Attaching the C-Source Debugger in DVE

You can debug designs containing C-source modules with or without the C debugger running. However, you must attach the C-source debugger to view and debug C-source code within the design.

Note that the `-debug_all` flag enables line breakpoints for the HDL (Verilog, VHDL) parts only but not for C files as well. C files must be compiled with the `"-g"` C compiler option. Do this as follows:

- When invoking the C/C++ compiler directly:

```
gcc ... -g ...  
g++ ... -g ...
```

- When invoking one of the VCS tools:

```
vcs      ... -cflags -g ...  
syscan   ... -cflags -g ...  
syscsim  ... -cflags -g ...
```

The following steps describe attaching the C-source debugger to run DVE to debug VCS or VCS MX (Verilog, VHDL, and mixed) simulations containing C, C++, and SystemC source code.

1. Compile your VCS or VCS MX with C, C++, or SystemC modules as you normally would, making sure to compile all C files you want to debug in DVE.


For example, with a design with Verilog on top of a C or C++ module:

```
gcc -g [options] -c my_pli_code.c
vcs +vc -debug_all -P my_pli_code.tab my_pli_code.o
```

Or with a design with Verilog on top of a SystemC model:

```
syscan -cpp g++ -cflags "-g" my_module.cpp:my_module
vcs -cpp g++ -sysc -debug_all top.v
```

Note that you must use `-debug` or `-debug_all` to enable debugging.

2. Open DVE.
3. Click  to start the simulation.
4. Select **Simulator > C/C++ Debugging**.

Or

Enter `cbug` on the console command line.

Debugging of C, C++, and SystemC source code is enabled and you see the following message in the console History tab:

```
CBug - Copyright Synopsys Inc. 2003-2006.
```

Note:

The C-source debugger will automatically attach when you set a breakpoint in a C / C++ file (extensions .c, .cc, .cpp, and .h are recognized).

Detaching the C-source Debugger

You can detach and reattach the C-source debugger at any time during your session.

To detach the C-source debugger:

Toggle the debugger off by selecting **Simulator > C/C++ Debugging**.

Or

Enter

`cbug -detach`

on the console command line.

Displaying C-source Files in the Source Pane

There are three ways to display a C source file in the Source Pane:

- Automatically when the simulation stops in a given C file due to a breakpoint or cross-step
- Explicitly through **File > Open**.

Note:

Select **C/C++ files** as File type in the Open Source File dialog box.

Double-click a SystemC process in the design hierarchy. When you double-click a process of an SystemC module, DVE automatically opens the source file if the file can be found and was compiled with `-g`; otherwise you are prompted to select the proceed.

Commands Supported by the C Debugger

These commands are supported by the C debugger:

- `continue`
- `run`
- `next`
- `next -end`
- `step`
- `get variable_name` (Returns the variable value)
- `finish`
- `stack`
- `dump` (of SystemC objects)
- `cbug`

These commands are not supported:

- `save`
- `restore`
- `release` (applied to C or SystemC signals)

- `drivers` (applied to C or SystemC signals)
- `loads` (applied to C or SystemC signals)

Note that save/restore is supported for Verilog and VHDL but not in C code models. The internal state of any user-written C, C++, and SystemC model is not saved/restored, meaning that the save/restore feature does not work when C code is involved in the simulation.

Note:

This section uses the full UCLI command names. If you are using a command alias file such as the Synopsys-supplied alias file, enter the alias on the UCLI command line. See the *UCLI User Guide* for more information.

`scope` **Command**

The scope command is supported for SystemC instances.

`show` **Command**

`show [-instances|-signals|-ports]` is supported for SystemC instances, for example "show -ports top.inst1". Any other type such as -scopes, -variables, -virtual is not supported for SystemC instances. A radix is ignored.

`change` **Command**

The change command is supported within these two strict limitations:

- Only variables that are visible in the current scope of the C function (e.g. local variables, global variables, class members.) can be changed. Hierarchical path names like top.inst1.myport are not supported.

- The type must be a simple ANSI type like `int`, `char`, `bool`. Changing SystemC bit-vector types like `sc_int<>` or user-defined types is not supported. Any attempt to set an unsupported datatype will trigger the error message "Unsupported type for setting variable".

`force` **Command**

Command '`force -deposit <var> <expr>`' is supported. It has the same functionality and restrictions as command '`change`' described above.

Other options of the `force` command like multiple values, timing information or `-repeat`, `-cancel`, `-drive` options are not supported.

`stack` **Command**

When you are stopped in C code, then you can see the stack list. Each entry of the list tells the source file, line number, function name. The function where you are stopped right now appears at the top of the list. If the source code for a given function has been compiled without compiler flag `-g`, then the file/line number information is not available. CBug selects `without-g.txt` in this case.

Command `stack -up|-down` move the active scope up or down. The source file corresponding to the active scope is shown and `get` command applies to this scope.

Accessing C/C++/SystemC Elements with the `get` Command

When stopped at a C source location, certain elements are visible and can be queried with the `ucli::get` command:

- Function arguments
- Global variables

- Local variables
- Class members (if the current scope is a method)
- Ports, `sc_signal` and plain members of SystemC modules anywhere within the combined HDL+SystemC instance hierarchy.
- Arbitrary expression including function calls, pointers, array indices etc. Please note that some characters such as '[']' need to be enclosed by '{ }' or escaped with '\ ' otherwise Tcl will interpret them.

Examples

- `ucli::get myint`
- `ucli::get this->m_counters`
- `ucli::get {this->m_counters[2]}`
- `ucli::get strlen(this->name)`

The `<name>` given with a `synopsys::get <name>` argument refers to the scope in the C source where the simulation stopped (the active scope). This is important to keep in mind because C source may have multiple objects with the same name but in different scopes. Which one is visible depends on the active scope. This means that `<name>` may not be accessible anymore once you step out of a C/C++ function.

Accessing SystemC Elements with the get Command through an hierarchical Path Name

The argument of `synopsys::get` may refer to an instance within the combined HDL/SystemC instance hierarchy. All ports, `sc_signals` and also all plain member variables of an SystemC instance can be

accessed with `synopsys::get` at any time. Access is possible independent of where the simulation is currently stopped, even if it is stopped in a different C/C++ source file or not in C/C++ at all.

Example

Assume this instance hierarchy

```
top           (Verilog)
  middle      (Verilog)
    bottom0   (SystemC)
```

whereby "bottom0" is an instance of this SC module:

```
SC_MODULE(Bottom) {
    sc_in<int> I;           // SC port
    sc_signal<sc_logic> S;  // SC signal
    int PM1;               // "plain" member variable, ANSI type
    str PM2;               // "plain" member variable,
user-def type
};
struct str {
    int a;
    char* b;
};
```

These accesses are possible:

```
synopsys::get top.middle.bottom0.I
synopsys::get top.middle.bottom0.S
synopsys::get top.middle.bottom0.PM1
synopsys::get top.middle.bottom0.PM2
synopsys::get top.middle.bottom0.PM2.a
```

Access is possible at any point in time, independent of where the simulation stopped. Note that this is a difference to accessing local variable of C/C++ functions. They can only be accessed if the simulation is stopped within that function.

Please also note that accessing plain member variables of SystemC instances is only possible with `synopsys::get` but not with `synopsys::dump`.

Format / Radix:

The C debugger will ignore any implicitly or explicitly specified radix. The format of the value returned is exactly as it is given by gdb (only SystemC data types are specially dealt with). Besides integers, you can also query the value of pointers, strings, structures, or any other object that gdb can query.

SystemC Datatypes:

The C debugger offers specific support for SystemC datatypes, for example, an `sc_signal<sc_bv<8>>`. When you do a `print` of such a value, gdb usually returns the value of the underlying SystemC data structure that is used to implement the data type. This is normally by no means what you want to see and is generally useless. The C debugger recognizes certain native SystemC data types and prints the value in an intuitive format. For example, it will print the value of the vector in binary format for an `sc_signal<sc_bv<8>>`.

The following native SystemC types are recognized.

Templatized channel types `C<T1>`:

```
C := { sc_in_clk, sc_in, sc_inout, sc_out, sc_signal,
ccss_param }
T1 := { bool, [[un]signed] char, [unsigned][long|short] int,
        [[long] double] float, sc_logic, sc_lv, sc_bit, sc_bv,
        sc_[u]int, sc_int_base, sc_big[u]int, sc_[un]signed,
        sc_fxval[_fast], sc_[u]fix[ed][_fast], sc_string,
        char*, void*, struct X* }
```

When the value of an object O of such a type C is to be printed, then the C debugger prints the value of O.read() rather than O itself.

Native SystemC data types:

```
T2 := { sc_logic, sc_lv, sc_bit, sc_bv,  
        sc_[u]int, sc_int_base, sc_big[u]int, sc_[un]signed,  
        sc_fxval[_fast], sc_[u]fix[ed][_fast], sc_string }
```

The C debugger prints values of these data types in an intuitive format. Decimal format is taken for `sc_[u]int`, `sc_int_base`, `sc_big[u]int`, `sc_[un]signed`, binary format for `sc_logic`, `sc_lv`, `sc_bit`, `sc_bv`.

Example

SystemC source code:

```
sc_in <int> A  
sc_out<sc_bv<8>>B;  
sc_signal <void*>;  
int D;  
synopsys::get A  
17  
synopsys::getB  
01100001  
synopsys::getC  
0x123abc  
synopsys::getD  
12
```

Using Breakpoints

You can set line breakpoints on C / C++ / SystemC source files using the Source view, the Breakpoints dialog box, or the command line. Breakpoints in C-source code support line breakpoints.

Set a Breakpoint from the Breakpoints Dialog Box

You can set a line, time, or signal breakpoint using the Breakpoints dialog box. See the section [“Managing Breakpoints from the Dialog Box” on page 4-15](#) for more information.

Control Line Breakpoints in the Source view

You can control line breakpoints in the Source pane in two ways:

- Clicking on the circular breakpoint indicator in the line attribute area.
- Selecting a line breakpoint, right-clicking from the attribute area, then selecting a context-sensitive menu command.

For more information, see the section [“Control Line Breakpoints in the Source view” on page 4-13](#).

Set a Breakpoint from the Command Line

To create a line breakpoint from the command line::

Enter the stop command into the console command line using the following syntax:

```
stop -file filename -line linenumber
```

For example:

```
stop -file B.c -line 10
stop -file module.cpp -line 101
stop -in my_c_func
stop -in timer::clock_action()
```

Instance Specific Breakpoints

Instance specific breakpoints are supported with respect to SystemC instances only. Specifying no instance or instance name "-all" means to always stop, no matter what the current scope is,

If the debugger reaches a line in C, C++, SystemC source code for which a instance-specific breakpoint has been set, then it will stop only if the following two conditions are met:

- The corresponding function was called directly or indirectly from a SystemC SC_METHOD, SC_THREAD or SC_CTHREAD process.
- The name of the SystemC instance to which the SystemC process belongs matches the instance name of the breakpoint.

Please note that C functions called through the DPI, PLI, DirectC or VhPI interface will never stop in an instance-specific breakpoint because there is no corresponding SystemC process.

Please also note that you must use the name of the Systemc module instance and not the name of the SystemC process itself.

Breakpoints in Functions

You can also define a breakpoint by its C/C++ function name with the

```
stop -in function
```

command.

Examples

```
stop -in my_c_function  
stop -in stimuli::clock_action()
```

Restriction

If multiple active breakpoints are set in the same line of a C, C++ or SystemC source code file, then the simulation will stop only once.

Deleting a Line Breakpoint

To delete a line breakpoint, do either of the following:

- Click the red button in the Source view to disable an enabled breakpoint.
- Select **View > Breakpoints**, select the breakpoint to delete, then click **Delete**.
- On the console command line, enter

```
stop -delete <index>
```

Then press **Enter**.

Stepping Through C-source Code

Stepping within, into, and out of C sources during simulation is accomplished using the `step` and `next` commands. Extra arguments to either `step` or `next` such as `-lang` or `-thread` are not supported for C code. **ONLY `next -end` IS ALLOWED.**

Stepping within C Sources:

You can step over a function call with `next` or step into a function with `step`.

Note:

Stepping into a function that was not compiled with `-g` is generally supported by `gdb` and also the C debugger. However, in some cases `gdb` becomes confused on where to stop next and may proceed further than anticipated. In such cases, we recommend to set a breakpoint on a C source that should be reached soon after the called function finishes and then issue command `synopsys::continue`.

Use the `stack -up` command to open the source code location where you want to stop, set a breakpoint and then continue.

Cross-stepping between HDL and C Code

Cross-stepping is supported in many but not all cases where C code is invoked from Verilog or VHDL code. These case are supported:

- From Verilog caller into a PLI C function: note that this is only supported for the "call" function but not that "misc" or "check" function and also only if the PLI function was statically registered.
- From the PLI C function back into the Verilog caller.
- From Verilog caller into DirectC function and also back to Verilog
- From VHDL caller into an VhPI "foreign" C function that mimics a VHDL function and also back to VHDL: please note that the cross-step is not supported on the very first occasion when the C function is executed. Cross-stepping is possible for 2nd, 3rd and any later call of that function.

- From Verilog caller into an export "DPI" C function and also back to Verilog
- At the end of a Verilog export "DPI" task or function back into the calling C function. Note that this cross step HDL->C is only possible if the Verilog code was reached via a cross-step from C->HDL in the first place.

All cross-stepping is only possible if the C code has been compiled with debug information (gcc -g).

Cross-stepping in and out of Verilog PLI Functions

When you steps through HDL code and come to a call of a user-provided C function such as a PLI function like `$myprintf`, then the `next` command will step over this function. But the `step` command will step into the C source code of this function. Consequent `step/next` commands walk through the C function and finally you returns to the HDL source. Seamless stepping HDL->C->HDL is thus possible. This feature is called cross-stepping.

Cross-stepping is supported only for function that meet this criteria:

- PLI function
- Statically registered through a tab file
- The `call` call only (but not `misc` or `check`)

Cross-stepping into other Verilog PLI functions is not supported. However, an explicit breakpoint can be set into these function which will achieve the same effect.

Cross-Stepping in and out of VhPI Functions

Cross-stepping from VHDL code into a C function that is mapped through the VhPI interface to a VHDL function is supported with certain restrictions: cross-step in is not possible on the very first occasion when the C function is executed. Only later calls are supported. A cross-step out of C back into VHDL code is always supported.

Cross-stepping is not supported for C code mapped through the VhPI interface onto a VHDL *entity*.

Cross-Stepping in and out of DirectC Functions

Cross-stepping from Verilog into a DirectC function is supported, also cross-step back out. There are no restrictions.

Cross-Stepping in and out of DPI Code

Cross-stepping between [System]Verilog and import/export DPI functions is supported with a few restrictions:

- Cross-step from Verilog into an import DPI function is always supported.
- Cross-step from an import DPI function back into the calling Verilog source code is supported only if this DPI function was entered with a cross-step in the first place. That means doing continuously step commands will lead from the Verilog caller, into and through the import DPI function and back to the Verilog caller. statement into the import DPI function, through that function and finally back into the calling Verilog statement.

However, if the DPI function was entered through a run command and the simulation stopped in the import C function due to a breakpoint, then the cross-step out of the import DPI function into the calling Verilog statement is *not* supported. The simulation will advance until the next breakpoint is reached.

- Cross-step from C code into an export Verilog task or function is always supported.
- Cross-step from an export DPI task/function back into the calling C source code is supported only if this DPI task/function was entered with a cross-step in the first place. That means doing continuously step commands will lead from the C caller, into and through the import DPI task/function and back to the C caller.

However, if the export DPI task/function was entered through a run command and the simulation stopped in the export task/function due to a breakpoint, then the cross-step out of the export DPI function into the calling C statement is *not* supported. The simulation will advance until the next breakpoint is reached.

Cross-stepping from C into HDL:

Stepping from C code (that is called as a PLI/... function) into HDL code is generally supported. There are two ways to do this.

- If the C function was reached by previously cross-stepping from HDL into C, then CBug is able to automatically transfer control back to the HDL side once you step out of the C function. In this case, just type `step` or `next` in C code.

- In all other cases, CBug is not able to detect that the C domain is exited and needs an explicit command to transfer control back to the HDL side. When you do a `step` or `next` command that leaves the last statement of a C function called from HDL, then the simulation will stop in a location that belongs to the simulator kernel. There will be usually no source line information available since the simulator kernel is generally not compiled with `-g`, so you will not see a specific line/file information. Instead, `file without-g.txt` will be displayed.

If this happens, you can proceed as follows:

```
synopsys::continue or run
```

or

```
next -end
```

The `continue` will bring you to the next breakpoint which could be in either HDL or C source code. The `next -end` command will stop as soon as possible in the next HDL statement or the next breakpoint in C code, whichever comes first. Again, use commands `synopsys::continue` or `synopsys::next -end` to proceed.

Cross-Stepping in and out of SystemC Processes

CBug offers specific support for stepping between SystemC or HDL processes:

- When you leave a function that defines a `SC_METHOD` process, then a 'step' or 'next' will automatically stop in the next SystemC or HDL process, whatever comes next.

- Similarly, when you 'step' or 'next' over a 'wait' statement that belongs to an SC_THREAD process, then you will stop in the next SystemC or HDL process, whatever comes next.
- Doing a 'step' or 'next' in Verilog or VHDL will automatically stop in a SystemC process if that process happens to be next SystemC or HDL process to be executed.

That means doing 'step' or 'next' repeatedly will follow to flow of SystemC and HDL processes in the exact order in which the simulator executes them.

Direct gdb Commands

You can send certain commands directly to the underlying gdb through UCLI command `cbug::gdb`. The command will be executed right away and the UCLI command will return the response from gdb.

The command is

```
cbug::gdb gdb-cmd
```

gdb-cmd is an arbitrary command accepted by gdb including an arbitrary number of arguments, for example `info sources`. Doing `cbug::gdb` will automatically attach CBug, send `<gdb-cmd>` to gdb and return the response from gdb as the return result of the Tcl routine. The result may have one or multiple lines.

The routine returns successfully in most cases, even if gdb itself gives an error response. The routine gives an Tcl error response only when *gdb-cmd* has the wrong format, for example when it is empty.

Only a small subset of gdb commands are always allowed. These are commands that for sure do not change the state of gdb or simv, e.g. commands show, info, disassemble, x, etc. Other command make cbug::gdb return with error cannot execute this gdb command because it would break CBug

Example:

```
ucli% cbug::gdb info sources
```

Source files for which symbols have been read in:

```
../pythag.c, rmapats.c, ctype-info.c, C-ctype.c, C_name.c,  
../../gcc/libgcc2.c
```

Source files for which symbols will be read in on demand:

```
ucli% cbug::gdb whatis pythag  
type = int (int, int, int)  
ucli%
```

Add Directories to Search for Source Files

This is directly done with the `gdb dir dir-name` command. For example:

```
ucli% gdb dir /u/joe/proj/abc/src
```

Use this command to check which directories are searched:

```
ucli% gdb show dir  
Source directories searched:  
/u/joe/proj/abc/src:$cdire:$cwd
```

Adding directories may be needed to locate the absolute location of some source files.

Example:

```
ucli% cbug::expand_path_of_source_file foo.cpp
      Could not locate full pathname, try "gdb list
      sc_fxval.h:1" followed by "gdb info source" for more
      details. Add directories
      to search path with "gdb dir <src-dir>".

ucli% gdb dir /u/joe/proj/abc/src

ucli% cbug::expand_path_of_source_file foo.cpp
      /u/joe/proj/abc/src/foo.cpp
```

Note that adding a directory partially invalidates the cache used to store absolute pathnames. Files for which the absolute path name has already been successfully found and cached are not affected. But files for which the pathname could not be located so far will be tried again the next time if a new directory was added after the last try.

Common Design Hierarchy

An important part of debugging simulations containing SystemC and HDL is the ability to view the common design hierarchy and common VPDtrace file.

The common design hierarchy shows the logical hierarchy of SystemC and HDL instances in the way it is specified by the user. See also the VCS / DK1 documentation for more information how to add SystemC modules to a simulation.

The common hierarchy shows these elements for SystemC objects:

- Modules (instances)
- Processes:
 - SC_METHOD, SC_THREAD, SC_CTHREAD
- Ports: `sc_in`, `sc_out`, `sc_inout`,
 - `sc_in<T>`
 - `sc_out<T>`
 - `sc_inout<T>`
 - `sc_in_clk` (= `sc_in<bool>`)
 - `sc_in_resolved`
 - `sc_in_rv<N>`
 - `sc_out_resolved`
 - `sc_out_rv<N>`
 - `sc_inout_resolved`
 - `sc_inout_rv<N>`
- Channels:
 - `sc_signal<T>`
 - `sc_signal_resolved`
 - `sc_signal_rv<N>`
 - `sc_buffer<T>`
 - `sc_clock`

- rvm_sc_sig<T>
- rvm_sc_var<T>
- rvm_sc_event
- With datatype T being one of
 - bool
 - signed char
 - [unsigned] char
 - signed short
 - unsigned short
 - signed int
 - unsigned int
 - signed long
 - unsigned long
 - sc_logic
 - sc_int<N>
 - sc_uint<N>
 - sc_bigint<N>
 - sc_biguint<N>
 - sc_bv<N>
 - sc_lv<N>
 - sc_string

All these objects can also be traced in the common VPD trace file. Port or channels that have a different type, for example a user-defined struct, will be shown in the hierarchy but cannot be traced.

The common design hierarchy is generally supported for all combinations of SystemC, Verilog and VHDL. The pure-SystemC flow (the simulation contains only SystemC but neither VHDL nor Verilog modules) is also supported.

All these objects can also be traced in the common VPD trace file.
Interaction between CBug and the Simulator

The common design hierarchy is supported in the following combinations:

- SystemC top
- Vlog down

and

- SystemC top
- Vlog down
- VHDL down

and

- VHDL top
- SystemC down
- Vlog down

and

- Vlog top
- SystemC down

and

- Vlog top
- SystemC down
- VHDL down

Common VPD tracing and other debugging features are not supported in the following combinations:

- SystemC-top
- VHDL down

and also not

- VHDL top
- Verilog down
- SystemC instantiated from Verilog

and also not

- sc_main()
- SystemC top
- Verilog top

Post-processing Debug Flow

One way to use DVE is to first let the simulation run, create a VPD file and then look at the VPD file afterwards. This is called post-processing mode. All data will be contained in a VPD file.

There are different ways to create a VPD file. Not all are supported for common VPD with SystemC:

Supported

- Interactive using DVE and the **Add to Waves...** command
- Run the simulation in `-ucli` mode and apply `synopsys::dump` command

Not Supported

- With `$vcdpluson()` statement(s) in Verilog code
- With VCS option `+vpdfile`

If you create a VPD file in one of the unsupported ways, then you will not see SystemC objects at all. Instead you will find dummy Verilog or VHDL instances at the place where the SystemC instances are expected. The simulation will print warning that SystemC objects are not traced.

Use these commands to create a VPD file when SystemC is part of the simulation:

```
Create file dumpall.ucli :
  cbug::config add_sc_source_info always      <-- this line
                                              is optional, *1
  synopsys::cbg synopsys::cbg                <-- this line
                                              is optional, *1

  synopsys::scope .
  set fid [synopsys::dump -file dump.vpd -type VPD]
  puts "Creating VPD file dump.vpd"
  synopsys::dump -add "." -depth 0 -fid $fid
  synopsys::continue
```

Then run simulation like this:

```
simv -ucli < dumpall.ucli
```

The line `synopsys : cbug` is optional. If specified, then CBug will attach and store in the VPD file the source file/line information for SystemC instances that are dumped. This is convenient for post-processing: a double-click on a SystemC instance or process will open the source-code file.

Note that all source code must be compiled with compiler flag `-g` which will slow down the simulation speed to some extent (how much varies greatly with each design). Furthermore, attaching CBug will take some CPU time during which the underlying gdb reads all debug information. This seconds runtime overhead is constant. Last, attaching CBug creates a gdb process that may need a large amount of memory if the design contains many C/C++ files compiled with `-g` flag. In summary, adding the `synopsys : cbug` is a tradeoff between better debugging support and runtime overhead.

Interaction with the Simulator

Usually the C debugger and the simulator (the tool, e.g. `simv`) work together unnoticed. However, there are a few occasion when the C debugger and the tool cannot fully cooperate and when this is visible. These cases depend on whether the active point (the point where the simulation stopped, for example due to a BP) is in the C domain or HDL domain.

Prompt Indicates Current Domain

The prompt reflects if the simulation is stopped in the HDL or C domain.

- `ucli%` -> HDL domain

- CBug% -> C domain

Commands affecting the C domain:

Commands that apply to the C domain, for example setting a breakpoint in C source code, can always be issued, no matter in which domain the current point lies.

Some commands, however, can only be applied when the simulation is stopped in the C domain:

- The stack command to show which C/C++ functions are currently active
- Reading a value from C domain (e.g. a class member) with "synopsys::get" command is sensitive to the C function where the simulation is currently stopped. Only variables visible in this C scope can be accessed. That means it is not possible to access, for example, local variables of a C/C++ function or C++ class members when stopped in HDL domain. Only global C variables can always be read.

Combined Error Message:

When the C debugger is attached and you enter a command such as `get xyz`, then UCLI issues the command to both the simulator and the C debugger (starting with the one where the active point lies, e.g., starting with the tool in case the simulation is stopped in the HDL domain). If the first one responds without error, then the command is not issued again to the second one. However, if both tool and the C debugger produce an error message, the UCLI combines both error messages into a new one which is then printed. Example:

```
Error: {  
    {tool: Error: Unknown object}  
    {cbug: Error: No symbol "xyz" in current context.;}  
}
```

Update of Time, Scope and Traces

Any time, when the simulation is stopped in C code, the following information is updated:

- Correct simulation time
- Scope variable (accessible with `synopsys::env scope`) is either set to a valid HDL scope or to string "<calling-C-domain>"
 - If you stop in C/C++ code while executing a SystemC process, then the scope of this process is reported.
 - String "<calling-C-domain>" is reported when the HDL scope that calls the C function is not known. This happens, for example, in case of DPI, PLI, VhPI or DirectC functions.
- All traces (VPD file) are flushed

Configuring CBug

Use the `cbug::config` UCLI command to configure the CBug behavior. The following modes are supported:

Startup Mode

When CBug attaches to a simulation, then there are two different modes to choose from. Enter the UCLI command:


```
cbug::config startup fast_and_sloppy|slow_and_thorough
```

to select the mode before attaching CBug.

Mode 'slow_and_thorough' is the default and may consume much CPU time and virtual memory for the underlying gdb in case of large C/C++/SystemC source code bases with many 1000 lines of C/C++ source code.

Mode 'fast_and_sloppy' will reduce the CPU and memory needed, however, it comes on the expense that not all debug information is available to CBug right away. Most debugging features will still work fine, but there may be occasional problems, for example, setting breakpoints in header files may not work.

Attach Mode

```
cbug::config attach auto|always|explicit
```

Mode 'attach' defines when CBug attaches. Value 'auto' is the default and attaches CBug in some situations, for example when you set a breakpoint in a C/C++ source files and when double-clicking a SystemC instance. Value 'always' will attach CBug whenever the simulation starts. If value 'explicit' is selected, then CBug is never attached automatically.

cbug::config add_sc_source_info auto|always|explicit

The `cbug::add_sc_source_info` command stores source file/line information for all SystemC instances and processes in the VPD file. Doing that may take a long time but is useful for post-processing a VPD file after the simulation ended. Value 'auto' invokes

`cbug::add_sc_source_info` automatically when CBug attaches and the simulation runs without the DVE GUI; 'always' invokes `cbug::add_sc_source_info` automatically whenever CBug attaches; 'explicit' never invokes it automatically. Default is 'auto.'

Supported platforms

Interactive debugging with CBug is supported on the following platforms:

- Linux RH3/RH4/Suse, 32-bit
- Linux RH3/RH4/Suse, 64-bit (VCS flag -full64)
- Solaris 5.9/5.10, 32-bit

Interactive debugging with CBug is not supported on these platforms:

- Solaris, 64-bit
- -comp64 flow of VCS, all platforms
- any other platform

An explicit error message is printed when you try to attach CBug on a platform that is not supported.

For Solaris 64-bit, debugging of SystemC modules is only possible in the post-processing flow. Port/signals of SystemC modules can be dumped in a VPD file and later displayed by DVE. Note that this specific platform does not allow to store source file/line information for SystemC instances; doing a double-click on a SystemC instance or process will not open the corresponding source file.

Using SYSTEMC_OVERRIDE

VCS ships with multiple SystemC versions (2.0.1, 2.1, 2.2) which are used per default. In rare cases, it might be necessary to use a different SystemC installation that you compiled on your own. This can be done by setting the SYSTEMC_OVERRIDE environment variable (see the *VCS / VCSi User Guide*).

If you use SYSTEMC_OVERRIDE, then some or all of the SystemC specific CBug features are not available.

These features are not available:

- tracing of SystemC objects (ports, sc_signals)
- printing of SystemC native datatypes like sc_int in an intuitive format. Instead you will see the usual form how gdb prints the data which is generally useless for SystemC objects
- stopping in the next SystemC user process with 'next' or 'step'.

These feature may or may no work depending on how much different the SystemC installation is compared to an OSCI installation:

- showing SystemC objects (instances, processes, ports) in the common hierarchy (hierarchy pane in DVE)
- double-clicking an SystemC instance or processes to open the source file
- cross-stepping in or out of SystemC user processes and HDL code

Any other SystemC specific CBug feature

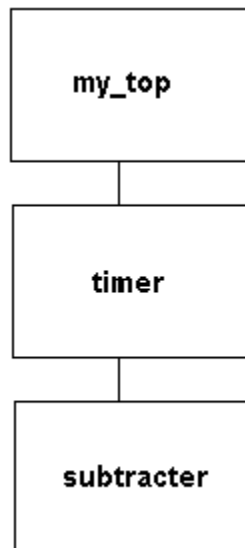
The following non-SystemC specific CBug feature will always work:

- setting breakpoints in SystemC source code (you may have to open the source file with File/Open File in DVE, though)
- stepping through SystemC source code. Note that stepping out of one SC user processes and stopping in the next one without a breakpoint is not supported).
- access a variable/class member with `synopsys::get`. The variable needs to be visible in scope of the C function where the simulation is currently stopped. Note that enhanced printing of native SystemC types is not available.

Example: A Simple Timer

The design is a simple timer that decrements a count starting from an initial value and signals an interrupt when the count reaches 0.

The testcase is a Verilog top + SystemC down testcase.



my_top is the driving module that drives the clock, reset, inval_valid, inval, current_val, and interrupt and samples the timer_val (current timer value and interrupt from the SystemC module).

Where:

- `reset` restores the initial value of the timer.
- `inval` is the timer initial value.
- `inval_valid` indicates to the timer that the input value is valid.
- `timer_val` is the current timer value.

The SystemC module for timer instantiates the `sc_subtractor` module that has `sc_current_value` as the input and returns `sc_current_value-1` as the result(`sc_current_value_minus_one`)

The `current_value-1` that is returned by the subtracter module is feedback to the subtracter module at every succeeding clock edge till the final value reaches 0 and interrupt is triggered.

1. Compile the testcase.

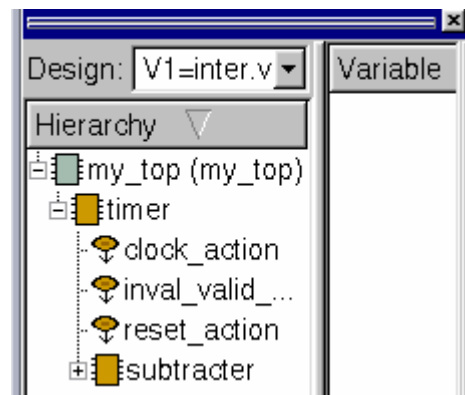
```
syscan -cpp g++ -cflags "-g" timer.cpp:timer sc_subtractor.cpp  
vcs -cpp g++ -sysc top.v -debug_all $VCS_HOME/lib/ucli.o -timescale=1ps/1ps
```


2. Star the DVE.

dve

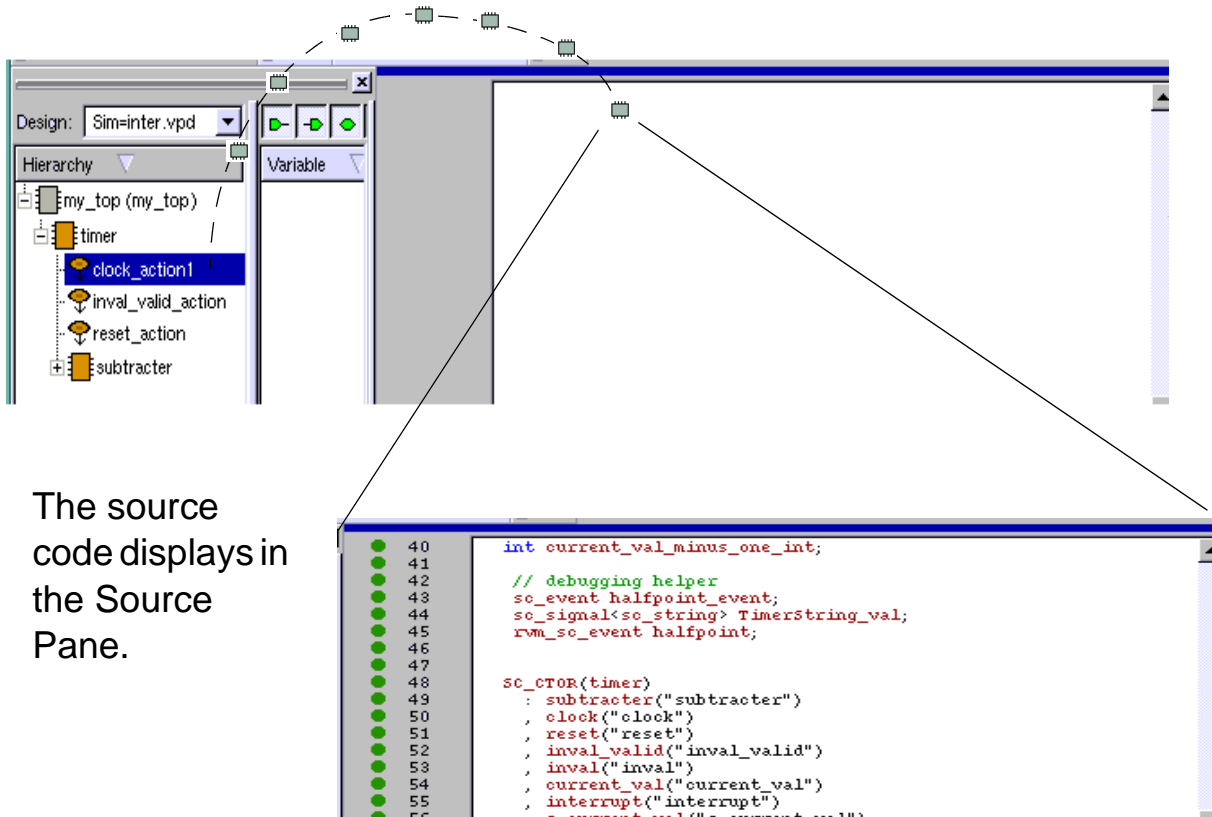
3. Click to start the simulation.

The DVE Hierarchy pane displays the design. Note that SystemC modules display in orange.



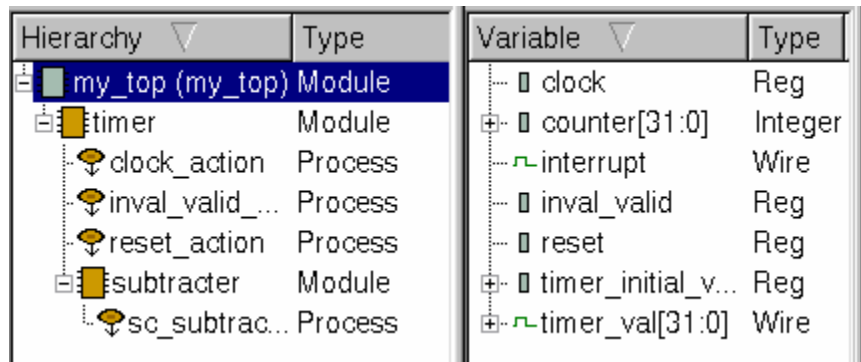
4. To display the source code of a process (), drag and drop the process from the Hierarchy Pane to the Source Pane..

Drag a process to the Source Pane



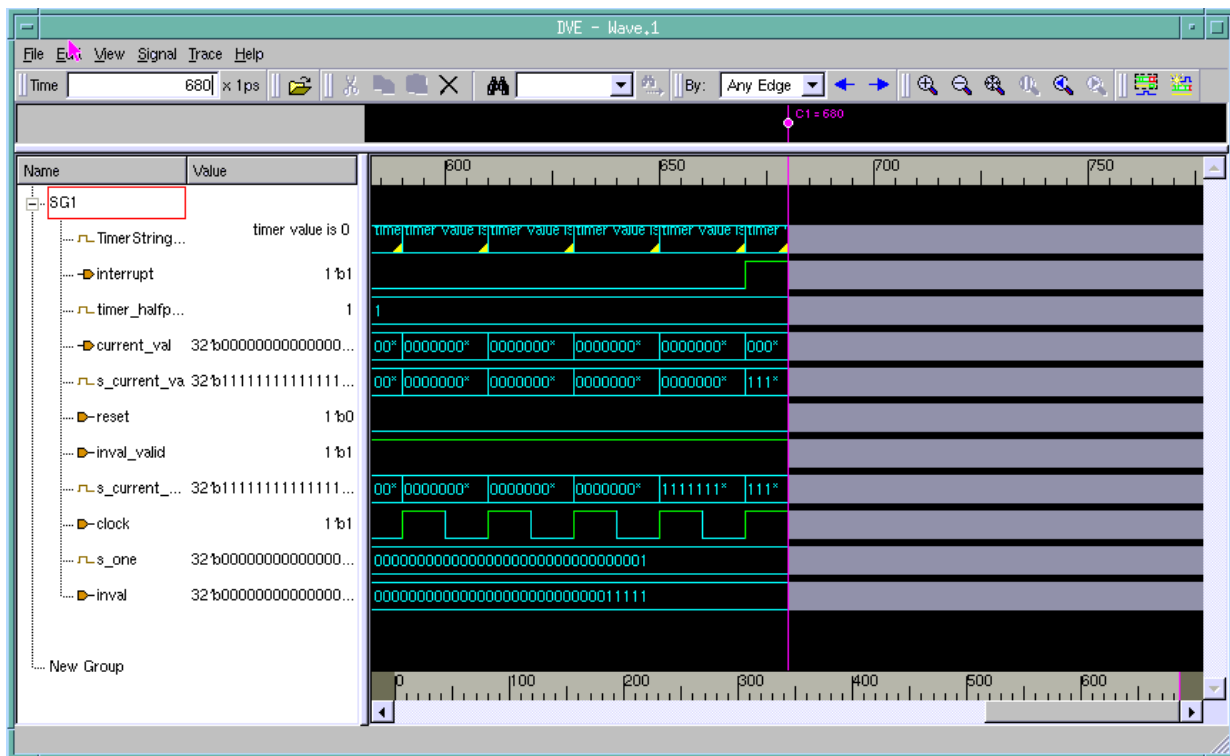
Note that anytime a C source code is opened or a breakpoint is set on a C source code the C debugger is launched automatically.

- Click on the design to display variables in the Variable Pane.



- In the Hierarchy Pane, select the top level of the design (**my_top (my_top)**), right-click, then select **Add to Waves** from the context-sensitive menu.

The Wave view displays waveforms up to the current simulation time. .

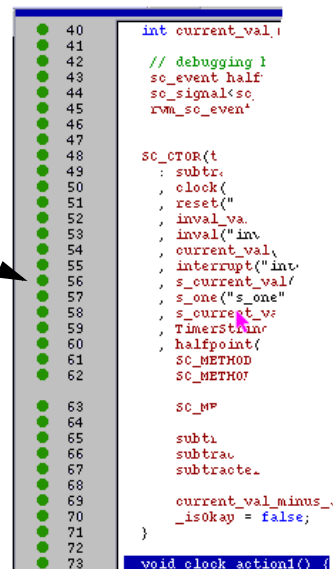


Note:

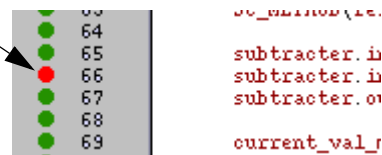
You cannot display the waveform while the simulator is stopped in C code.

7. To set a line breakpoint, click any green circle in the Line Attribute area of the Source Pane as shown below.

Click any breakable line indicator to activate line break.




Red circle indicates activated line breakpoint.



Note:

Similar to setting a breakpoint in the C code, you can set a breakpoint in HDL code and step back and forth between C and HDL code using the debugger.

8. Click  (**Continue**) in the Top Level Window toolbar.

The simulation advances to the breakpoint.

9. Enter finish in the console command line.

The simulation runs to the end.

A

Menu Bar and Toolbar Reference

This chapter describes the menu bar, toolbar, and command-line options. It includes the following topics:

- [“Menu Bar Options”](#)
- [“Editing Preferences”](#)
- [“Keyboard Shortcuts”](#)
- [“Toolbar Reference”](#)
- [“Using the Context-sensitive Menu”](#)
- [“Using the Command Line”](#)

Menu Bar Options

This section provides an overview of the following TopLevel window menus:

- “File Menu”
- “Edit Menu”
- “View Menu”
- “Simulator Menu”
- “Signal Menu”
- “Scope Menu”
- “Trace Menu”
- “Window Menu”
- “Help Menu”

File Menu

The following options comprise the **File** menu:

Options	Description
Open Database...	Displays the Open Database dialog box, which enables you to select and open simulation database (VCD or VPD) files for post-processing.
Close Database...	Displays the Close Database dialog box, which enables you to close an open simulation database (VPD) file.
Load Waveform Updates	Loads the waveforms updates.

Reload Databases	Reloads the open databases for post-processing.
Open File	Displays the Open Source File dialog box, which enables you to select and display a source file in the Source view.
Close File	Closes the source file displayed in the active Source view or window.
Save Values	Saves values according to the selection: Tabular List Event Based List Memory Contents
Execute Tcl Script...	Displays the Execute Tcl Script dialog box, which enables you to select and source a Tcl script.
Load Session...	Displays the Load Session Dialog, which enables you to load a saved session.
Save Session...	Displays the Save Session dialog box, which enables you to save the current session.
Load Last Auto-Session	Loads the previously saved session.
Print	Prints to printer or file the contents of an active wave, list, or Schematic view.
Recent Databases	Displays a list of recently opened databases to choose from.
Recent Tcl Scripts	Displays a list of recently run scripts to choose from.
Recent Sessions	Displays a list of recently opened sessions to choose from.
Close View/Pane	Closes the currently active view or pane but the TopLevel window will still be open.
Close Window	Closes the currently active TopLevel window.
Exit	Exits DVE.

Edit Menu

The following options comprise the **Edit** menu:

Options	Description
Cut/Copy/Paste/Paste From/Delete	<p>Copy works on any text. If the copy function can determine the text to be an object, copy will copy the object, otherwise it will copy the selected text. Copied text can be pasted in any widget that supports text, for example an editor or the DVE command line. Object copies work in widgets, such as DVE panes, which support DVE objects that sort DVE objects such as any DVE panes.</p> <p>Note: Cut and Delete work only on DVE objects and are limited to some windows, such as the Wave, List, and Memory views.</p> <p>Paste From generates a list of items that were copied from the clipboard.</p>
Expand By Levels >	<p>Contains the following submenu items:</p> <ul style="list-style-type: none">All – Expands all items for all levels under the currently selected item. This option can take a long time if executed at the root level of a huge design.2 – Expands 2 levels of child items from the currently selected item.3 – Expands 3 levels of child items from the currently selected item.4 – Expands 4 levels of child items from the currently selected item.5 – Expands 5 levels of child items from the currently selected item.
Expand All	Expands all child items under all parents regardless of what item is selected.
Collapse Parent	Collapses to the parent of the currently selected item. If no item is selected, no action is taken.
Collapse All	Collapses all children to the topmost parent regardless of what item is selected.

Synchronize Selection	<p>Selection is not global. You can have different items selected in different panes at any time. Synchronize Selection allows you to sync up all panes to one selection.</p> <p>For example, if you have a signal selected in the Wave window, but its parent scope is not shown in the Hierarchy window, clicking on Synchronize Selection will cause the parent scope of the signal to be highlighted in the Hierarchy window and also that signal will be highlighted in the Data pane and any other pane where it exists. This functionality is particularly useful in the Schematic window.</p>
Select by Levels	<p>Contains the following submenu items:</p> <ul style="list-style-type: none"> All – Selects all items for all levels under the currently selected item. Note: This can take a long time if executed at the root level of a huge design. 2 – Selects 2 levels of child items from the currently selected item. 3 – Selects 3 levels of child items from the currently selected item. 4 – Selects 4 levels of child items from the currently selected item. 5 – Selects 5 levels of child items from the currently selected item.
Select All	Selects all objects in a pane or window.
Find...	Displays the Find dialog box.
Find Next	Active if any text exists in the Find dialog box or find menu line edit. If clicked, it finds the next occurrence of the text in the active pane.
Find Previous	Similar to Find Next, but finds the previous occurrence of text.
Go To Address...	Displays a dialog box in which you can enter an address. This menu option is active if the Memory view is open and populated.
Search for Signals/ Instances...	Displays the Search for Signals dialog box. Use this to find any object that exists in the opened and current database. If the object is not loaded, this dialog box will attempt to load it.

Create Marker	Creates a marker in the Wave view. This is only active if a Wave pane exists in the TopLevel window for which the Edit menu is activated. If a Wave pane exists, clicking this menu option puts you into create marker mode. A white hashed marker is created and it follows the mouse in the Wave pane. The marker is placed at the location of the next mouse-click.
Markers...	Displays the Marker dialog box (see “Cursors and Markers” on page 5-17).
Go To Marker	Provides a list of markers that you have created in the current Wave view. You can select any marker to view the marker in the center of the Wave view. If no markers are present in the current window, the submenu will be empty.
Delete Marker	Provides a list of markers that you have created in the current Wave view. You can select any marker to be deleted. If no markers exist, the submenu will be empty.
Move Marker	Provides a list of markers that you have created in the current Wave view. You can select any marker, then click the desired location in the Wave pane to move the marker to that location. If no markers are present in the current window, the submenu will be empty.
Set Reference Marker	Sets a currently selected marker as the reference marker for displaying values in relation to other markers.
Show Marker Values	Displays Absolute, Adjacent, or Relative values for signals at a selected marker (see “Cursors and Markers” on page 5-17).
Preferences...	Displays the Application Preferences dialog box (see “Editing Preferences” on page 2-11).

View Menu

The following options comprise the **View** menu:

Options	Description
---------	-------------

Selection Tool	Used for schematic views. You can use this option to select objects. Control-click adds objects to the selected set. Click and drag creates a box. Everything inside the box will be selected.
Zoom In Tool	Used for Schematic views. Enlarges the view of the selected object.
Zoom Out Tool	Used for Schematic views. Enables you to reduce the magnification level of an active object.
Pan Tool	Used for schematic views. Allows you to interactively move the center of the object without changing the scale. When you select this tool, the cursor changes to a hand. Click the toolbar selection arrow or press the Esc key to change the cursor back to normal.

Zoom>	<p>Contains a submenu for all zoom operations. These menu items are only applicable for Schematic and Wave views.</p> <p>Zoom Full – Fits all viewable objects into the current view.</p> <p>Zoom In – Enlarges the object.</p> <p>Zoom Out – Reduces the magnification level of the selected object.</p> <p>Zoom Fit Selection – Changes the view to center the selected set of schematic objects so all selected objects are visible in the current pane.</p> <p>Zoom Fit Highlight – Same as Zoom Fit Selection, but for highlighted schematic objects. This option is available if there are highlighted objects.</p> <p>Pan To Selection – Moves the view so the selected set of schematic objects is centered and viewable in the current pane. It does not change the zoom.</p> <p>Pan To Highlight – Same as Pan To Selected, but for highlighted schematic objects.</p> <p>Zoom to Cursors – If the two Wave view cursors are present, this puts the area between the cursors in the view with each cursor at opposite sides of the view.</p> <p>Zoom to Time Range... – Displays a dialog box for entering a time range, then zooms to that time range.</p> <p>Back in Zoom and Pan History – Iterates through saved zoomed or panned views for the current pane. When you change a zoom or a pan in a view, DVE stores the previous view so you can retrieve it.</p> <p>Forward in Zoom and Pan History – If you have gone backwards in zoom/pan history, this menu item provides an easy way to go to the next view. Clicking this item will eventually get to the current view.</p> <p>Named Zoom and Pan Settings... – Displays a dialog box that allows you to choose from any view that you saved with a name.</p>
Set Time Scale...	Opens the Set Time Scale dialog box that allows you to change the time unit.
List Window Time Range...	Opens the Set Time Range dialog box that allows you to change the start time and end time in the List view. This option is available only from the List view menu bar.

Delta Cycle >	<p>This is available only if the delta cycle information exists in the database. (See also Capture Delta Cycle Values in Simulator Menu.)</p> <p>Expand Time – Expands at the simulation time (C1) to show the delta cycles within that time.</p> <p>Collapse Time – Collapses the expanded delta cycle display at C1 time.</p> <p>Collapse All – Collapses all the expanded delta cycles.</p>
Associate With>	Associates a signal group in a waveform or signal group pane with a specified database, so that when a signal from a different database is being added, the signal with the same hierarchical name of the specified database is actually added. Associate with "Any Database" to remove this setting.
Go to Beginning	Goes to the start of C1 (usually 0). This moves the C1 cursor and changes the view in the Wave pane to show the beginning of the simulation time.
Go to End	Same as above, but for the end of C1.
Go to Time	Displays the Go To Time dialog box that allows you to change the C1 time in the debugger. The Wave view shows the new C1 time.
Link C1 to Sim Time	Sets the debugger time with the current interactive simulation time. This option is available only in an interactive debug session.
Move C1 to Sim Time	Synchronizes the debugger time with the current interactive simulation time. This option is available only in an interactive debug session.
Use Global Time (C1)	Keeps C1 in the current TopLevel with the global C1 marker.
Increase Row Height	Increases the height of all traces in the Wave view.
Decrease Row Height	Decreases the height of all traces in the Wave view.
Set Default Row Height	Resets the height of all traces in the Wave view to the default.
Watch	<p>Provides some operations for the Watch pane:</p> <p>Add New Page - Adds a new page in the Watch pane.</p> <p>Delete Current Page - Deletes the currently selected page from the Watch view.</p> <p>Rename Current Page - Assigns a new name to the current page.</p> <p>Edit Variable - Lets you edit the variable name.</p>

Toolbars >	Contains the following submenu. You can select or clear the following check boxes to view or hide them as desired.
	<ul style="list-style-type: none"> Edit File Scope Trace Window Signal Simulator Time Operations Zoom Zoom and Pan History

Simulator Menu

The following options comprise the **Simulator** menu.

Options	Description
Setup...	Displays the Simulation Setup dialog box to allow modification of default simulation runtime settings. Note: This does not allow you to control simulator compile-time settings (see “Running a Simulation from the GUI” on page 1-12).
Rebuild and Start	Rebuilds the simulator by executing the VCS Makefile, then starts the simulation.
Start/Continue	Runs the simulation until a breakpoint is hit, the simulation finishes, or for the duration specified in the Set Continue Time dialog box.
Stop	Stops a running simulation (same as Control-c in UCLI mode).
Step	Moves the simulation forward by stepping one line of code, irrespective of the language of the code. This is the same as the UCLI Step command.
Next	For VHDL, Verilog, and TB code, next steps over tasks and functions.
Next in CBug	Advances to the next line in CBug code.
Step In Active Thread	Stops at the next executable line in the current active thread.

Step In Testbench	For Native Testbench (NTB) OpenVera and SystemVerilog testbenches, stops at the next executable line in the testbench.
Step Out	Steps to the next executable line outside of the current function or a task.
Restart	Stops the currently running simulation and restarts it with the current simulation setup. This retains all open windows and GUI setups. If the simulation is not running, this option will start it.
Show stack	Displays C-language, SystemVerilog and Native Testbench stacks in Console pane.
Move up Stack	Steps up the current stack.
Move down Stack	Steps down the current stack.
Breakpoints...	Displays the Breakpoints dialog box that allows you to view, create, edit, enable, disable and delete breakpoints (see “Setting Breakpoints in Interactive Simulation” on page 4-8).
Save State...	Displays a File Browser dialog box that allows you to save the current state of the simulator as a file.
Restore State...	Displays a File Browser dialog box that allows you to restore a saved simulation state.
Terminate	Kills a running simulation.
Dump Full Hierarchy	Dumps the hierarchy of the entire design into a VPD file without triggering the dump of values.
Add Dump...	Displays the Dump Values dialog box to specify scopes or signals to dump value change information starting at the current time (see “Dumping Signal Values” on page 3-10).
Dump	Turns on value change dumping at the current time for any selected scope or signal in the active pane.
Add Force...	Displays the Force Values dialog box to allow you to interactively change the value of a signal (see “Forcing Signal Values” on page 3-11).
Capture Delta Cycle Values	Toggles delta cycle dumping on/off starting at the current time. Note: This substantially increases the VPD file size. You should try to limit the time span for dumping delta cycle values.

Continue For Time...	Displays the Continue for Time dialog box where you enter a time. The time specifies the duration for which the simulation runs if no breakpoints are hit. For example, if set to 10, the simulation runs ten times when you click the Continue toolbar button.
Periodic Waveform Update Interval...	Opens the Value Update Interval dialog box. You can choose time interval to enable periodic waveform update. This allows you to see waveforms dynamically as the simulator runs. The smaller the interval the worse the performance.
C/C++ Debugging	<p>Enable - Enables debugging of C, C++, and SystemC source code. Allows you to step in C/C++ code, set breakpoints, etc.</p> <p>Show External Functions - Shows user-defined external functions (PLI, DPI, Direct C).</p>

Signal Menu

The following options comprise the **Signal** menu:

Options	Description
Display Signal Group >	<p>Contains the following submenu that allows you to group the signals in the Wave view.</p> <p>New Signal Group - Creates a new signal group.</p> <p>All – Turns on visibility for all existing signal groups. All the signal groups are listed. You can choose to display or hide the particular signal group by selecting or clearing the respective check box.</p>
Add to Waves	<p>Contains the following submenu:</p> <p>New Wave view - Adds the selected signal to a new Wave view.</p> <p>Recent (New View) - Adds the selected signal to a recently created Wave view. If none exists, creates a new Wave view.</p> <p>Create New Group - Creates a new group in the Wave view and adds the selected signal under the newly-created group.</p>

Add to Lists	<p>Contains the following submenu:</p> <p>New List view - Adds the selected signal to a new List view.</p> <p>Recent (New View) - Adds the selected signal to a recently created List view. If none exists, creates a new List view.</p> <p>Create New Group - Creates a new group in the List view and adds the selected signal under the newly-created group.</p>
Add to Groups	<p>New Group – Creates a new signal group. The name will be Group<<i>n</i>>, where <i>n</i> is one more than the highest number of existing signal group. The new signal group is created at the top of the signal list.</p> <p>The existing signal groups are also displayed. You can add signals to the existing signal groups.</p>
Add to Watches	Adds a signal to the Watch pane.
Show Memory	Displays a signal in the Memory viewer if the selected signal is a memory or MDA. If there is no memory viewer, DVE creates one according to the target policy.
Set Insertion Bar	Sets the insertion bar above the selected signal in the Wave or List view. This menu item is not available in other windows. If more than one signal is selected, DVE places the insertion bar above the signal closest to the top of the signal list.
Insert Divider	Inserts a blank divider row in the waveform display.
Sort Signals	Displays signals by declaration, ascend, or descend.
Set Bus...	Displays the Bus/Expression dialog box for managing the bus and expression creation/deletion.
Set Expressions...	Displays the Bus/Expression dialog box for managing the bus and expression creation/deletion.

Set Search Constraints>	<p>Searches for signals per the search constraint. When the constraint is matched, the C1 cursor moves to that time location. Searching works only on the selected set of signals. If no signals are selected, it will search all the signals in the current view.</p> <p>Following are search constraints:</p> <p>Any Edge – (Default) Searches for signals with any edge.</p> <p>Rising – Searches for signals with rising edge.</p> <p>Falling – Searches for signals and points to the falling edges.</p> <p>Failure – Stops on next or previous assertion failure. This options is available for assertion signals.</p> <p>Success – Stops on next or previous assertion success. This options is available only if the signal is an assertion.</p> <p>Match – Searches for the next or previous match.</p> <p>Mismatch – Stops on next or previous assertion mismatch. This option is available only if the signal is an assertion.</p> <p>X Value – Searches for any signal that contains x value.</p> <p>Signal Value... – Opens the Value Search dialog box that allows you to enter a specific value as the constraint. If the value is found, the search will stop and the cursor C1 is positioned at that value. Values must match the radix that is currently selected for the signal.</p>
Search Backwards	Searches backwards in time.
Search Forward	Searches forward in time.
Compare...	Displays the Signal Compare dialog box where you can select waveforms to compare (see “Comparing Signals, Scopes, and Groups” on page 5-24.)
Show Compare Info	Shows the results of the last signal comparison. This option is available only if a signal comparison has been performed.
Analog Overlay	Overlays selected signals in a Wave view in a overlay signal group and displays the waveform in analog style.
Unoverlay	Unoverlays the overlaid signals.
Shift Time...	Displays the Shift Time dialog box to specify the parameters to shift a signal in time.

Set Radix >	<p>Provides options for radix changes of the selected signal. Changing radix on a signal is global and will change the radix wherever the signal is displayed in DVE.</p> <p>User Defined > Allows you to specify and edit user-defined radices (see “Managing User-defined Radix” on page 5-9.) The other radices available are:</p> <p>Enumerated Type ASCII Binary Octal Decimal Hexadecimal Unsigned Signed magnitude One's Complement Two's Complement Strength Default</p>
Default Properties	Applies default properties to the selected signal. Clear the check box to allow the selected item to have its own radix and signal property.
Properties...	Displays the Signal Properties dialog box that allows you to change the signal properties.

Scope Menu

The following options comprise the **Scope** menu:

Options	Description
Show Source	Shows the source of the object selected. If multiple objects are selected, shows the source of the first object in the selected set. If the Use check box is selected, DVE uses the currently open Source view. If no Source view exists, DVE creates a new Source view according to the target policy.
Show Schematic	Same as above except that it shows the object in a Design Schematic pane.
Show Path Schematic	Same as above except that it shows the object in a Path Schematic pane.

Note: The following menu items affect the currently active Source and Schematic views.

Move Up to Parent	Moves the selection in the Source view up one level of hierarchy from the scope of the currently selected line. If the current line is the top of the hierarchy or no line is selected, this item is not available.
Move Down to Definition	Moves the selection to the start of the definition of the currently selected object. Note that in the Source view, this only works if the object itself is selected. It does not work if the entire line is selected and more than one token is selected on that line.
Back	Moves to the previous view of source information in the current Source view. DVE maintains a Source view history, so that it is easy to go back to a previous view of the source. This is useful for large source files and reduces the need for scrolling.
Forward	Same as above but moves forward in the Source view history if you have previously gone backwards.
Show >	<p>Allows navigation to certain types of relative source lines based on the selected object type. The Source view only shows definition. This menu contains the following submenu:</p> <ul style="list-style-type: none"> Definition – Shows the definition of the selected object. Current Scope – Changes the selection to the first line of the current scope. Assertion – Changes the Source view to the definition of the selected assertion if the object is an instance of an assertion. Unit Binding – Changes the Source view to the location where the assertion is bound to a module with a bind statement for OVA assertion instances. Entity – Changes the Source view to the architecture's entity if the current selection is in a VHDL architecture. Architecture – Changes the Source view to the entity's architecture if the current selection is in a VHDL entity. Macro - Shows the value and information of the selected macro in a separate Source view. The Source view contains all the queried macros in it. Macro Definition - Shows the definition of the selected macro in the Source view.
Edit Source	Displays a text editor based on your Editor source preference setting (vi is the default editor). DVE preloads the editor with the source file that is in the currently active DVE Source view and positions it on the same selected line. If no file is open in the Source view or a different kind of DVE pane is active, this menu option is not available.

Edit Parent	Same as above except that the source of the parent instance is preloaded in the text editor.
Expand Path	Expands fanin or fanout of the current path to one additional level from the selected object.
Add Fanin/ Fanout...	Displays the Fanin/Fanout dialog box in which you specify fanin/fanout parameters for the path schematic. This menu option is available for Path Schematic only.
Annotate Values	Allows you to toggle signal annotation on and off for the current scope in Source/Data/Schematic/Path Schematic views. Annotation allows you to see values within the context of the display. For example, in the Source view, the annotation will be below the source text for variables, while in the schematic, the annotation will be on pins or nets.
Properties	Opens the Properties dialog box that shows all available properties for the currently selected schematic or path schematic object. This menu option is available for Schematic and Path Schematic only.

Trace Menu

The following options comprise the **Trace** menu:

Options	Description
Trace Assertion	<p>Traces the assertion and automatically displays the results in the current or new Wave view. This menu option is active only if an assertion is selected in the currently active pane and certain conditions are met (<code>libassertiondebug.so</code> must be available).</p> <p>If a specific assertion attempt is selected, that attempt will be traced. If you select no specific attempt, DVE traces the first attempt. The assertion trace gives detailed debugging information for a specific assertion attempt, so you can easily point to the expression in the assertion that failed.</p>
Assertion Attempts...	Displays the Assertion Attempts dialog box. If the selected object in the currently active pane is an assertion, the dialog box will be populated with all attempt information for that assertion. If the selected object is not an assertion, the dialog box is empty.

Trace Drivers	<p>Finds the active driver of the object, shows the driver in the drivers/loads pane and shows the driver in the reusable Source view in the current window. If no window is present, DVE creates one. If a drivers pane already exists, the new trace information is added to it.</p> <p>This menu option will be active if the object selected in the currently active pane is a variable or signal. For this capability to work, the design must be compiled with one of the <code>-debug</code> options with a complete <code>simv.daidir</code> directory.</p>
Trace Loads	Finds loads of the signal.
Drivers/Loads	Allows you to navigate and manage drivers/loads (see Chapter 9, "Tracing Drivers and Loads").
Follow Signal	Displays a list of instances of a selected signal in the Source view.
Highlight >	<p>Allows you to highlight and manage highlighted objects in all panes.</p> <p>Recent Color - Highlights any currently selected object in the current view with the recently used color.</p> <p>Clear Selected – Removes all highlight color from the selected object.</p> <p>Clear All – Removes all highlights regardless of whether the object is selected or not.</p> <p>Clear by Color - Removes the highlights from the object. (The color is selected in the submenu.)</p>
Spot Signal Path	Follows the selected signal through the schematic design path.
Stop Signal Spotting	Stops following the signal.
Trace X	Traces the X value in the selected signal to its source so it can identify the signals that caused the X value in the Schematic and Path Schematic views.

Window Menu

The following items comprise the **Window** menu:

New >	Opens a new instance of one of the following window selections according to the current target criteria: Source View Schematic View Path Schematic View Wave View List View Memory View
SetThisFrame Target For >	Sets the currently active frame as the target for one of the following selections: Source View Schematic View Path Schematic View Wave View List View Memory View
Panes >	Displays the selected pane in the active TopLevel window. Console Hierarchy Data Signal Groups DriverLoad Stack Local Watch Assertion
New Top Level Frame >	Opens a new TopLevel frame displaying one of the following selections: Empty Assertion + Hierarchy Hierarchy + Data Hierarchy + Data + Console Console
Load Default Layout	Returns the currently active TopLevel frame to the default layout.

Load Layout	<p>Loads a layout session.</p> <p>Reset Layout - Resets the layout to the initial layout.</p> <p>From File - Loads a pre-saved layout session file.</p>
Save Current Layout	<p>Saves the current DVE layout.</p> <p>To default - Saves the current DVE layout as <code>~/ .synopsys_dve_default_layout.tcl</code>. When DVE restarts, it uses this default layout instead of the last layout when you have exited DVE.</p> <p>To File - Saves the current DVE layout to a layout session file.</p>
Arrange	<p>Arranges all the non-docked panes in current TopLevel window.</p> <p>Cascade - Display windows in cascade format.</p> <p>Tile - Display windows in tile format.</p> <p>Vertical - Display windows in vertical tile format.</p> <p>Horizontal - Display windows in horizontal tile format.</p>
Dock in New Row >	<p>Positions the currently active window or pane in a new row of the current TopLevel frame according to one of the following selections:</p> <p>Left</p> <p>Right</p> <p>Top</p> <p>Bottom</p>
Dock in New Column >	<p>Positions the currently active window or pane in a new column of the current TopLevel frame according to one of the following selections:</p> <p>Left</p> <p>Right</p> <p>Top</p> <p>Bottom</p>
Undock	Undocks the selected window from the TopLevel window.
Move To	Moves to a new TopLevel window.
Set Top Level Title	Sets the title of the TopLevel window.
Current Window List	Lists the current TopLevel windows.

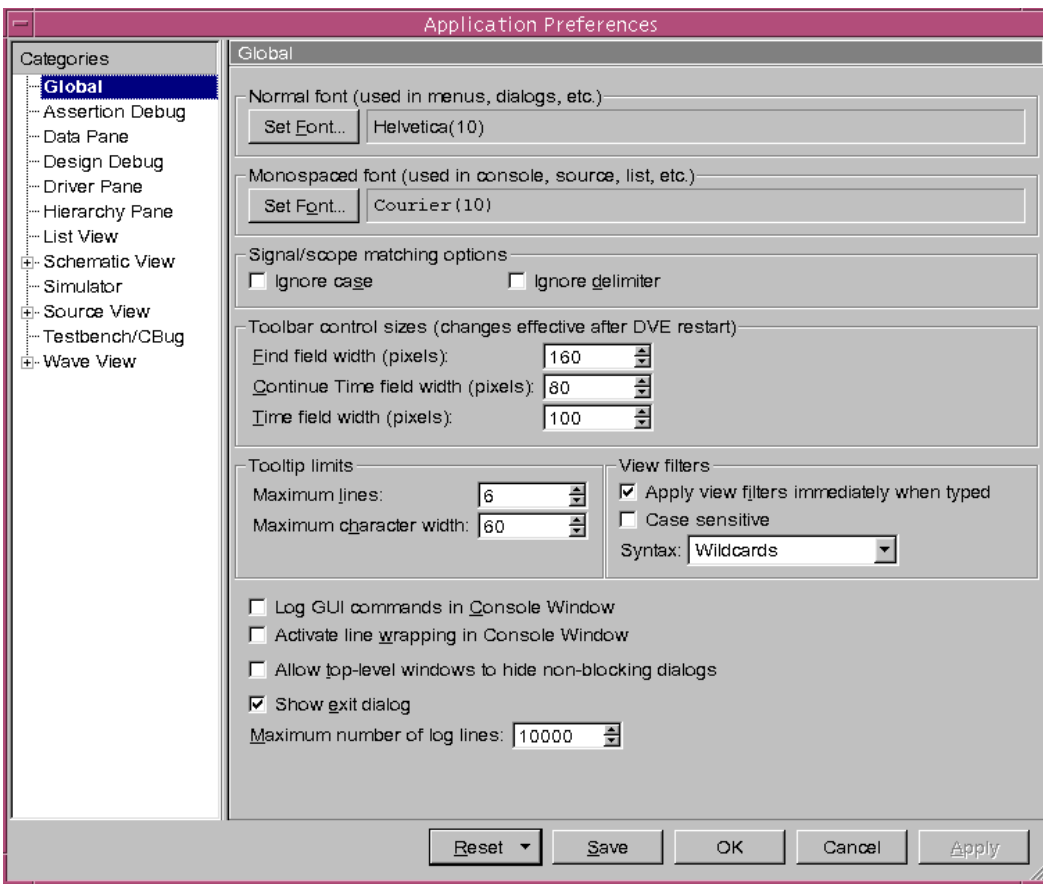
Help Menu

The following items comprise the **Help** menu:

DVE Help	Opens the DVE documentation.
A Quick Start Verilog Example	Loads an example design.
A Quick Start Mixed Example	Loads an example design when VCS/VCS MX is running.
Tutorial for Mixed Example	Loads an example design for VCS/VCS MX.
About	Displays DVE version and copyright information.

Editing Preferences

DVE creates the `.synopsys_dve_prefs.tcl` file, which stores user preference information. The Application Preferences dialog box contains the following categories and options.



- Global - Select settings to set the font and font sizes to display in DVE windows. You can specify multiple database options by selecting whether to ignore the case or delimiter when matching signals and scopes.

The default is to log only UCLI commands. To also log GUI commands, select the "Log GUI commands in Console Window" check box. You can specify the maximum number of log lines.

- Source view – Specifies data and annotation loading options, line wrap, line number display, tab width, default editor, and automatic reload of changed source code.

Source colors – Specifies colors to display Source view components by clicking the drop-down arrows and clicking on a color.

- Schematic View:
 - Click the up and down arrows to set the maximum number of cells.
 - Specify display characteristics for the number of cells and text size.
 - Check or clear the **Visibility** check boxes to filter the display of design elements.
 - In the Line column, select colors for design elements.
 - To customize value annotations, click on the plus sign (+) in the categories pane next to the Schematic view and select **Value Annotation**.
 - Select the Port/Pin visibility and color.
- Waveform view:
 - **Show grid** – Show the signal name by level, and justify text preferences for the signal pane. These settings are similar to data pane preferences.
 - **Seek next can advance simulation** – Valid for interactive simulation only. Controls whether or not searching (seek) for a value or edge with the Seek Next functionality advances the simulation to complete the requested seek.

- **Show marker values** – Markers can show 1, 2, or 3 values. This preference controls which values (can be any combination or all) are displayed.

Absolute – The time at which the marker is set.

Adjacent – The time between the marker and the next closest marker on either side.

Relative – Time with respect to the reference marker (usually C1).

- **Analog values change within one pixel** - Sets the number of values to plot. **Rough plot** can speed up performance. **Plot All Values** can be slower.

Rough plot – Smooths plot value display.

Plot All Values – Displays all values. This option can be slower.

- **Label Assertion attempts with start time** – Chooses when to annotate assertion attempt waveform (up arrow) with the time the assertion started. This is useful to see start and end time in one location, but in the case of short duration assertions, it can add clutter to the waveform display.

Failures only – Annotates failures.

Always – Shows the start and end time of all assertions.

Never – Does not annotate assertions.

- **List view** – Specifies grid display, signal name truncation, signal levels to display, and column spacing settings.

- Debug Settings – Select signal compare parameters, value transition, and assertion window docking defaults, and first frame target setup options.
- Hierarchy pane – Sets the appearance and initial filter states.
- Data pane – Sets the appearance parameters, signal sorting, signal levels to display, and scroll bar conditions.

Keyboard Shortcuts

You can create a `.synopsys_dve_usersetup.tcl` file in your home directory for storing shortcuts.

Example

```
gui_set_hotkey -menu "Signal > Compare..." -hot_key "c"
```

File Command Shortcuts

Control+O	Open database
Control+W	Close window
Control+U	Load Waveform updates

Edit Command Shortcuts

Control+X	Cut
Control+C	Copy
Control+V	Paste
DEL	Delete
Control+Y	Synchronize selection
Control+A	Select all
Control+F3	Find
F3	Find next
Shift+F3	Find prev
Control+G	Go to address
Control+H	Search for signals/ instances

View Command Shortcuts

ESC	Selection tool
=	Zoom in tool
-	Zoom out tool
F	Zoom full
+	Zoom in
O	Zoom out
Control+T	Zoom fit selection
Control+Alt+T	Zoom fit highlight
Control+Q	Zoom to cursors

Simulator Command Shortcuts

F5	Start/Continue
F11	Step
F10	Next
Control+F10	Next in CBug
F12	Step in active thread
Control+F11	Step in testbench
F9	Step out
Control+F5	Restart

Signal Command Shortcuts

Control+4	Add to waves
Control+5	Add to lists
Control+6	Add to watches
Control+7	Show memory
<	Search backward
F4	Search forward

Scope Command Shortcuts

Control+1	Show source
Control+2	Show schematic
Control+3	Show path schematic
F8	Move up to parent
Shift+F8	Move down to definition

Trace Command Shortcuts

Control+D	Trace drivers
-----------	---------------

Control+L	Trace loads
Control+E	Highlight recent color
Control+Shift+M	Clear selected
Control+M	Clear all

Help Command Shortcuts

F1	Help
----	------

Toolbar Reference

This section describes all toolbar text fields, menus, and icons. You can drag and drop toolbars into any location in a TopLevel DVE window toolbar using the toolbar handles.

To toggle the display of toolbars on and off, select **Edit > Toolbars**, then select the desired toolbar.

Edit

The following items comprise the **Edit** toolbar:

Icon	Description
------	-------------



Cut, Copy, Paste, Delete

Copy works on any text. If the copy function can determine the text to be an object, copy will copy the object, otherwise it will copy the selected text. Copied text can be pasted in any widget that supports text, for example, an editor or the DVE command line.

Object copies work in widgets, such as DVE panes, which support DVE objects that sort DVE objects such as any DVE panes.

Cut and Delete works only on DVE objects and some windows, and are limited to some windows, such as the Wave, List, and Memory views.



Search for Signals/Instances

Displays the Search for Signals dialog box. Use this to find any object that exists in the opened and current database. If the object is not loaded, this dialog box will attempt to load it.



Selects string to search, then press Enter to search.

Find







Find Previous/Next

Active if any text exists in the Find dialog box or the Find menu text box. If clicked, finds the previous or next occurrence of the text in the active pane.

File




The following items comprise the **File** toolbar:


Icon	Description
------	-------------

 Open Database or File	Displays the Open Database or Open File dialog box, depending on the DVE window displayed, and enables you to select and open a VPD file.
 Close Database	Displays the Close Database dialog box, which enables you to close an open simulation database (VPD) file.
 Load Waveform Updates	Loads waveform updates
 Print	Prints to a printer or file the contents of an active wave, list, or Schematic view.

Scope



The following items comprise the **Scope** toolbar:

Icon	Description
 Annotate Values	Displays the currently active scope signal values at the current time in the Source view.
 Move Up One Level	Moves the selection in the Source view up one level of hierarchy from the scope of the currently selected line. If the current line is the top of the hierarchy or no line is selected, this item is not available.
 Move Down One Level	Moves the selection to the start of the definition of the currently selected object. Note: In the Source view, this only works if the object itself is selected. It does not work if the entire line is selected and more than one token is selected on that line.

 Move Backward/Forward in List of Scopes	<p>Backward or forward arrow moves to the previous view of source information in the current source view, or forward in the source view history if you have previously gone backwards.</p> <p>DVE maintains a history of Source views, so that it is easy to go back to a previous view of the source. This is useful for large source files and reduces the need for scrolling.</p>
------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------








Trace


The following items comprise the **Trace** toolbar:

Icon	Description
 Trace Drivers/Trace Loads	<p>This menu item is active if the object selected in the currently active pane is a variable or signal. For this capability to work, the design must be compiled with one of the <code>-debug</code> options and an <code>mdb</code> library must exist.</p> <p>When selected, this finds the active driver of the object, shows the driver in the Drivers/Loads pane and shows the driver in the reusable Source view in the current window. If no window is present, DVE creates one. If a drivers pane already exists, the new trace information is added to it.</p>
 Find Next/Previous	<p>Finds the next or previous driver or load, or the next or previous driver or load in the current instance, respectively.</p>

View

The following items comprise the **View** toolbar:

Icon	Description
 Show Source	Opens a new Source view and displays source for the selected object.
 Show Schematic	Opens a new Schematic view.
 Show Path Schematic	Opens a new Path Schematic view.
 Show Wave	Opens a new Wave view or displays a previously opened view.
 Show List	Opens a new List view or displays a previously opened view.
 Show Memory	Opens a new Memory view.
 Add to Watches	Adds a signal to the Watch pane.

 <p>Assertion</p>	<p>Opens the Assertion pane.</p>
----------------------------------------------------------------------------------------------------	----------------------------------

Signal









The following items comprise the **Signal** toolbar:


Icon	Description
------	-------------

<div data-bbox="274 275 636 348" data-label="Image"> </div> <p data-bbox="245 369 365 405">Search</p>	<p data-bbox="792 260 1382 359">Backward or forward arrow launches search in time for the constraint selected in the list box.</p> <ul data-bbox="792 369 1382 1394" style="list-style-type: none"> •Any Edge – (Default) Search stops and positions C1 cursor on the next or previous edge found. •Rising – Search stops and positions C1 cursor on the next or previous rising edge only. •Falling – Search stops and positions C1 cursor on the next or previous falling edge only. •Failure – Available only if the signal is an assertion; stops on next or previous assertion failure. •Success – Available only if the signal is an assertion; stops on next or previous assertion success. •Vacuous - Available only if the signal is an assertion; stops on the next or previous vacuous success. •Signal Value ... - Displays a small dialog box that allows you to enter a specific value as the constraint. If the value is found, the search will stop and position C1 where the signal takes on the entered value. Values must match the radix that is currently selected for the signal.
<div data-bbox="258 1451 354 1495" data-label="Image"> </div> <p data-bbox="245 1556 587 1591">Set number of seeks</p>	<p data-bbox="792 1394 1382 1499">Sets the number of matched values to seek for one search backward/forward operation by clicking the search icon.</p>

Simulator


The following items comprise the **Simulator** toolbar:

Icon	Description
 Start/Continue	Runs the simulation until a breakpoint is hit, the simulation finishes, or for the duration specified in the Set Continue Time dialog box or toolbar time entries.
 Run for Specified Time	Runs the simulation for the specified time, then stops.
 Stop	This icon is active when the simulation is running. Click to stop the simulation.
 Next	For VHDL, Verilog, and TB code, next steps over tasks and functions.
 Step In	Moves the simulation forward by stepping one line of code, irrespective of the language of the code. This is the same as the UCLl Step command.
 Step In Active Thread	Steps to the next executable line in the current active thread.
 Step In Any Testbench Thread	For Native Testbench (NTB) OpenVera and SystemVerilog testbenches, stops at the next executable line in the testbench.
 Step Out	Steps to the next executable line outside of the current function or task.

 Restart	Stops the currently running simulation and restarts it with the current simulation setup. This retains all open windows and GUI setups. If the simulation is not running, it starts it.
----------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------


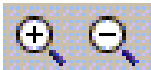
Time Operations


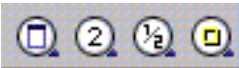
The following items comprise the **Time Operations** toolbar:

Icon	Description
 Set Time and Precision	Displays current time of the C1 cursor. Sets the current time by entering a new time in this field. Displays the time units for displaying simulation data. Select View > Set Time Scale to set the time units and precision.

Zoom

The following items comprise the **Zoom** toolbar:

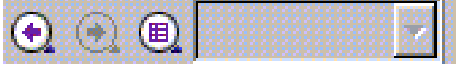

Icon	Description
 Selection Tool	<p>Used for schematic views. Changes the mouse behavior, so clicking and dragging selects objects. Clicking a single object selects it.</p> <p>Control-click adds objects to the selected set. Click and drag creates a box. Everything inside the box will be selected.</p>
 Zoom In/Out	<p>Zoom In – Makes objects in the selected area fill the pane.</p> <p>Zoom Out – Makes objects in current view fit the selected area.</p>

 Pan Tool	<p>Used for schematic views. Changes the mouse behavior, so clicking and dragging pans all the objects in the schematic view. For example, click drag down, moves all the objects down in the view.</p> <p>Choosing this menu item also changes the cursor to a hand. Click the toolbar selection arrow to change the cursor back to normal.</p>
 Open Database or File	<ul style="list-style-type: none"> •Zoom Full – Fits all viewable objects into the current view. •Zoom In 2x – Makes objects in the current view twice as big, so fewer objects will be viewable. •Zoom Out 1/2– Makes objects in the current view twice as small, so more objects will be viewable. •Zoom Fit Selection – Changes the view to center on the selected set of schematic objects, so all selected objects are visible in the current pane.

Zoom and Pan History

The following items comprise the **Zoom and Pan History** toolbar:

Icon	Description
------	-------------

 <p>Go to Zoom and Pan Settings</p>	<p>Back in Zoom and Pan History – Back arrow iterates through saved zoomed or panned views for the current pane. When you change a zoom or a pan in a view, DVE stores the previous view, so you can retrieve it.</p> <p>Forward in Zoom and Pan History – If you have gone backward in zoom/pan history, the forward arrow provides an easy way to go to the next view. Clicking this item will eventually get to the current view.</p> <p>Named Zoom and Pan Settings – Goes to a saved named view selected in the pull-down menu.</p>
<p>Zoom to cursors</p> 	<p>Zooms between markers C1 and C2 in the waveform.</p>









Using the Context-sensitive Menu

In any window, right-click to display a CSM, then select a command. The Hierarchy pane CSM is as follows:

Command	Description
Copy	Copies selected text.
Add to Waves	Displays the selected signal or signals in the Wave view.
Add to Lists	Displays the selected signal or signals in the List view.
Show Source	Displays source code in the Source view for the selected scope.
Expand By Levels	Allows expansion by multiple levels with a single action.
Expand All	Expands the entire hierarchy at once. There may be a delay getting the hierarchy from the simulation when working interactively.

Command	Description
Collapse	Collapses the selected scope.
Collapse All	Collapses all expanded scopes.
Select Scope By Levels	Allows you to select scopes by levels. You can select more than one level at a time.
Select All	Selects all scopes that are visible in the hierarchy (does not implicitly expand).

Schematic View Toolbar Options

Toolbar Command	View Menu Command	Action
	Selection Tool	Prepares the cursor for selecting objects (the default cursor).
	Zoom In Tool	Prepares the cursor for zooming in. The cursor becomes a magnifying glass. Drag a bounding box around the area to enlarge.
	Zoom Out Tool	Prepares the cursor for zooming out. The cursor becomes a magnifying glass. Drag a small box to zoom out by a large amount, or a large box to zoom out by a small amount.
	Pan Tool	Prepares the cursor for panning the window view. The cursor becomes a hand shape. Point and drag to pan the view.
	Zoom Full	Zooms out to display the entire design.
	Zoom In	Zooms in 2x.
	Zoom Out	Zooms out 2x.
	Zoom to Selection	Zooms to the area selected with the Selection Tool.

Using the Schematic View CSM

The following table describes the CSM commands:

Command	Action
Copy	Copies the selection to the clipboard.
Show Source	Displays the source code for the selection.

Show Path/ Schematic view	Opens a new Path or Schematic showing the current selection.
Add to Waves	Adds the selected signal set from the active pane and adds the signal to the default Wave view. If there is no Wave view, DVE creates one and adds the signals under the insertion bar.
Add to Lists	Adds the selected signal set from the active pane and adds the signal to the default List view. If there is no List view, DVE creates one and adds the signals under the insertion bar.
Add to Group	Selects New Group to create a new signal group. The name will be Group< <i>n</i> >, where <i>n</i> is one more than the highest numbered existing signal group. The new signal group is created at the top of the signal list.
Expand Path	Displays the path of a selected signal across boundaries in a path schematic view.
Add Fanin//Fanout	Displays the Add Fanin/Fanout to add the fanin logic to, or fanout logic from, a specified object in a currently active path.
Move Up to Parent	Displays a higher-level schematic view of the parent design in the active schematic view.
Move Down to Definition	Displays a design schematic for the selected design instance in the active schematic view.
Back>	Displays the previous schematic from the history.
Forward>	Displays the next schematic in the history.
Selection Tool	Changes the mouse behavior, so clicking and dragging selects the object. Used for schematic views.
Zoom In Tool	Changes the mouse behavior, so click zooms in. Click drag creates a box that will become the new view extents for the schematic. Used for Schematic views.
Zoom Out Tool	Changes the mouse behavior, so click zooms out. Used for Schematic views.
Pan Tool	Changes the mouse behavior, so clicking and dragging spans all the objects in the schematic view. For example, click drag down, moves all the objects down in the view. Choosing this menu item also changes the cursor to a hand. Click the toolbar selection arrow to change the cursor back to normal. Used for Schematic views.
Zoom>	Creates a submenu for all zoom operations. These menu items are only applicable for Schematic and Wave panes.
Annotate Values	Displays values at the pins.

Set Radix >	<p>Creates a submenu giving reasonable choices for radix changes for the selected signal. Changing radix on a signal is global and will change the radix wherever the signal is displayed in the debugger.</p> <p>User Defined - Creates a sub-submenu that allows you to choose user-defined types, if any, and allows you to specify user-defined types from a dialog box.</p> <p>Edit ... - Displays the user-defined radix dialog box (see “Using User-defined Radices” on page 5-9).</p>
Trace Drivers	<p>Finds the active driver of the object, shows the driver in the drivers/loads pane and shows the driver in the reusable Source view in the current window. If no window is present, DVE creates one. If a drivers pane already exists, the new trace information is added to it.</p> <p>This menu item will be active if the object selected in the currently active pane is a variable or signal. For this capability to work, the design must be compiled with one of the <code>-debug</code> options and an <code>mdb</code> library must exist.</p>
Trace Loads	Same as above, except for loads of the signal.
Drivers/Loads	Creates a dynamic submenu that allows you to navigate and manage drivers/loads displays (see Chapter 9, “Tracing Drivers and Loads”).
TraceX	Traces the X value.
Add Dump . . .	Displays the Dump Values dialog box to specify signals to dump value change information starting at the current time (see “Dumping Signal Values” on page 3-10).
Dump	Turns on value change dumping at the current time for any selected signal in the active pane.
Add Force	Displays the Force Values dialog box to allow you to interactively change the value of a signal (see “Forcing Signal Values” on page 3-11).

Driver Pane Menu

Show Active Drivers	Toggles current active driver display on.
Show All Drivers	Toggles all drivers display on.
Clear Trace	Removes the selected trace or traces from the driver pane.
Clear All Traces	Removes all information from the driver pane.
Synchronize Source view	Toggle button On (checked) means that the Source view tracks the selection from the driver pane. Selecting the signal highlights the signal port or declaration in the Source view. Selecting the driving statement highlights that statement.
Add Trace to Waves	Adds the trace information to the Wave view.
Go to Time	Advances the simulation to the specified time.
Synchronize Path Schematic	Toggle button On (checked) means that the Schematic view tracks the selection from the driver pane. Selecting the signal highlights the signal. Selecting the driving statement shows the design with the scope selected.

Using the Command Line

Use the command line to enter DVE and Unified Command-line Interface (UCLI) commands. The following table describes these commands (commands marked with an asterisk (*) are UCLI commands):

Command	Description
open_db	Opens a database file.
close_db	Closes a database file.
open_file	Opens a file.
exit	Exits the application.
Design Query:	

show*	Displays design information for a scope or nested identifier.
drivers*	Obtains driver information for a signal/variable.
loads*	Obtains load information for a signal/variable.
fanin	Extracts the fanin cone of the specified signal(s).
search	Locates design objects whose names match the name specification you provide.

Simulator:

open_sim	Setup Simulator executable and arguments.
start*	Starts tool execution.
step*	Advances the tool one statement.
next*	Advances the tool stepping over tasks and functions.
run*	Advances the tool and stop.
finish*	Allows the tool to finish then return control back to UCLI.
restart*	Restarts tool execution and keeps the setting in the last run.

Breakpoints:

stop	Adds or displays stop breakpoints.
------	------------------------------------

Navigation:

scope*	Gets or changes the current scope.
thread*	Displays thread information or moves the current thread.
stack*	Displays thread information or moves the call stack.
listing*	Displays source text.
add_schem	Shows scope in Schematic view.
add_source	Shows signal/scope in Source view.

Signal/Variable/Expression:

get*	Obtains the value of a signal/variable.
force*	Forces or deposits a value on a signal/variable.
release*	Releases a variable from the value assigned using 'force'.
call*	Executes a system task or function within the tool.
sexpr*	Evaluates an expression in the tool.
vbus*	Creates, deletes, or displays a virtual object.
add_group	Adds signals to the Group.
add_list	Adds signals to the List view.

add_mem	Adds memory to the Memory view.
add_pathschem	Shows path schematic for signal(s)/ scope(s).
add_watch	Adds signals to the Watch view.
add_wave	Adds signals to the Wave view.
delete_group	Deletes signals from the given group.
delete_list	Deletes signals from the given list view.
delete_watch	Deletes signals from the global watch view.
delete_wave	Deletes signals from the given wave view.
compare	Compares signal/scopes.
view	Opens, closes or lists view.

Signal Value and Memory:

dump*	Creates/manipulates/closes dump value change file information.
memory*	Loads/writes memory type values from/to files.
add_mem	Adds memory to the Memory view.

Session Management:

save*	Saves simulation state into a file.
restore*	Restores simulation state saved in a file.
save_session	Saves the session.
open_session	Opens a session.

Help Routines:

help	Lists basic commands. Use -all for listing all commands.
alias*	Creates an alias for a command.
unalias*	Removes one or more aliases.
config*	Displays/sets current settings for configuration variables.

Macro Control:

do*	Evaluates a macro script.
onbreak*	Specifies script to run when a macro hits a stop point.
onerror*	Specifies script to run when a macro encounters an error.
resume*	Resumes execution of a macro file.
pause*	Pauses execution of a macro file.
abort*	Aborts evaluation of a macro file.
status*	Displays the macro file stack.

Misc.:

ace*	Evaluates the analog simulator command.
cbug*	Debugs support for C, C++ and SystemC source files.
coverage*	Evaluates coverage command(s).
power	Powers measure.
quit*	Exits the application.
senv*	Displays one or all <code>synopsys : : env</code> array elements.
setenv*	Sets the value of a system environment variable.
sn*	Displays the Specman prompt when used without arguments, and executes e code commands when they are entered as optional arguments.
tcheck*	Disables/enables timing check upon a specified instance/port at runtime.
virtual	Creates, deletes or displays a virtual object.

Index

A

- About DVE(Help menu selection) A-19
- Add Fanin/Fanout A-15
- Add to Lists 7-6, A-11
- Add to Waves 7-6, A-11
- Annotate Values A-15
- Assertion Attempts A-16
- Assertion Failure Summary Pane 8-5
- Assertion failures 8-5
- Assertion Unit 3-4
- Assertion Window 8-1

B

- Back A-14
- Beginning (menu selection) A-8
- binary radix 5-8
- bits, displaying in waveform 5-9
- Breakpoints A-10
- Building Buses 5-30
- Bus Builder 5-30
- buses, building 5-30

C

- c code simulation 1-9

- C1 cursor 5-8
- Capture Delta Cycle Values 7-7, A-10
- Close Database (File menu selection)
reference A-2
- Close File (File menu selection)
reference A-2
- Close Window (File menu selection)
reference A-3
- closing a VPD database 1-19
- column headings
rearranging 3-9, 3-18
- Compare A-13
- compile-time options 1-2
- context-sensitive menus, using A-33
- Continue 1-14, A-9
- coverage databases 1-4
- cursor C1 5-19
- cursor C2 5-20
- cursors, inserting 5-19

D

- database
 - closing 1-19
 - opening 1-10
- debug, option 1-3
- debug_all, option 1-3

- debug_pp, option 1-3
- debugging options 1-2
- Design Root icon 3-4
- Display Signal Groups A-11
- displaying
 - scope data 3-16
- dividers, signal 5-11
- Dock
 - Window menu selection A-19
- dock and undock windows 2-6
- drag zooming 5-25
- Dump Values 7-7, A-10
- duplicate signals, displaying 5-11
- DVE
 - exiting 1-19
 - starting 2-7
- DVE Help (Help menu selection) A-19
- DVE version 1-2

E

- Edit Bus A-12
- Edit menu
 - A-3
- Edit menu, reference A-3
- Edit Parent A-15
- Edit Source A-15
- Edit User-Defined Radices A-13
- End (View>Go To menu selection)
 - reference A-8
- Execute Tcl Script (File menu selection)
 - reference A-2
- Exit (File menu selection)
 - reference A-3
- exitingDVE 1-19
- Expand Path A-15

F

- File menu, reference A-2

- filee, required 1-3
- Find (Edit menu selection)
 - reference A-4
- Forward A-14

G

- Go To (View menu selection)
 - reference A-8
- gui, option 1-6

H

- hexadecimal radix 5-8

I

- icons
 - Design Root 3-4
 - Verilog Function 3-4
 - Verilog Instance 3-4
 - Verilog Named Begin 3-4
 - Verilog Named Fork 3-4
 - Verilog Task 3-4
- inserting new markers 5-24
- integers, storing 5-9
- interactive options 1-6
- interval between cursors 5-20
- invoking DVE 2-7

L

- List Window A-17, A-18
 - customizing 6-7
 - panes 6-2
 - save format 6-13
 - set markers 6-6
 - set signal properties 6-9
 - using 6-1
 - view simulation data 6-5
- Load Session (File menu selection)

- reference A-2
- loading a VPD database 1-10
- log files 1-7
- lower timescale 5-15

M

- Main Window
 - example 2-3
 - using 2-1
- Markers dialog box 5-21, 5-24
- markers, inserting 5-19
- Menu Bar
 - reference A-1
 - using 2-7
- Mixed Simulation 1-8
- mixed Verilog/VHDL 1-9
- Move Down to Definition A-14
- Move Up to Parent A-14

N

- Name column (signal pane) 5-8
- new markers, inserting 5-24
- nogui, option 1-6

O

- Open Database
 - Toolbar icon A-25
- Open Database (File menu selection)
 - reference A-2
- Open Database dialog box 1-10
- Open File (File menu selection)
 - reference A-2
- opening
 - database 1-10
- OVA library 1-4

P

- post-processing, options 1-6

Q

- quitting DVE 1-19

R

- radix
 - binary 5-8
 - hexadecimal 5-8
- radix, user-defined 5-13
- rearranging column headings 3-9, 3-18
- Reload Database A-2
- Required Files 1-3
- Restore State A-10
- Run A-9
- Run Example (Help menu selection) A-19

S

- Save Session (File menu selection)
 - reference A-2
- Save State A-10
- scalar signals 5-8
- schematic views 7-2
- Scope Types
 - example 3-3
- Scopes
 - example 3-3
- script, running 1-7
- Search Backward A-12
- Search Forward A-12
- searching in the Waveform pane 5-26
- selecting
 - scopes 3-16
- Set Expression A-12
- Set Precision text field A-31

- Set Radix 7-7, A-13
- Set Search Constant A-12
- set signal properties 6-9
- Set Time text field A-31
- Setup A-9
- Shift Time A-13
- Show A-15
- Show Comparison Info A-13
- Show Memory A-11
- Show Path Schematic A-14
- Show Scehmatic A-14
- Show Source A-14
- signal dividers, adding 5-11
- Signal Menu A-11
- Signal Properties A-13
- signals
 - scalar 5-8
 - vector 5-8
- simulation time
 - setting
 - example 2-9
- Source view
 - Toolbar icon 2-5, 2-6, A-27
- starting DVE 2-7
- Step 1-15, A-9, A-10
- Stop 1-14, A-9

T

- Terminate A-10
- time
 - simulation
 - setting 2-9
- time data type 5-9
- Time... (View>Go To menu selection)
 - reference A-8
- Tips button 5-25
- Toolbar
 - using 2-7

- Toolbars A-9
- Trace Drivers 7-7, A-16
- Trace Loads 7-7, A-16

U

- ucli, option 1-6
- Undock A-19
- undock windows 2-6
- upper timescale 5-15

V

- Value column (Signal pane) 5-8
- value transitions 5-14
- VCS 1-8
- vector signals 5-8
- Verilog Function icon 3-4
- Verilog Instance icon 3-4
- Verilog Named Begin icon 3-4
- Verilog Named Fork icon 3-4
- Verilog Task icon 3-4
- version 1-2
- version, check for 1-2
- VHDL 1-8
- View menu, reference A-5
- VPD file 1-4
 - closing 1-19
 - loading 1-10

W

- Waveform pane
 - context sensitive menu 5-22
 - cursors 5-20
- Waveform pane, using 5-1
- Window Menu
 - reference A-17

Z
zooming

by dragging 5-25

