cadence™

SimVision User Guide

Product Version 8.2
November 2008
June 2009

Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Contents

Preface 17
Related Documents 18
About Videos 18

1
Introduction 19
Using SimVision 21
Managing Windows 23
Opening and Closing Windows 23
Iconifying and Activating Windows 24
Tiling Windows 25
Renaming Windows 25
Using the Console Window 25
Managing Toolbars 27
Using the Sidebar 28
Managing Time in the SimVision Windows 30
Managing Simulation Objects 31
Selecting and Deselecting Objects 32
Using Pop-Up Menus 32
Adding Objects to a Window 33
Using Keyboard Shortcuts and Hotkeys 33
Using the SimVision Command Language 34
Aborting a SimVision Action 34
Getting Help 34

2
Invoking SimVision 37
Preparing Your Design for Simulation 37
Invoking the Simulator with SimVision 38

Disconnecting and Terminating the Simulation . . . 42
Disconnecting from a Simulation . . . 42
Terminating and Disconnecting from a Simulation . . . 43
Terminating and Post-Processing . . . 43
Invoking SimVision Separately . . . 44
Connecting to a Simulation . . . 47
Opening a Simulation Database . . . 48
Invoking SimVision in Post-Processing Mode . . . 50
Making Simulation Processes Secure . . . 51

3
Saving and Restoring Your Debugging Environment . . . 53
The Simulator and SimVision Environments . . . 53
Using Automatic Save and Restore . . . 54
Saving and Restoring a Command Script for Simulator Connection . . . 56
Saving and Restoring a Command Script for Post-Processing . . . 57
Examining the Contents of a Command Script . . . 59
The Simulator Command Script . . . 60
The SimVision Command Script . . . 60
Executing SimVision Commands at Startup . . . 61

4
Accessing the Design Source Code . . . 63
Opening a Source Browser Window . . . 63
Navigating the Design Hierarchy . . . 65
Opening a File in the Source Browser . . . 65
Selecting Text in the Source Browser . . . 66
Searching a Source File . . . 66
Accessing Design Objects and Values . . . 68
Displaying ‘define Macros in the Source Browser . . . 69
Performing Functions on Design Objects . . . 69
Setting Breakpoints . . . 71
Setting Probes . . . 73
Running the Simulation . . . 74
Viewing the Call Stack . . . 74

Finding the Cause of a Signal Transition . . . 76
Using Bookmarks in the Source Browser . . . 77
Editing a Source File . . . 78

5
Accessing Design Objects . . . 81
Using the Design Browser Sidebar . . . 81
Selecting a Design Hierarchy to View . . . 85
Setting the Root of the Scope View . . . 86
Expanding and Collapsing a Scope . . . 86
Locking and Unlocking a Scope . . . 87
Filtering the Scope View . . . 87
Searching for Scopes . . . 88
Selecting Scopes . . . 89
Displaying Objects in Selected Scopes . . . 89
Setting Scope View Options . . . 90
Setting Signal List Options . . . 91
Using the Design Search Sidebar . . . 92
Searching for Objects . . . 93
Displaying the Search Results . . . 95

6
Monitoring Signal Values . . . 97
Opening a Design Browser Window . . . 97
Displaying Signals and Variables . . . 99
Sorting the Signal List . . . 100
Choosing the Format of Signal Names . . . 100
Choosing the Radix for Signal Values . . . 101
Justifying the Signal Values . . . 101
Filtering Signals in the Signal List . . . 101
Viewing Aggregate Signals . . . 102
Expanding and Collapsing Aggregates . . . 103
Splitting a Signal . . . 104
Creating a Scrollable Region . . . 105
Making the Design Browser Window Compact . . . 106

Using Bookmarks in the Design Browser . . . 106
Setting Design Browser Preferences . . . 108

7
Managing Simulation Databases . . . 109
Creating an SST2 Database . . . 110
Creating a Database with the Incisive simulator . . . 111
Creating a Database with Verilog-XL . . . 112
Opening an SST2 Database . . . 113
Converting a Database to SST2 Format . . . 113
Exporting a Database . . . 115
Using the Batch Database Translation Utility . . . 117
simvisdbutil Command Syntax . . . 117
Generating a Database That Contains a Range of Times . . . 120
Generating a Database for a Specific Set of Signals . . . 120
Reloading a Database . . . 121
Renaming a Database . . . 121
Displaying Information about Databases . . . 121
Displaying Information about Databases Created during Simulation . . . 121
Displaying Information about Databases Loaded into SimVision . . . 122
Closing a Database . . . 123

8
Creating and Managing Probes . . . 125
Setting a Probe . . . 125
Setting a Probe with the Incisive Simulator . . . 125
Setting a Probe with Verilog-XL . . . 126
Managing Probes in the Properties Window . . . 127

9
Setting and Managing Breakpoints . . . 129
Setting a Time Breakpoint . . . 129
Setting a Time Breakpoint with the Incisive Simulator . . . 129
Setting a Time Breakpoint with Verilog-XL . . . 131

Setting a Line Breakpoint 132
Setting a Line Breakpoint with the Incisive Simulator 133
Setting a Line Breakpoint with Verilog-XL 133
Setting a Signal Breakpoint 133
Setting a Signal Breakpoint with the Incisive Simulator 134
Setting a Signal Breakpoint with Verilog-XL 134
Setting a Condition Breakpoint 135
Setting a Process Breakpoint for VHDL 136
Setting a Subprogram Breakpoint 136
Breakpoint Options 137
Managing Breakpoints in the Properties Window 138

10
Changing and Monitoring the Value of an Object during Simulation 139
Forcing and Releasing a Signal Value 139
Depositing a Signal Value 141
Creating, Replacing, and Deleting a VHDL Run-Time Driver 143
Monitoring Signal Value Changes in Verilog-XL 143

11
Controlling the Simulation 145
Running the Simulation 145
Resetting the Simulation 147
Reinvoking the Simulation 147
Saving and Restarting a Simulation Checkpoint 148
Saving a Simulation Checkpoint 148
Restarting during the Same Simulation Session 149
Restarting from a New Simulator Session 149
Creating and Deleting an Alias in the Incisive Simulator 150
Setting Variables in the Incisive Simulator 150

12
Debugging at the Delta Cycle Level 153
How a Delta Cycle Executes 153
Opening the Simulation Cycle Debugger 155
Running the Simulation at the Delta Cycle Level 157
Viewing the Source Code for an Event 157
Printing and Saving the Simulation Cycle Debugger Window 157

13
Debugging API Applications 159
Preparing to Debug an API Application 159
Using the Native-Mode GDB Debugger 160
Starting GDB 161
Viewing Source Code 162
Setting Breakpoints in the API Code 162
Setting Watchpoints 164
Passing Control between the Simulator and GDB 164
Viewing the SystemC/C++/C Call Stack 165
Viewing API Function Parameters and Local Variables 166
Handling Signals When Debugging 167

14
Viewing a Design Schematic 169
Opening a Schematic Tracer Window 169
Adding Objects to the Schematic 170
Adding Details of a Scope 170
Selecting Objects in the Schematic 171
Removing Scopes from the Schematic 172
Using Color in the Schematic Diagram 172
Displaying Signal Names 172
Displaying Signal Values 173
Setting the Radix of Values 174
Zooming the Schematic 174
Using the Panner 175

Searching for Objects . . . 175
Tracing Paths with the Schematic Tracer . . . 176
Tracing Paths from Point to Point . . . 177
Tracing the Path between Two Pins . . . 178
Tracing a Path between a Pin and a Scope . . . 178
Tracing the Path between a Pin and a Cell . . . 179
Using Bookmarks in the Schematic Tracer . . . 180
Printing and Saving the Schematic Window . . . 181

15
Controlling the Appearance of Schematic Elements . . . 183
Blocks . . . 183
SystemVerilog Interfaces and Modports . . . 184
Expressions . . . 185
Abstracted RTL Elements . . . 186
Gate Primitives . . . 188
Source . . . 188
Creating a Cell Map File . . . 189
Format of a Cell Map File . . . 189
Creating a Cell Map Template . . . 192
Translating a Liberty File into Cell Map File . . . 193
Viewing Mapped Cells in the Schematic . . . 194
Handling Errors in the Cell Map File . . . 195
Updating a Cell Map File . . . 196

16
Tracing Paths with the Trace Signals Sidebar . . . 197
Accessing the Trace Signals Sidebar . . . 197
Tracing the Loading Logic . . . 198
Tracing the Driving Logic . . . 200
Tracing Active Drivers . . . 201
Tracing X Values . . . 202
Tracing an X Value with the Trace Signal Buttons . . . 202
Tracing an X Value with the Trace X Buttons . . . 203
Shifting Time in the Trace Path . . . 204

Sending Signals to the Containing Window 204
Using the Trace Signals Sidebar with the Schematic Tracer 206
Performing Functions on Objects in the Sidebar 206

17
Displaying Waveforms 209
Opening a Waveform Window 209
The Signal List 210
The Waveform Area 211
Maximizing the Waveform Viewing Area 211
Default Waveform Shapes and Color 212
Adding Signals to the Waveform Window 213
Probing Signals in the Waveform Window 213
Adding Contributing Signals to the Waveform Window 214
Displaying Aggregate Signals 214
Expanding and Collapsing Signals 215
Splitting a Signal 215
Creating a Scrollable Region 216
Selecting the Format of Signal Names 216
Changing a Signal Name 217
Setting the Radix of Signal Values 218
Using Mnemonic Maps 218
Creating a Mnemonic Map 219
Applying a Mnemonic Map 220
Using a Mnemonic Map to Display Booleans as Logic Signals 220
Adding Comments to the Waveform Window 221
Rearranging Objects in the Waveform Window 222
Removing Objects from the Waveform Window 223
Zooming the Waveform Data 223
Zooming with the Scroll Bar 223
Zooming with the Mouse 224
Zooming with Toolbar Buttons and View Menu Choices 225
Searching the Signal List and Waveform Area 226
Time-Shifting Waveform Data 228
Defining and Displaying a Grid 228

Turning the Grid On and Off . . . 228
Defining the Grid . . . 228
Saving and Printing Waveforms . . . 229

18
Organizing Signals—Groups, Buses, Conditions, Virtual Signals, and Comparisons . . . 231
Working with Groups . . . 231
Creating a Group in the Waveform Window . . . 231
Creating a Group in the Properties Window . . . 232
Creating Copies and Instances of Groups . . . 233
Splitting and Deleting a Group . . . 234
Performing Other Operations on Groups . . . 234
Working with Buses . . . 235
Creating a Bus from Selected Signals . . . 235
Creating a Bus from an Array Range . . . 235
Naming a Bus . . . 236
Splitting a Bus . . . 236
Moving a Bus . . . 237
Deleting a Bus . . . 237
Working with Conditions and Virtual Signals . . . 237
Creating an Expression . . . 238
Adding an Expression to the Waveform Window . . . 240
Searching for an Expression . . . 240
Editing an Expression in the Expression Calculator . . . 241
Accessing an Expression in the Properties Window . . . 241
Comparing Waveforms . . . 242

19
Managing Time in the Waveform Window . . . 243
Using Cursors and the Baseline . . . 243
Synchronizing the Primary Cursor with the Simulation or Database . . . 245
Creating a Cursor . . . 246
Changing a Cursor Name . . . 246

Choosing a Primary Cursor . . . 246
Setting the Location of a Cursor . . . 246
Tracking Signal Changes with the Primary Cursor . . . 247
Finding a Cursor . . . 248
Linking a Waveform Window to a Cursor . . . 248
Using Markers . . . 249
Creating a Marker . . . 249
Changing the Name of a Marker . . . 249
Changing the Location of a Marker . . . 250
Changing the Color of a Marker . . . 250
Finding a Marker . . . 251
Linking a Waveform Window to a Marker . . . 251
Locking and Unlocking a Marker . . . 251
Measuring Time in the Cursor Delta Area . . . 252
Using Time Ranges . . . 253
Saving a Time Range . . . 253
Linking Waveform Windows to a Time Range . . . 254
Managing Time Ranges in the Properties Window . . . 254
Viewing Events in Sequence Time . . . 255

20
Debugging Memories . . . 259

Opening a Memory Viewer Window . . . 260
Adding Memories to the Window . . . 261
Color Coding Memory Value Changes . . . 262
Viewing a Range of Memory Cells . . . 262
Changing the Order of Memory Addresses . . . 263
Changing the Radix of Addresses and Cell Values . . . 263
Changing the Number of Columns . . . 263
Searching for a Value in Memory . . . 264
Going to a Memory Cell . . . 264
Viewing a Memory during Simulation . . . 264
Probing Memories . . . 265
Setting Breakpoints . . . 265
Setting the Value of a Memory Cell . . . 265

Saving and Restoring a Memory State 266
Printing a Memory 268

21
Viewing Analog Data 269
Displaying Analog Signals in the Waveform Window 269
Displaying Real Values with Full Precision 272
Displaying wrealXState and wrealZState Values 272
Creating an Overlay Group 273
Creating a Bus with Analog Signals 274
Changing the Colors of Waveforms 275
Changing the Format of the Waveform Lines 275
Displaying a Y-Axis Grid 276
Creating a Reference Line 277
Setting the Scale 277

22
Viewing Transactions 279
Viewing Transaction Streams 279
Finding Streams using the Design Search Sidebar Tab 280
Finding Streams in the Design Browser Signal List 280
Viewing Transaction Information 281
Viewing Error Information in the Waveform Window 282
Viewing Attributes 283
Viewing Overlapping Transactions 284
Changing the Stacking Order 284
Filtering Streams 286
Viewing Transactions with a Given Attribute or Attribute Value 286
Expanding Streams by Depth 287
Filtering a Stream Based on a Condition 288
The streamoverlay Function 290
The format, match, and rematch Functions 290
Using the Transaction Explorer Toolbar 293

23
Creating Custom Views of Simulation Data 295
Opening a Register Window 295
Managing Register Pages 296
Using the Target Icon in Register Windows 297
Adding Objects to a Register Page 297
Display Format of Signals and Variables 298
Changing the Radix of a Signal 298
Shifting a Signal in Time 298
Annotating Register Pages 299
Changing the Appearance of Objects 300
Grouping and Ungrouping Objects 301
Navigating through Simulation Time 301
Printing and Saving Register Pages 302

24
Measuring Signal Values 303
Opening a Measurement Window 303
Setting the Location of the Cursors 304
Moving Rows and Columns 305
Sorting the Contents of the Window 305
Setting Rise/Fall Time Parameters 306
Saving Measurements 306

25
Setting Preferences 307
General Options 308
Signal Options 309
VHDL Signal Options 310
Verilog Signal Options 310
Verilog AMS Signal Options 311
Keyboard Shortcuts 311
Toolnet Cross-Probing 312
Simulation Settings 313

Waveform Window . . . 313
Waveform Display . . . 315
Analog Waveform . . . 315
Waveform Keyboard Shortcuts . . . 316
Design Browser . . . 316
Scope View . . . 317
Signal List . . . 318
Source Browser . . . 318
Source Browser Colors . . . 319
Source Browser Files . . . 320
Measurement Window . . . 320
Schematic Tracer . . . 321
Schematic Tracer Colors . . . 322
Memory Viewer . . . 323
Memory Viewer Colors . . . 323
Simulation Cycle Debug . . . 324
Transaction Explorer . . . 324

26
Customizing Toolbars . . . 325
Adding and Removing Toolbars and Buttons . . . 325
Creating a Custom Toolbar . . . 326
Adding a Custom Toolbar to Other Windows . . . 329
Writing a Window-Independent Script . . . 329
Modifying Custom Toolbars . . . 330

Index . . . 331

Preface

The SimVision analysis environment provides debugging features for simulation and formal analysis. It provides graphical control of the Incisive™ Unified Simulator, Verilog-XL, and Incisive Formal Verifier.

SimVision includes the following components:

- Properties window
- Design Browser
- Waveform window
- Source Browser
- Schematic Tracer
- Register window
- Expression Calculator
- Measurement window
- Assertion Browser
- Memory Viewer
- Simulation Cycle Debugger
- API Debugger
- Console window
- SimCompare Manager window

The SimVision analysis environment also includes these associated tools:

- SimCompare, for strobed or filtered comparison between two simulation waveform databases.
- NCLaunch, for compiling designs and specifying simulation invocation options.

- NCBrowse, for sorting and filtering messages from the simulator tools.

Related Documents

For more information about SimVision:

- *Incisive Simulator Tutorial with SimVision*

  This document describes how to use the NC Launch and SimVision to simulate and debug a mixed-language design.

- *SimVision Command Reference*

  This document describes how to use SimVision command language to perform SimVision functions in Tcl scripts and plug-in applications.

- *What's New in Incisive Simulator*

  This document describes the features of all Incisive simulator products, including SimVision.

- *SimVision Known Problems and Solutions*

  This document describes any problems with SimVision that were known at the time the product was released, and any workarounds or solutions to those problems.

For information about the Cadence simulators:

- *NC-Verilog Simulator Help*, *NC-VHDL Simulator Help*, *NC-SC Simulator User Guide, NC-SC Simulator Reference, SystemVerilog Reference, SystemVerilog in Simuation,* and *Verilog-XL User Guide*

About Videos

Throughout this book, you will find links to videos that demonstrate the features described in the document. Click on the link to run the video.

To run these videos, you need an Adobe Flash player, which you can download for free from http://www.adobe.com/downloads/.

1

Introduction

SimVision is a unified graphical debugging environment for Cadence simulators. You can use SimVision to debug digital, analog, or mixed-signal designs written in Verilog, SystemVerilog, VHDL, SystemC®, or a combination of those languages.

SimVision is made up of several tools. The following tools are available as toolbar buttons and as choices on menus:

The Properties window lets you manage the cursors, markers, expressions, and other debugging objects that you have created during the SimVision session.

The Design Browser window lets you monitor the signals and variables in the design.

The Waveform window plots simulation data along an X and a Y axis. Data is usually shown as signal values versus time, but it can be any recorded data.

The Source Browser gives you access to the design source code.

The Schematic Tracer displays a design as a schematic diagram and lets you trace a signal through the design.

The Memory Viewer lets you observe changes in the internal state of memory locations. During simulation, it also lets you set breakpoints, and force and deposit values to memory locations.

The Register window lets you use a free-form graphics editor to define any number of register pages, each containing a custom view of the simulation data.

The Expression Calculator lets you define expressions, which combine signals to form buses, conditions, and virtual signals.

Other buttons might appear in your SimVision windows, depending on the tools that you have installed on your system.

The following tools are available only as choices on the *Windows* menu:

New – Measurement	The Measurement window lets you display analog measurements, such as the peak-to-peak ratio or root square mean value of selected signals.
New – Assertion Browser	The Assertion Browser lets you access any assertions that are present in your design. An icon for this window appears whenever assertions are compiled into the design. For information on assertions, see the *Simulation-Based Assertion Checking Guide*.
Tools – Simulation Cycle Debug	The Simulation Cycle Debugger lets you step through a simulation cycle, stopping at each time point, delta cycle, simulation phase, or scheduled process. It is not available in Verilog-XL or AMS Designer.
Tools – Console	The Console window lets you enter simulator commands, SimVision commands, or Tcl commands.
Tools – SimCompare Manager	The SimCompare Manager window lets you perform complex comparisons of signal transitions in simulation databases. This window, plus the Comparisons sidebar, are the graphical user interface to the SimCompare Tool. For more information on SimCompare, see the *SimCompare User Manual and Reference*.

Other tools may appear in your menus, depending on the tools that you have installed on your system.

A Properties window lets you create and manage the following types of objects:

Markers	Flags that you can position at any point in simulation time.
Cursors	A cursor is like a marker, but you can use it to sample signal values, and you can use it together with a baseline to measure simulation times.
Databases	Simulation data saved in SST2 format. Although data accessed by SimVision is in SST2 format, you can supply databases in the following formats: VCD, HSPICE list, HSPICE transient output, Nutmeg, Epic, and Qsim.

Groups	Sets of objects that you want to treat as a unit.
Expressions	Expressions, which perform arithmetic or logical operations on one or more objects.
Mnemonic Maps	User-defined strings, attributes, and icons for displaying simulation values.
Time Ranges	Ranges of time that you can view in the Waveform window. You can name and save time ranges, and link multiple Waveform windows to view the same time ranges.
Bookmarks	A saved state of a Source Browser, Design Browser, or Schematic Tracer window. SimVision saves the bookmarks that you create, so that you can return to the saved state at any time.
Simulator	Breakpoints, probes, databases, variables, aliases, and forces associated with a running simulation.

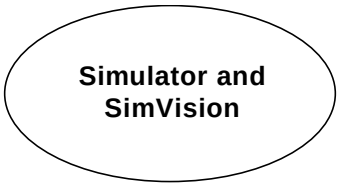
To access these objects in the Properties window:

➤ Click *Properties*,  , or choose a property type from the *Windows – Tools* menu.

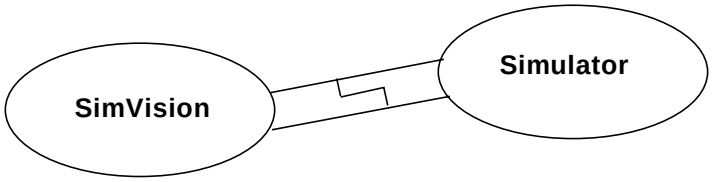
Using SimVision

There are several ways to simulate your design and debug it using SimVision:

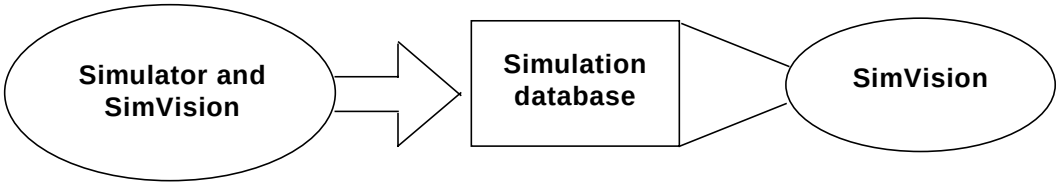
■ Invoke the simulator with SimVision. Using this method, you control the simulation with the full set of debugging tools. For example, you can set breakpoints, force and release signals, and view waveforms as they are generated during the simulation.



- Invoke SimVision and connect to a simulation that is running on the same system or over the network. Using this method, you have the same debugging features that you would have if you invoked the simulator with SimVision.

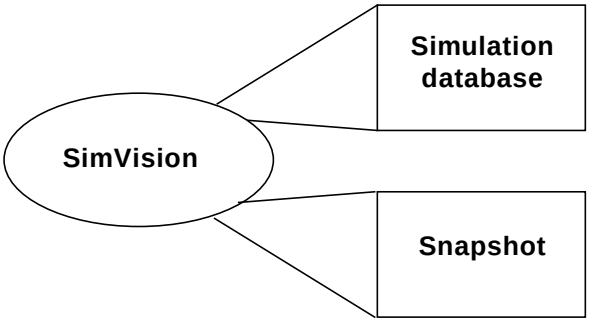


- Run the simulator with SimVision and probe signals to a database, then terminate the simulator connection and debug the design data in the SimVision post-processing mode. Using this method, you have access to all of the SimVision tools, but you have no simulation control. For example, you can view waveforms but you cannot set breakpoints or force and release signals.



Post-processing mode lets you analyze digital simulation results stored in a database. You cannot use post-processing mode to analyze mixed-signal results.

- Invoke SimVision in post-processing mode and open an existing simulation database. If you need to use the Source Browser, Simulation Cycle Debugger, or Schematic Tracer, SimVision loads the snapshot associated with the database. This gives you access to the design connectivity, but you do not have simulation control. That is, you cannot run the simulation or set breakpoints and probes.



See Chapter 2, “Invoking SimVision” for more information.

Managing Windows

SimVision lets you create multiple instances of some windows, such as the Waveform window and Register window. You can create a new (empty) window, which you populate with signals and variables, or you can create a replica (clone) of an existing window, which contains the same signals and variables as the original window.

To create a window:

➤ Choose *Windows – New* or *File – New* from the window's menu bar, then choose the type of window that you want to create.

To create a clone of a window:

➤ Choose *File – New – Clone* from the menu bar.

When you have multiple instances of a window type, one window of that type is the target. The target window is the one to which any operation is applied. For example, if you select objects in one window, you can add those objects to the target window.

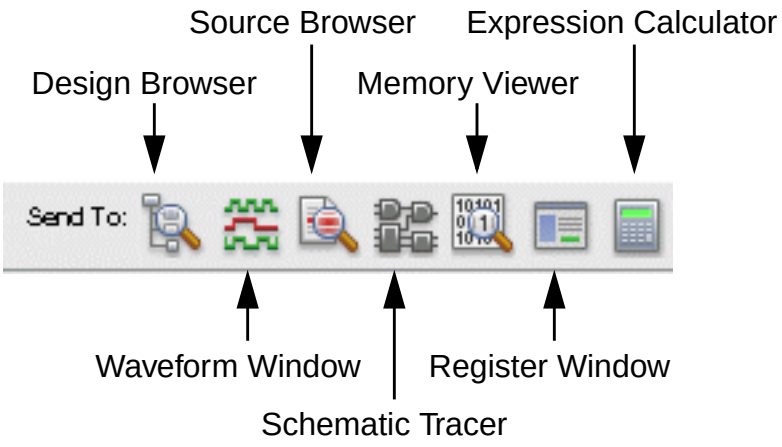
To make the window the target:

➤ In the bottom left corner of each window, enable the *Target* icon, .

Opening and Closing Windows

You can access many SimVision windows with buttons on the Send To toolbar, shown in Figure 1-1 on page 23.

Figure 1-1 Send To Toolbar



The window buttons have drop-down menus to help you manage multiple windows. You see these menus by hovering the cursor over the button for a few seconds. The drop-down menu contains the following choices:

- *Send to target*—Adds any selected objects to the target window. If no window of that type exists, SimVision creates a new one. If no objects are selected, it either brings the current target window to the foreground or creates a target window of that type.
- *Send to new*—Creates a window and adds the selected objects to that window. If no objects are selected, SimVision creates an empty window.
- *Window name*—All of the windows that you create are listed in the menu, with the target window in bold letters. Choose a window from the menu, and SimVision adds any selected objects to that window.

From any window, you can open any other SimVision window using the following menu choices:

- *Windows – New* and *File – New* create a window of the type you choose. If you select objects before creating the window, those objects are added to the new window.
- *Windows – Tools* lets you access the specified tab of the Properties window. The Properties window lets you manage your debugging environment, including markers, cursors, mnemonic maps, and simulation properties, such as breakpoints and probes.
- *Window Name*—The menu lists all windows that are currently open and lets you select a window from the list to bring the window to the foreground and give it focus.
- *Windows – Windows* opens the Windows form. This form lists the windows that are currently open. You can select one or more windows from the list, then activate, iconify, or close the window. This form also lets you lay out the windows, as described in "Tiling Windows" on page 25.

To close a window:

➤ Choose *File – Close Window* from the window's menu bar.

Iconifying and Activating Windows

To iconify an individual window:

➤ Click the *Iconify* button in the window's title bar, or select the window in the Windows form and click *Iconify*.

When the Windows form is open, you can activate an individual window by selecting it in the list of windows and then clicking *Activate*.

To iconify all SimVision windows into a single SimVision icon:

➤ Choose *Windows – Iconify All* from any SimVision window.

Tiling Windows

By default, SimVision windows are different sizes, and they appear in different parts of your terminal display. You can move windows around in any way you like. However, if you create many windows, you may find it difficult to lay them out so that they are easy to find.

SimVision gives you the following layout options for your windows:

- *Windows – Tile Horizontally* lays out the windows so that each spans the terminal display horizontally. The windows are of equal height and fill the display from top to bottom. This option makes each window the same height and width.
- *Windows – Tile Vertically* lays out the windows so that each spans the terminal display vertically. The windows are of equal width and fill the display from left to right. This option makes each window the same height and width.
- *Windows – Cascade* lays each window on top of another, offset from the upper left corner of each window. This option does not change the size of the windows.

Renaming Windows

SimVision gives every window a default name, such as *Waveform 1* or *Design Browser 1*.

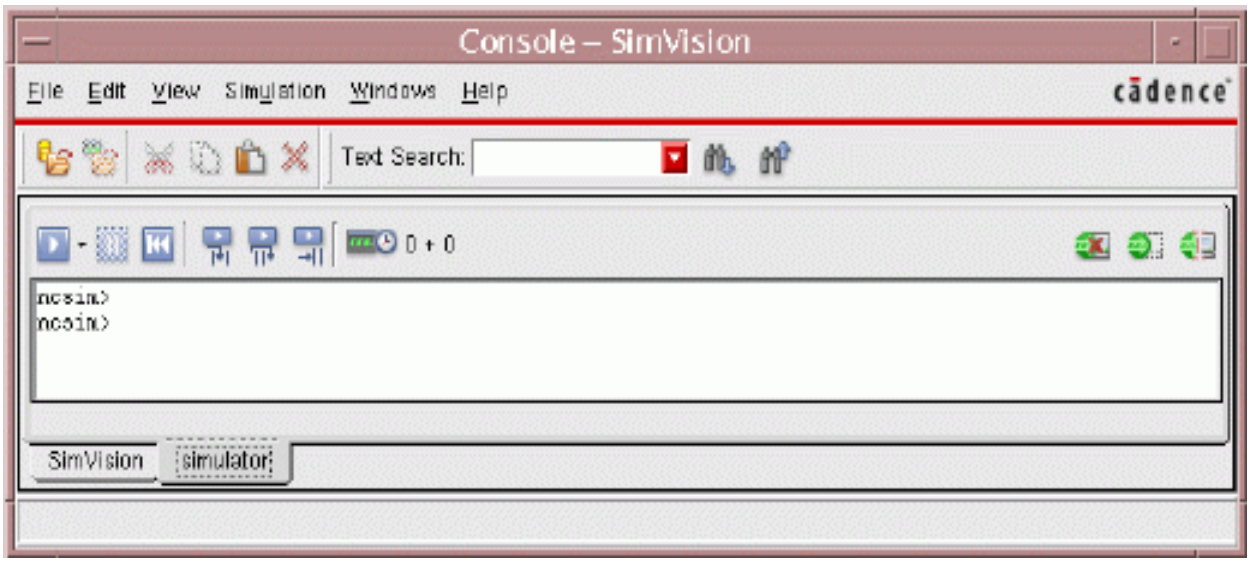
To rename a window:

1. Choose *File – Rename Window* from the menu bar of the window you want to rename. SimVision opens the Rename Window form.
2. Enter the name in the *New Name* field and click OK.

Using the Console Window

The Console window gives you access to the command languages for SimVision and for any simulator you are running. Each tool is represented by a tab in the Console window. For example, if you run SimVision with the simulator, the Console window contains a *SimVision* tab and a *simulator* tab, as shown in Figure 1-2 on page 26. If you are debugging a SystemC or API application, the Console window also contains a tab for the GDB debugger.

Figure 1-2 Console Window with the SimVision and Simulator Tabs



You can enter commands at the Console window prompt and monitor the messages returned by the tool.

To search for a text string within the Console window:

➤ Enter a search string in the *Text Search* field, and click *Search Up*, , or *Search Down*, .

The search string can include either of the following special characters:

*	Match any number of characters
?	Match a single character

Because SimVision keeps a history of the search strings that you enter, you can enter a partial string and press Tab. If the history contains one matching string, SimVision seeds the *Search* field with that string. If the history contains more than one matching string, it displays a list of those strings, and you can choose one.

If you need more flexibility when searching for text strings or lines, use the *Edit* menu.

To search for a text string from the *Edit* menu:

1. Choose *Edit – Text Search* from the Console menu. SimVision opens the Text Search form.
2. Enter a search string in the *Find what* field, including the special characters, * and ?.

The Text Search form also lets you specify the following options:

- Disable *Regular Expression* to specify the exact string you want to find. When this button is disabled, special characters (* and ?) are treated like ordinary characters.
- Enable *Match Case* if you want the search to be case-sensitive.
- Enable *Up* or *Down* to control the direction of the search.

3. Click *Find Next* to find each occurrence.
4. Press *Close* to end the search.

If you want to select all of the text in the Console window, for example, to copy and paste the results of a debugging session into a text file, choose *Edit – Select All* from the menu bar or choose *Select All* from the pop-up menu.

For information about the commands that you can enter in the *SimVision* tab, see the *SimVision Command Language Reference*. For information about the commands that you can enter in the *simulator* tab, see the *Help* for your simulator.

Managing Toolbars

Many common menu functions are also available through buttons on toolbars. The toolbars that appear in a window differ from one window type to another. For example, the Design Browser window contains the following toolbars:

Send to	The Send To toolbar, which invokes the SimVision tools, such as the Waveform window, Souce Browser, and Schematic Tracer
Standard	The standard toolbar, which performs functions such as opening a database, and cutting and pasting objects
User Toolbar	User-defined toolbars and buttons

The Waveform window contains those toolbars, plus the following toolbars:

Cursor Control	The cursor control toolbar, which lets you choose a primary cursor and specify its location
Signal List Search	The signal list search toolbar, which lets you search for signals and groups

Search Times	The time search toolbar, which lets you search for values, rising edges, falling edges, expressions, markers, and assertion states
Zoom	The zoom toolbar buttons

Because toolbars take up space that you might want to use for displaying data, you can choose which toolbars you want to display in a window and remove those toolbars that you do not use.

To choose the toolbars that you want to display:

1. Choose *View – Toolbars* or press the right mouse button over any blank area of a toolbar to display a menu of toolbars that are available in the window.
2. Enable the toolbars you want to display; disable the toolbars you want to remove from the window.

You can move a toolbar to any position within the toolbar area, between the menu bar and the data area.

To move a toolbar:

➤ Place the mouse over the small grip on the left side of the toolbar, and drag the toolbar to its new location.

Changing the appearance of toolbars in one window does not change their appearance in any other windows that are already open. However, these settings are applied to any new windows that you open.

SimVision also lets you add and remove buttons from the SimVision toolbars, and it lets you create your own toolbars and buttons. For more information, see Chapter 26, “Customizing Toolbars.”

Using the Sidebar

Many windows contain a sidebar, which you use to access objects in your design. The sidebar can contain the following tabs, depending on the window type and language in which the design is written:

The Design Browser sidebar lets you access the scopes in the design hierarchy. For information on the Design Browser sidebar, see “Using the Design Browser Sidebar” on page 81.

The Design Search sidebar lets you search for objects in all open simulations and databases, without regard to the design hierarchy. For information on the Design Search sidebar, see “Using the Design Search Sidebar” on page 92.

The Trace Signals sidebar lets you trace a signal value either forward or backward through the design hierarchy. For information on the Trace Signals sidebar, see Chapter 16, “Tracing Paths with the Trace Signals Sidebar.”

The Call Stack sidebar displays a call stack for VHDL procedures, processes, and functions, API applications, and a thread manager for SystemC processes. This tab appears in the Source Browser sidebar. See “Viewing the Call Stack” on page 74 for more information about this tab when the design is written in VHDL. For information on how to use this tab for API debugging, see Chapter 13, “Debugging API Applications.” For information on how to use this tab with SystemC designs, see the *NC-SC User Guide.*

The SystemC/C++/C Variables sidebar displays information about local variables and data members in SystemC and API applications. This tab appears in the Source Browser sidebar. For more information, see the *NC-SC User Guide.*

The Bookmarks sidebar lets you manage bookmarks, which save the current state of a window, so that you can return to it at any time. The bookmarks tab appears in the Design Browser, Source Browser, and Schematic Tracer windows. For more information, see “Using Bookmarks in the Source Browser” on page 77, “Using Bookmarks in the Design Browser” on page 106, and “Using Bookmarks in the Schematic Tracer” on page 180.

The Class Hierarchy sidebar displays information about SystemVerilog classes. For more information, see *SystemVerilog in Simulation*.

The Power sidebar displays information about powered-up and power-down signals in the design. For more information, see the *Low-Power Simulation Guide*.

The Comparison Results sidebar lets you access the mismatches found by SimCompare. For information on SimCompare, see the *SimCompare User Guide and Reference*.

Use the following buttons and menu choices to control the sidebar:

- To collapse the sidebar, click *Collapse Sidebar*,  , in the upper right corner of the sidebar.
- To expand the sidebar, click *Expand Sidebar*,  , or select the tab that you want to expand.

- To hide the sidebar, disable the *View – Sidebar* option or click *Hide Sidebar*, , in the upper right corner of the sidebar.
- To show the sidebar, enable the *View – Sidebar* option.
- To tear off the sidebar, click on the dotted line in the tab control button.

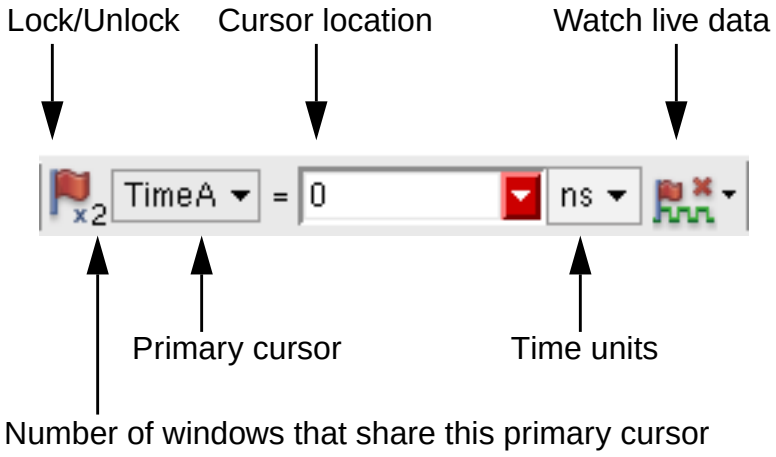
Tip

Because of SimVision's cross-selection capability, any object you select in the sidebar is also selected in all SimVision windows. Therefore, you can tear off a sidebar and close the sidebars in your other windows to maximize space in your SimVision windows.

Managing Time in the SimVision Windows

Most windows contain a time toolbar, shown in Figure 1-3 on page 30.

Figure 1-3 Time Toolbar



This toolbar lets you control the simulation time associated with the window, as follows:

- Click the *Lock/Unlock* button to lock and unlock the primary cursor in the window at a particular time. When simulation time is locked, the values displayed in the window remain at the locked time, even as simulation progresses. When you unlock the window, the primary cursor tracks simulation time.

  Note: Although the primary cusor and the values displayed in the window remain at the locked time, the simulation time in the simulation toolbar continues to update as

simulation progresses.

- The *Number of windows* icon shows you the number of windows that share the same primary cursor. When multiple windows share a primary cursor, the values displayed in those windows reflect the same simulation time.
- The *Primary Cursor* field lets you select the primary cursor for the window. A drop-down list contains the names of all cursors currently defined in all windows. You can choose a cursor from the list or choose *New Cursor* to create a cursor.
- The *Cursor Location* field lets you place the primary cursor at a specific simulation time. As you enter simulation times in the field, SimVision adds them to a drop-down list. You can quickly return to a time by selecting it from the list.
- The *Time Units* field lets you select the time units used in the window.
- The *Watch Live Data* button indicates whether the primary cursor is synchronized with the simulator or database. This can affect performance, but it lets you track signal value changes during simulation. The button has a drop-down menu from which you can choose the simulator or database that you want to track.

When synchronization is on, the *Watch Live Data* button shows a green arrow. The cursor location updates in the window as simulation time progresses. Other windows that share the primary cursor do not get updated until simulation pauses or stops.

When synchronization is off, the *Watch Live Data* button shows a red X and the cursor location does not change. Simulation data is generated, but the view remains stationary.

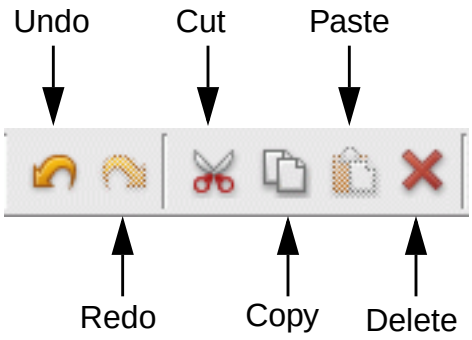
Important

If you move the primary cursor when *Watch Live Data* is enabled, synchronization is turned off.

Managing Simulation Objects

The *Edit* menu lets you cut, copy, paste, and delete objects in the window. These functions are also available as buttons in the standard toolbar, shown in Figure 1-4 on page 32. Many operations that you perform in a window can be undone. If so, the *Undo* menu choice and button are activated. When you undo an operation, the *Redo* menu choice and button are activated.

Figure 1-4 Standard Toolbar



Note: The *Undo* and *Redo* functions are not available in the Console window.

Selecting and Deselecting Objects

Many operations begin by selecting one or more objects.

To select individual objects or multiple objects:

- Select a single object by clicking on it.
- Select several contiguous objects by `Shift`-clicking on the objects, or by holding down the left mouse button and moving the cursor over the objects.
- Select non-contiguous objects by `Control`-clicking on the objects.

To deselect one or all selected objects:

- Deselect a single object by `Control`-clicking on the selected object.
- Deselect all selected objects by clicking on an empty space in the window.

Using Pop-Up Menus

When you have selected one or more objects, you can perform operations on them by using pop-up menus. Select the objects and right-click to display the pop-up menu. The choices in the pop-up menus differ, depending on the objects that you have selected.

Adding Objects to a Window

The *Add* button,  , is available in the standard toolbar of most SimVision windows. Click this button to add to the current window any objects that you have selected in another SimVision window.

You can quickly move objects from one window to another by performing a drag-and-drop operation.

To drag and drop objects:

1. Select the objects, then hold down the middle mouse button and move the cursor to the new location.

   As you move the mouse, the cursor indicates whether you may drop the objects at the current cursor location:

    The mouse is in an invalid location. You cannot drop the objects here.

    The mouse is in a valid location. You can drop the objects here.

2. Release the mouse button to drop the objects at the new location.

Drag-and-drop operations do not require you to first select an object. If you do not select an object, SimVision drags and drops the object closest to the mouse pointer. If that object is a scope in the design hierarchy, all objects in that scope are dragged and dropped to the destination. If that object is a signal, only that signal is dragged and dropped.

Using Keyboard Shortcuts and Hotkeys

Every menu choice is bound to a shortcut key sequence. You may find these shortcuts faster to use than the pull-down menus. To use a shortcut, press the `Alt` key, then type the underlined letter in each menu choice. For example, to choose *File – Open Database*, type `Alt-FO`.

SimVision also defines a set of hotkeys, which are bound to certain commands. You can see a list of hotkey definitions by choosing *Help – Keyboard Shortcuts*. You can change these keyboard shortcuts, as described in “Keyboard Shortcuts” on page 311, or define your own keyboard shortcuts, as described in Chapter 2, “Re-Creating an Interactive Session,” in the *SimVision Command Reference*.

Using the SimVision Command Language

Every operation that you can perform with the SimVision graphical user interface can also be performed with a command, as follows:

- Enter commands in the Console window while running SimVision.
- Create a command script that calls SimVision commands to open databases, create windows, cursors, and so on. You can then execute the command script at startup to initialize a SimVision session. SimVision can create this script for you, as described in Chapter 3, “Saving and Restoring Your Debugging Environment.”
- Create a command script that calls SimVision commands and Tcl commands to create a plug-in application. Plug-ins define new window types and add new features to SimVision.

See the *SimVision Command Language Reference* for more information.

Aborting a SimVision Action

There might be times when you want to stop an action before it has completed. For example, you might want to stop loading a very large database. When SimVision detects that an action will take a long time to complete, it adds an *Abort* button to the relevent window. You can click this button to stop the action and resume your SimVision session.

There might be times that no *Abort* button is available, for example, if you are trying to execute a command script that contains an infinite loop. If no *Abort* button is available, you can still stop an action and regain control of the SimVision windows by pressing the Escape key three times. The triple-Escape sequence will return control of the SimVision windows to you.

Getting Help

There are several ways to get help when running SimVision:

- Tooltips appear whenever you point your cursor at a button for a few seconds. The same message appears in the status bar at the bottom of the window.
- The *Help* button on the Preferences form displays interactive help. This help feature lets you click on a field in the form to see a description of the field.
- *Help* buttons on other forms take you to the relevant parts of the documentation.

- The *Help* menu contains the following choices:
  - *Help – Cadence Help Library* opens the Cadence Help Library window. This gives you access to all of the Cadence documentation installed on your system.
  - *Help – SimVision User Guide* opens this document.
  - *Help – SimVision Windows* gives you help on any type of SimVision window.
  - *Help – Tutorials* takes you to any of the tutorials installed on your system.
  - *Help – Keyboard Shortcuts* displays a list of hotkey definitions supplied with SimVision.
  - *Help – Command Reference* opens the *SimVision Command Language Reference*, which describes how to perform any SimVision function and create plug-in applications with the SimVision command language.
  - *Help – Related Products* takes you to the documentation for any of the simulators and other simulation tools installed on your system.
  - *Help – What's New* takes you to *What's New in IUS, which* describes the new features in this release.
  - *Help – KPNS* takes you to the *Known Problems and Solutions* documents for the products installed on your system. These documents describe any problems that are known to exist at the time of this release.
  - *Help – Customer Service* takes you to the Cadence web pages for SourceLink and Web Collaboration.
  - *Help – About SimVision* displays the SimVision version number and copyright information.

2

Invoking SimVision

Before you can invoke SimVision to simulate a design, you must compile and elaborate the design for debugging. You can then invoke SimVision in the following ways to access a design or a simulation database that you have already created:

- Invoke the simulator with SimVision, so that you have access to the simulator controls and the debugging tools.
- Invoke SimVision and connect to a running simulation.
- Invoke the SimVision post-processing mode, so that you have access to the simulation data but not to the simulator controls. Post-processing mode lets you analyze digital simulation results stored in a database. It lets you debug your design without using a simulator license. You cannot use post-processing mode to analyze mixed-signal results.

When you are connected to a simulation, you can disconnect or terminate the simulation. In the first case, the simulation continues to run; in the second case, the simulation ends. SimVision provides a password facility to control who is able to connect to a simulation.

Preparing Your Design for Simulation

When debugging your design, you may need to set breakpoints at particular line numbers in the source code. To make source line numbers available to SimVision, include one of the following options when compiling the design:

Command	Option
`ncvlog`	`-linedebug`
`ncvhdl`	`-linedebug`
`irun`	`-linedebug`

You may also want to make internal signals visible or trace the connectivity of the signals in your design. To control visibility and connectivity, include one of the following options when elaborating a design:

Command	Option
`ncelab`	`-access rwc` `-afile` *`access_file`*
`irun`	`-access rwc` `-afile` *`access_file`*

The `-access` option controls access for the entire design. The `r` argument enables read access, which makes internal signals visible to the simulator. The `w` argument enables write access, so that you can change the value of a signal during simulation with the `force` or `deposit` command. The `c` argument enables connectivity access. You can use one or all of these arguments to enable the types of access that you require.

If you need to control access to particular instances and portions of your design, you can create an access file. You specify the access file with the `-afile` or `+afile+` option.

For more information on these options, see “Enabling Read, Write, or Connectivitiy Access to Simulation Objects” in your simulator *Help*.

Invoking the Simulator with SimVision

You can invoke the simulator with SimVision, as follows, and have access to the full debugging features of the simulator and SimVision:

Simulator	Command
Verilog-XL	`verilog +gui` *`source_files`*
Incisive Simulator	`ncsim -gui` *`snapshot`*
`irun`	`irun -gui` *`source_files`*

To pass arguments to SimVision, use the `-simvisarg` command-line option. For example, to open a Waveform window and specify a command script for SimVision, invoke `ncsim` as follows:

```
ncsim -gui -simvisargs “ -waves” -simvisargs “ -input simvision.sv” test_drink
```

Important

You must put the SimVision option in quotes, regardless of whether it takes an argument. Otherwise, it as treated as an `ncsim` option. You must also add a space before the option. Otherwise, the hyphen causes a Tcl syntax error.

You can also invoke the simulator in Tcl mode, then invoke SimVision. For example:

```
ncsim -tcl test_drink
ncsim: version: (c) Copyright 1995-2004 Cadence Design Systems, Inc
ncsim> source your_install_dir/tools/inca/files/ncsimrc

ncsim> simvision
simvision: version: (c) Copyright 1995-2004 Cadence Design Systems, Inc.

-------------------------------------
Relinquished control to SimVision...
```

You can also specify a SimVision command file as an argument to the `simvision` command. SimVision executes the commands in that file at startup. For example, the following SimVision commands open a Waveform window, run the simulation, add some signals to the Waveform window, then terminate the simulator session:

```
set w [waveform new]
simcontrol run
scope set test_drink.top
waveform add -using $w -signals { nickel_in dime_in quarter_in dispense nickel_out
dime_out two_dime_out clk vending.current_state }
simcontrol command exit
```

If you put these commands in a file, you can invoke SimVision with the `simvision -input` command, as follows:

```
ncsim -tcl test_drink
ncsim: version: (c) Copyright 1995-2004 Cadence Design Systems, Inc
ncsim> source your_install_dir/tools/inca/files/ncsimrc

ncsim> simvision -input simviscmds.tcl
simvision: version: (c) Copyright 1995-2004 Cadence Design Systems, Inc.

-------------------------------------
Relinquished control to SimVision...
```

When you invoke SimVision with this command file, it runs the simulation, sets up the Waveform window, and terminates the simulator at startup. You can then debug the design with SimVision.

Important

When invoking SimVision from the simulator, you can start only one SimVision process. However, you can use this process to connect to any number of active simulator sessions and to open any number of simulation databases. If you try to start another SimVision session—that is, if you invoke `simvision` without any arguments—you receive the following error:

```
ncsim: *E, TCLERR: simvision is already active
```

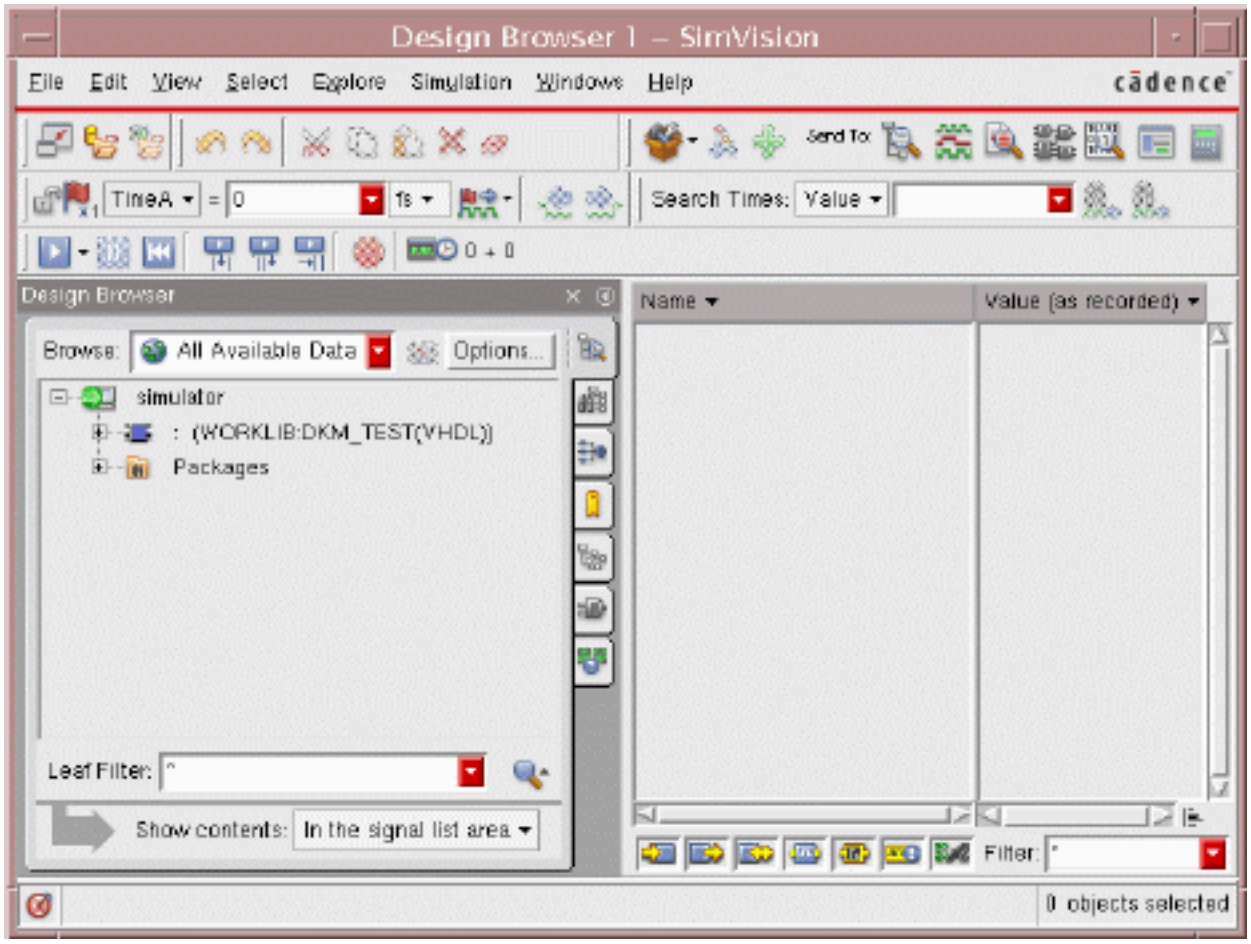
To send options to SimVision—for example, if you want to open other windows—issue the `simvision` command with the `-input` option.

For more information on the `simvision` command, see your simulator *Help*. For information on the SimVision commands that you can include in an input file, see the *SimVision Command Reference*.

When SimVision starts up, it opens a Design Browser window and a Console window. You access your design hierarchy in the Design Browser window, and enter SimVision and simulator commands in the Console window. The location of the Design Browser window is controlled by your window manager. If there is enough space below it, the Console window is placed there. Otherwise, it is place on top of the Design Browser window.

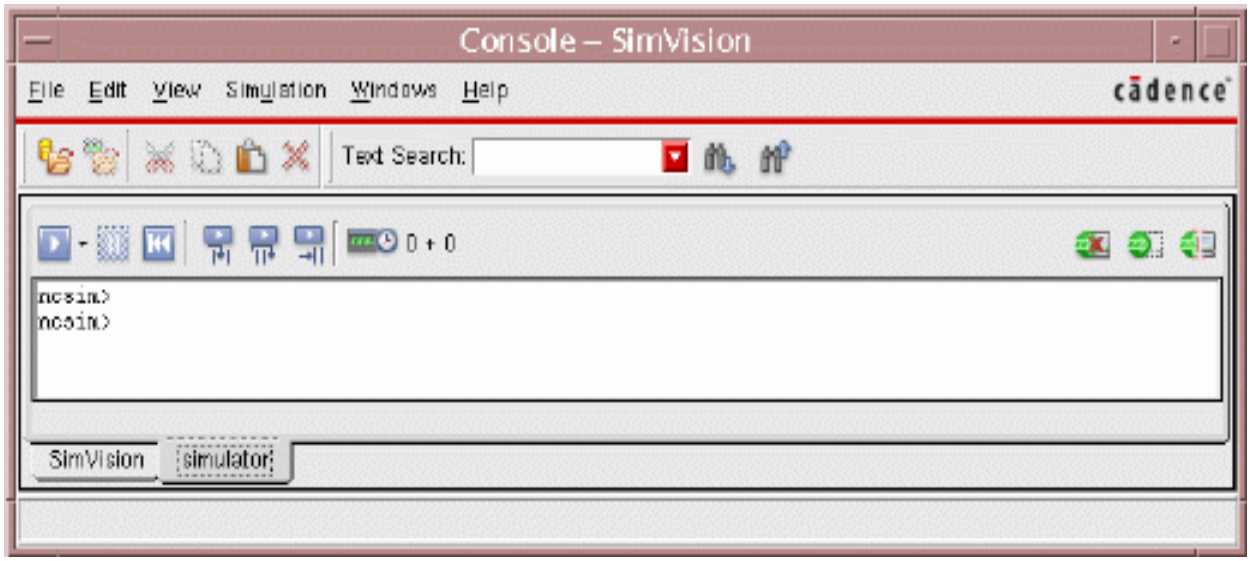
Figure 2-1 on page 41 shows a Design Browser window displayed at startup. SimVision places the simulation at the top of the hierarchy and assigns it the name `simulator`.The top-level of the design hierarchy is placed below the simulation. In this example, it is named `dkm`.

Figure 2-1 Design Browser Window



At startup, the Console window has two tabs, as shown in Figure 2-2 on page 42. The *SimVision* tab lets you enter SimVision commands. The *simulator* tab lets you enter simulator commands. As you run the simulation, the Console window also displays messages from SimVision and the simulator.

Figure 2-2 Console Window



Disconnecting and Terminating the Simulation

There are several ways to break the connection between SimVision and the simulator:

- Disconnect from the simulation. This leaves the simulation running but makes it unavailable from the SimVision user interface. You can reconnect to a disconnected simulation.
- Terminate and disconnect the simulation. This stops the simulator. You cannot reconnect to the simulation after you have terminated it.
- Terminate and post-process. This stops the simulation but keeps the connection with the simulator in post-processing mode.

Disconnecting from a Simulation

To disconnect from a simulation:

➤ Click *Disconnect*, , in the *simulator* tab of the Console or Properties window. When you disconnect, the simulation toolbars are deactivated in all windows, and the *simulator* tab is removed from the Console window.

You can reconnect to the simulation by choosing *File – Open Simulation* and selecting the simulation from the Open Simulation form.

You can also disconnect from a simulation from the *File* menu:

1. Simvision opens the Close Database/Simulator form, which contains a list of running simulations and open databases.
2. Select *simulator* from the list and click *OK*.

For information on using the Close Database/Simulator form to close a database, see "Closing a Database" on page 123.

Terminating and Disconnecting from a Simulation

To terminate and disconnect a simulation:

➤ Click *Terminate & Disconnect,*  , in the *simulator* tab of the Console or Properties window, or choose *Simulation – Terminate & Disconnect* from the menu bar of any SimVision window. When you terminate, the simulation toolbars are deactivated in all windows, and the *simulator* tab is removed from the Console window.

SimVision displays a warning message, informing you that you cannot return to simulation mode for this design. When you click *OK*, the *File – Open Simulation* menu choice is deactivated.

 *Tip*

You might also type `exit` at the `ncsim` prompt to terminate the simulation. This command deactivates the simulation toolbar and the *File – Open Simulation* menu choice, removes the *simulator* tab from the Console window, and puts you in post-processing mode. However, SimVision does not display a warning message.

Terminating and Post-Processing

You might switch to post-processing mode during simulation. To do this, first probe the signals whose values you want to save, and run the simulation to the point that you are interested in debugging. You can then terminate the simulator session and enter post-processing mode.

To terminate a session:

➤ Click *Terminate & Post-Process,*  , in the *simulator* tab of the Console window or Properties window, or choose *Terminate & Post-Process* from the *Simulation* menu.

Before it enters post-processing mode, SimVision displays a warning message. Entering post-processing mode releases the simulator license. When you terminate the simulation,

you cannot reconnect to it. If you need to rerun the simulation, you must exit SimVision and reinvoke the simulator.

When you enter post-processing mode, SimVision prompts you to press `Return` or click the *X* to remove the simulator tab. When you close the tab, the simulation toolbar is disabled in all SimVision windows.

In post-processing mode, you can access the entire simulation hierarchy, including objects you have probed and objects you have not probed to a database. Probed objects are shown in regular font; unprobed objects are shown in italic font.

To see only the objects that you have probed:

➤ Enable the *Show/Hide unprobed signals* filter button in the signal list.

Invoking SimVision Separately

You can invoke SimVision separately, and then connect to a running simulation or open a database containing the results of a simulation that you have previously run. You invoke SimVision with the `simvision` command.

The simivision command has the following syntax:

`simvision [`*`options`*`] [`*`database`*`]`

The `simvision` command accepts the following options:

`-64BIT`	Invokes the 64-bit version of SimVision. When you use this option, you must also use the `-64BIT` option to compile, elaborate, and simulate your design. However, you may also configure your environment to use the 64-bit version exclusively. For more information, see "Configuring Your Environment for the 64-Bit Version", in the *IUS 8.1 Configuration Guide*.
`-APPEND_KEY`	Appends captured keyboard input to the key file specified by the `-KEYFILE` option.
`-APPEND_LOG`	Appends messages from the SimVision session to the log file specified by the `-LOGFILE` option.

`-COMPRESS`	Translates a database to compressed SST2 format. The compressed database uses less disk space at the possible expense of taking more time and memory to translate. Use this option when you are translating a VCD, HSPICE list, HSPICE transient output, Nutmeg, Epic, or Qsim database, if you want to compress the resulting SST2 database.
`-CONNECT` *`session`*	Connects to a simulator session at startup. You can specify the *`session`* argument in any of the following ways: `simvision -connect dialog` SimVision displays the Open Simulation form at startup, so you can choose the simulation to which you want to connect. For more information, see "Connecting to a Simulation" on page 47. `simvision -connect` *`pid`* Connects to the simulation that has the specified process ID (*`pid`*) on the local host. `simvision -connect` *`host/pid`* Connects to the simulation that has the specified process ID (*`pid`*) on the specified host.
`-DISPLAY` *`screen`*	If you are running SimVision remotely, you can display the output on your system monitor. For example: `simvision -display MY_SYS:0.0` This has the same effect as setting the `DISPLAY` environment variable.
`-HELP`	Displays a short description of the `simvision` command-line options.
`-INPUT` *`file`*	At startup, executes SimVision commands stored in *`file`*. If you want to execute more than one command script, use multiple `-INPUT` options. The command scripts are executed in the order in which they appear on the command line.
`-KEYFILE` *`file`*	Specifies the name of the key file, where captured keyboard input is written during the session. If you do not specify `-KEYFILE`, no keyboard input is saved.
`-LOGFILE` *`file`*	Specifies the name of the log file, where messages are written during the session. If you do not specify `-LOGFILE`, no messages are saved.

-NOCOPYRIGHT	By default, SimVision displays a splash screen and copyright notice when it starts up. If you do not want to see this notice, use the -NOCOPYRIGHT option.
-NOPLUGINS	Disables the loading of plug-in applications.
-NORC	Specifies that you do not want to execute the commands in a .simvisionrc file at startup.
-NORESTORE	You can set a preference to restore the application state when you reinvoke SimVision. SimVision reopens any databases and windows that were open. It also restores the windows to their previous state, including signals, cursors, markers, groups, mnemonic maps, and expressions that you defined. If you do not want to restore these settings, use the -NORESTORE option.
-PRIVATE	Installs a private colormap for SimVision.
-REGISTER	Opens the Register window at startup.
-SCHEMATIC	Opens the Schematic Tracer window at startup.
-SEQUENCE	Includes sequence time information in a translated database. Use this option when you translate a VCD, HSPICE list, HSPICE transient output, Nutmeg, Epic, or Qsim database, and you want to include sequence information in the database.
-SNAPSHOT *snapshot*	Invokes SimVision in post-processing mode and loads the specified snapshot, including a partially-elaborated snapshot. When you use this option, you can view the design hierarchy in the Design Browser and Schematic Tracer, and you can access the source code in the Source Browser, but no simulation data is available, such as waveforms or signal values. If you also specify a database command-line argument, the simulation database is merged with the design hierarchy.
-SOURCE	Opens the Source Browser at startup.
-SST *sst_dbase*	Specifies the name that you want to give to the translated database. By default, SimVision creates two files, with the same name as your original database and the .dsn and .trn extensions. When you use this option, it creates two files, named *sst_dbase*.dsn and *sst_dbase*.trn.

`-TITLE` *`title`*	Specifies a string that you want to include in the title bar of every SimVision window that you create during the session. If the *`title`* argument contains spaces, you must enclose it in quotation marks. For example: `simvision -title "My SimVision"`
`-VERSION`	Displays the version of SimVision installed on your system.
`-WAVES`	Opens the Waveform window at startup.

You can specify default command-line options by setting the `SIMVISIONOPTS` environment variable. For example, if you want the Source Browser to open when you invoke SimVision, you can set `SIMVISIONOPTS` as follows:

```
setenv SIMVISIONOPTS “-source”
```

If you want to specify multiple command-line options, separate them with spaces within the quotation marks. For example:

```
setenv SIMVISIONOPTS “-source -input mytcl.sv”
```

These options open a Source Browser window and execute the commands in the file called `mytcl.sv` whenever you run SimVision.

When you invoke SimVision with no simulation database or simulator connection, it does not open a Console window. It opens a blank Design Browser window. You can access a design by connecting to a simulation or opening a simulation database.

Connecting to a Simulation

To connect to a running simulation:

1. Click *Open Simulation*,  , or choose *File – Open Simulation* from the menu bar. SimVision opens the Open Simulation form.

   When *Show all hosts* is disabled, the form displays all of the simulations that the user is running on the selected host. When *Show all hosts* is enabled, the form displays either the names of the designs being simulated or *No sessions available* for each host in the *Add Host* drop-down list.

2. Select a simulation and click *Open*. SimVision adds the design to the Design Browser sidebar. The Design Browser sidebar gives you access to the design hierarchy.

   SimVision also adds a *simulator* tab to the Console window. The Console window lets you enter simulator commands.

By default, the Open Simulation form displays only the simulations that you have started. However, you can connect to simulations started by other users.

To access simulations started by other users:

➤ Enter the user name in the *Show Simulations for User* field. When you enter a user name, SimVision adds it to the drop-down list.

Because SimVision keeps a history of the user names that you enter, you can enter a partial name and press Tab. If the history contains one matching name, SimVision seeds the field with that name. If the history contains more than one matching name, it displays a list of those names, and you can choose one.

When you try to connect to a simulation started by another user, SimVision prompts you to enter a password. For information on protecting your simulations with passwords, see "Making Simulation Processes Secure" on page 51.

To add a host to the *Add Host* list:

➤ Enter the host name in the text field and click *Add*,  . As you enter host names, SimVision adds them to the drop-down list.

Because SimVision keeps a history of the host names that you enter, you can enter a partial name and press Tab. If the history contains one matching name, SimVision seeds the field with that name. If the history contains more than one matching name, it displays a list of those names, and you can choose one.

To remove any simulations running on a particular host:

➤ Choose the host from the drop-down list in the *Add Host* field and click *Remove*.

Simulations are protected by passwords. For information on the security features of SimVision, see "Making Simulation Processes Secure" on page 51.

Opening a Simulation Database

To open a database:

1. Click *Open Database*,  , or choose *File – Open Database* from the menu bar. SimVision opens the Open Database form.

   The path shown in the *Directory* field is determined by the SIMVISION_WORKDIR environment variable. If you have not set this variable, the path is to the last directory visited by any of the file browser forms. You can navigate to other directories to find the database that you want to open.

2. Select the simulation database file and click *Open*. The Design Browser sidebar displays the design hierarchy. If you saved simulation information for only some of the design objects, only those objects appear in the Design Browser sidebar.

A simulation database does not include all of the connectivity information required by the Schematic Tracer and Trace Signals sidebar. In addition, it does not include source line information required by the Source Browser. This information is stored in the snapshot.

The simulator stores the location of the snapshot in the simulation database. If have loaded a database into SimVision and you try to open one of these windows, SimVision looks for the snapshot associated with the database. If it finds the snapshot, SimVision loads it before opening the window. If SimVision cannot find the snapshot, it prompts you for the snapshot name.

Important

SimVision cannot locate a snapshot for a database created before Version 5.3. For these older databases, SimVision prompts you for the snapshot information.

This feature is supported by Verilog-XL. If you are a Verilog-XL user and you want to use this feature, you need to recompile your design with `ncverilog`, and specify the `-c` option to prevent simulation.

To specify the snapshot:

1. Enter the name of the library mapping file in the *cds.lib* field, or choose one by clicking the browse button. If you used the default library mapping file when you created the snapshot, leave this field blank. However, to use the default `cds.lib` file, you must be in the directory where the snapshot resides.
2. Enter the snapshot name in the *Snapshot* field, or choose a snapshot from the drop-down list.

Tip

You can also load a snapshot from a Design Browser sidebar. Right-click the database name and choose *Explore Full Design* from the pop-up menu.

After you load the snapshot, you will be in full post-processing mode. You can access all of the design information and simulation data.

Invoking SimVision in Post-Processing Mode

If you have already simulated your design and saved the simulation results to a database, you can run SimVision in post-processing mode. In post-processing mode, you have access to the same tools as you would have during an interactive session, except for the simulation controls. That is, the Simulation menu and toolbar are not available. You cannot run the simulation or set breakpoints and probes.

You invoke SimVision in post-processing mode, as follows:

Tool	Command
Verilog-XL	`verilog +ppe` *`source_files`*
Incisive Simulator	`ncsim -ppe -ppdb` *`database snapshot`* You can use multiple `-ppdb` options to specify multiple databases for the same snapshot.
SimVision	`simvision -snapshot` *`snapshot database`*

When you invoke `ncsim` with the `-ppe` option and then open a simulation database, SimVision does not overlay the signals in the snapshot and database in the Design Browser sidebar. The database and snapshot are displayed in separate hierarchies.

When you invoke `ncsim` with both the `-ppe` and `-ppdb` options, SimVision overlays the snapshot and database. As a result, only one hierarchy is displayed in the Design Browser sidebar. In the Design Browser signal list, the signals in the database are displayed in regular font, indicating that they have been probed. The signals that have not been probed are displayed in italic font.

See Chapter 5, “Accessing Design Objects” for information about the Design Browser sidebar, and Chapter 6, “Monitoring Signal Values”. for information about the Design Browser signal list

Tip

When you invoke `ncsim` with a Tcl input file, the simulator executes the commands in that file before starting SimVision. If the last command in the file is an `exit` command, the simulator invokes SimVision in post-processing mode and terminates the simulation. To prevent the simulator from terminating, remove the `exit` command from the input file before you invoke the simulator.

Making Simulation Processes Secure

Because users can access simulation sessions running on a network, SimVision provides features to help you make them more secure.

You can control the visibility of a session by setting the `LDV_SIMVISION_CONNECTIONS` environment variable as follows:

- When set to `on`, `1`, or `yes`, the simulation is available for connection. That is, it can appear in the Open Simulation form.
- When set to `hidden`, the simulation is not visible in the Open Simulation form, but it is still available for connection. For example, you can specify its process ID when you invoke SimVision with the `-connect` option.
- When set to `off`, `0`, or `no`, connections are not allowed.

SimVision also lets you place passwords on your simulation processes, as follows:

- SimVision stores your passwords in a file called `$HOME/.simvision/passwd`. You have read-write access to the file, so that you can connect to the simulations you start on your local machine or on any machine on the network that shares the same home directory.

The `passwd` file contains encrypted information. Do not edit this file.

The simulator uses your password for all simulator processes that you run. When you connect to a simulation, SimVision compares the passwords in the password file to the password that is set on the simulation. If it finds a match, SimVision connects to the simulation. If not, it prompts you for the password. This allows other users to connect to simulations that you start, if they know the password.

To set a password for a simulation:

1. Invoke the simulator with the `-password` option. For example, if you are running the Incisive simulator:

   ```
   ncsim -password -tcl test_drink.top:module
   ncsim: 04.20-b002: (c) Copyright 1995-2002 Cadence Design Systems, Inc.
   Simulation password:
   ```

   The simulator prompts you for a password.

2. Enter a password and press `Return`. The simulator prompts you to verify the password by entering it again, and asks if you want to make this the default password.

3. Choose `yes`, and the new password replaces the default password, if one exists. Choose `no`, and the new password is added to the `passwd` file along with the current default password.

To change the location of the `passwd` file:

➤ Set the `LDV_SIMVISION_PASSWORD_FILE` environment variable to the new location. For example:

```
setenv LDV_SIMVISION_PASSWORD_FILE $HOME/mypassword/pwd
```

To prevent any user from connecting to a simulation:

➤ Set the `LDV_SIMVISION_PASSWORD_FILE` environment variable as follows:

```
setenv LDV_SIMVISION_PASSWORD_FILE --
```

This is useful when running in batch mode if you do not want anyone to connect to the simulation with SimVision.

3

Saving and Restoring Your Debugging Environment

Sometimes you will want to re-create a debugging session. For example, you might want to save a set of probes and breakpoints on a design under development, so that you can re-create them at any time. You may want to save SimVision preferences, re-create a complex mnemonic map, or open the same set of SimVision windows every time you simulate a design. SimVision provides several ways for you to save and restore your debugging environment.

The Simulator and SimVision Environments

The simulator and SimVision operate in their own environments:

- The simulator environment is made up of the probes, breakpoints, forces, and deposits that you set during simulation.
- The SimVision environment is made up of your preference settings and the windows, cursors, groups, and other objects that you create to make debugging your design easier.

  The following are not treated as part of the environment and, therefore, cannot be saved and restored:

  - The scope set in the Design Browser sidebar. When you restore this sidebar, it displays the top-level scope.
  - The current simulation time, if you are saving the connection to the simulator. Simulation time is restored at time 0.
  - The contents of the Trace Signals sidebar.
  - The contents of the Console window.
  - The Expression Calculator window, if it is open when you save the environment. However, any expressions that you create are saved.

Most of the objects in the simulator and SimVision environments are unique, but both tools operate on databases, as follows:

- The simulator creates the database. You set probes on the signals whose transitions you want to save. When you run the simulation, the simulator generates the data and writes it to the database. When you restore a simulator environment, the command script creates the database and sets the probes, but it does not run the simulation. Therefore, the database itself is not restored—only the means to re-create it.
- SimVision reads the simulation database and displays the data in various windows. You can tie a command script to a specific database or use it as a template.

  When you tie a command script to a specific database, references to signals include the database name, as in `waves::test_drink.top.dispense`. You can use this script only on the specified database.

  When you want to use a command script as a template, references to signals do not include the database name, as in `test_drink.top.dispense`. You can use this script on any database that has the same design hierarchy. For example, you could simulate a design twice and save the results in different databases. (Perhaps you forced a signal to a particular value in one simulation to produce different results.) You can use the same command script for both databases, because they contain similar data.

SimVision can save the SimVision environment automatically on exit and restore it the next time you start another session, but you cannot automatically save and restore the simulator environment. If you want to save both the SimVision environment and the simulator environment, you must create command scripts that contain the Tcl commands necessary to restore the environment.

Using Automatic Save and Restore

Important

The automatic save and restore feature will not be supported in the next release of SimVision. You should save and restore your simulation environment in a command script.

When you exit a SimVision session, SimVision can automatically save your debugging environment. By default, SimVision saves your environment in the following directories:

- A `.simvision` directory in your `$HOME` directory stores global settings, such as options you have set in the Preferences window.
- A `.simvision` directory in your working directory stores design-specific data, such as bookmarks you have created during a SimVision session.

You can specify a different `$HOME/.simvision` directory by setting the `SIMVISION_HOME` environment variable. For example, if you want to store your global settings in the same directory as your working directory, you can define the variable as follows:

```
setenv SIMVISION_HOME .
```

To specify that you want to save and restore your SimVision environment automatically:

1. Choose *Edit – Preferences* from any SimVision menu and select the *General Options* tab in the Preferences window.
2. Enable *Restore Application State* to automatically restore your environment at startup.

Tip

If you have enabled this button and you do not want to restore the application state, use the `-norestore` command-line option when you run `simvision`.

3. Enable *Save Application State* to automatically save your settings on exit.

For example, suppose you invoke the simulator with SimVision, send some signals to the Waveform window, run the simulation, and create a marker. When you exit this session, SimVision saves the Waveform window, the marker, and the simulation database created during simulation.

There are several ways to restore this environment:

- Run SimVision in the original design directory.

  SimVision creates the windows and marker, and loads the simulation database. As a result, the Design Browser contains the database hierarchy, and the Waveform window contains the signals and waveforms from the simulation database. This re-creates the SimVision environment so that you can debug the simulation database in post-processing mode.

- Run the simulator with SimVision in the original design directory.

  SimVision loads the snapshot into the Design Browser and creates the Waveform window and marker. SimVision then opens the simulation database from the previous session. As a result, the Design Browser contains both the design hierarchy and the database hierarchy, and the Waveform window contains the signals and waveforms from the simulation database. You can run the simulation again and also examine the simulation data generated from the previous simulation run.

- Run SimVision with or without the simulator, but in a different design directory.

SimVision creates the window and marker, but does not load the simulation database. This is useful if you want to use the same SimVision environment on another design or simulation database.

Saving and Restoring a Command Script for Simulator Connection

When you save a command script for simulator connection, SimVision creates two scripts—one containing simulator commands, and one containing SimVision commands. By default, the simulator command script is named `restore.tcl`, and the SimVision command script is named `restore.tcl.svcf`.

To save the command scripts for simulator connection:

1. Choose *File – Save Command Script* from any SimVision window. SimVision opens the Save Command Script form.
2. Enable *Save connection to simulator*.
3. If you have loaded any preexisting databases during the simulation session, you can choose whether you want to use the script to reload those databases, or to load any similar databases:
   - Enable *Save all database names* to use the script to load the same databases that you have loaded in the current session.
   - Enable *Prompt me later for database names* to use the script as a template. When you restore this command script, SimVision tries to use a database that is already open. If no open databases contain the necessary data, it prompts you to choose another database to open.

   Note: These options do not apply to databases you have created during simulation. If you have not loaded any preexisting databases during this session, these options are disabled.
4. Click *Advanced* to open the advanced options area.
5. Check the items that you want to save in the command script, uncheck those items that you do not want to save.
6. When *Save these advanced settings* is enabled, SimVision applies these settings whenever you save a command script. If you do not want these settings to be the default, disable this option.

7. Click *OK*. When you exit SimVision, the command to restore the environment is displayed on `stdout`. This command also appears at the beginning of the command script.

   Note: The suggested command line may not produce the results you expect, especially if you have used an input file to set up the simulation environment. SimVision cannot parse the input file to determine whether the file has, for example, set probes or deposited values. These operations may be performed twice if you use the suggested command line. You may need to modify the command line to recreate the desired environment.

To restore the debugging enviroment at startup:

➤ Invoke the simulator with the `-input` option. For example:

```
ncsim -gui test_drink -input restore.tcl
```

The `restore.tcl` command script restores the simulator environment, then it invokes `simvision` to restore the SimVision environment from the `restore.tcl.svcf` command script.

To source the command script after startup:

➤ Invoke the simulator with the `-gui` option, then Choose *File – Source Command Script* from any SimVision window. SimVision opens the Source Command Script form.

In the *Filename* field, enter the name of the simulator command script, or click *Browse*,  , to locate the file. In the *Send commands to* field, choose *simulator Console*.

To source the command script from the Console window:

➤ Invoke the simulator with the `-gui` option, then open the *simulator* tab of the Console window and enter the command: `input restore.tcl`. The commands are executed in the Console window.

Saving and Restoring a Command Script for Post-Processing

There are several ways to run SimVision in post-processing mode, as described in Chapter 2, “Invoking SimVision.” When you save a command script for post-processing, SimVision creates only one script to restore the SimVision environment. By default, the command script is named `simvision.svcf`.

To save a command script for post-processing:

1. Choose *File – Save Command Script* from any SimVision window. SimVision opens the Save Command Script form.
2. Enable *Save data for post-processing*.
3. Choose whether you want to apply the the script to a specific database or use it as a template, as follows:
   - Enable *Prompt me later for database names* to use the script with any database that contains similar data. When you restore this command script, SimVision tries to use a database that is already open. If no open databases contain the necessary data, it prompts you for a database to open.

     This is recommended if you have switched to PPE mode in the simulator, or if you ran the `ncsim` with the `-ppe` and `-ppdb` options.
   - Enable *Save all database names* to use the script with the same databases that are open in the current session.
4. Click *Advanced* to open the advanced options area.
5. Check the items that you want to save in the command script, uncheck those items that you do not want to save.
6. When *Save these advanced settings* is enabled, SimVision applies these settings whenever you save a command script. If you do not want these settings to be the default, disable this option.
7. Click *OK*. When you exit SimVision, the command to restore the environment is displayed on `stdout`. This command also appears at the beginning of the command script.

To restore the SimVision environment at startup:

➤ If you chose to save all database names, you can restore the command script as follows:

```
simvision -input simvision.svcf
```

➤ If you chose to prompt for database names, you restore the command script as follows:

```
simvision -input simvision.svcf
```

Then SimVision prompts you for the database name in the Open Database form.

Enter the name of the database, or click *Open* to choose a database. If you do not want to open a database at this time, click *Skip this database*.

Note: SimVision does not prompt you for the database name if it can find a database with the same logical name as the one that you saved.

➤ If you know the names of the databases that you want to use, you can specify them on the command line, as follows:

```
simvision -input simvision.svcf database1 database2
```

When you specify one or more databases on the command line, SimVision loads the databases and does not prompt you, unless the databases do not contain the necessary data.

To source the command script after startup:

1. Invoke SimVision in PPE mode, then choose *File – Source Command Script* from any window. SimVision opens the Select SimVision Command Script form.
2. Enter a filename or browse the directory structure to select a file.

   The path shown in the *Directory* field is determined by the `SIMVISION_WORKDIR` environment variable. If you have not set this variable, the path is the last directory visited by any of the file browser forms. You can navigate to other directories to find the database that you want to open. Select a file from the directory list, and click *Open*.

To source the script from the Console window:

➤ Open the *SimVision* tab and enter the `source` command.

Examining the Contents of a Command Script

If you need to edit a command script, you should become familiar with its contents and with the command languages provided by the simulator and SimVision.

Command scripts are text files. You can edit the files if you want to customize the environment that the script restores. For example, you could create other SimVision windows that were not open when you saved the environment, or you could create additional simulation objects, such as probes, breakpoints, forces, or deposits.

You can also edit the command script to perform Transaction Explorer commands, as described in the *Transaction Explorer Reference Manual*, or any general-purpose Tcl/Tk commands.

If you save the simulator connection, SimVision creates two scripts—one to restore the simulator environment, and one to restore the SimVision environment. If you choose to use the script for post-processing, SimVision creates only one script that restores the SimVision environment.

The Simulator Command Script

The simulator command script begins with a comment that describes how to restore the debugging environment. For example:

```
# You can restore this configuration with:
#
# ncsim -gui test_drink -input restore.tcl
#
```

The simulator command script opens databases, creates probes and breakpoints, and invokes the SimVision command script.

For example, if you opened a database called `waves.shm`, the command script contains the following simulator command:

```
database -open -shm -into waves.shm waves -default
```

If you probed any signals, it contains a `probe` command, such as the following:

```
probe -create -database waves test_drink.top.cans test_drink.top.clk
test_drink.top.dime_in test_drink.top.dime_out test_drink.top.dimes
test_drink.top.dispense test_drink.top.empty test_drink.top.exact_change
test_drink.top.load test_drink .top.nickel_in test_drink.top.nickel_out
test_drink.top.nickels test_drink.top.quarter_in test_drink.top.reset
test_drink.top.two_dime_out
```

The last line of the script invokes the SimVision command script, as follows:

```
simvision -input restore.tcl.svcf
```

See the *Help* for your simulator for details about the simulator commands that you can add to the simulator command script.

The SimVision Command Script

SimVision command script begins with a comment that describes how to restore the debugging environment. For example, the following command restores a simulator connection:

```
# SimVision Command Script (day MM dd hh:mm:ss EST yyy)
#
# You can restore this configuration with:
# ncsim -gui worklib.test_drink:module -input restore1.tcl
```

The command script contains commands to specify the preferences that you have set. This is done with a series of `preferences set` commands. For example, if you have disabled the *Prompt on Exit* option in the General Preferences tab, the command script contains the following command:

```
#
```

```
# preferences
#
preferences set prompt-exit 0
```

If you chose to save database names, the script contains a `database require` command, which specifies the name of the database to load.

For example:

```
#
# databases
#
database require waves -search {
        ./waves.shm/waves.trn
        /hm/yourhome/tutorial/waves.shm/waves.trn
}
```

The script then creates the markers, groups, windows, and other SimVision objects. For example, if you created a marker, the script contains the following commands:

```
#
# markers
#
set time 68058ns
if {[catch {marker new -name  {Marker 1} -time $time}] != ""} {
    marker set -using {Marker 1} -time $time
}
```

See the *SimVision Command Reference* for a description of the SimVision commands that you can place in a command script.

Executing SimVision Commands at Startup

At startup, SimVision looks for a file named `.simvisionrc`. This file can contain any SimVision or Tcl commands that you want to execute every time you start up SimVision.

The commands that you place in the file can set up your debugging environment. For example, if you place the following commands in a `.simvisionrc` file, SimVision opens a database, creates a waveform window, and adds two signals to the window:

```
database open waves.shm/waves.trn
waveform new
waveform add -signals {test_drink.top.clk test_drink.top.vending.current_state}
```

In the following `.simvisionrc` file, the `plugin::application` command disables certain options in the Advanced portion of the Save Command Script form:

```
plugin::application script categories configure dbnames -enabled 0
plugin::application script categories configure markers -enabled 0
plugin::application script categories configure conditions -enabled 0
plugin::application script categories configure mmaps -enabled 0
```

When SimVision executes these commands, it disables the *Write database names*, *Markers*, *Conditions*, and *Mnemonic Maps* options in the Save Command Script form. When you exit SimVision, it does not save this information to the `simvision.svcf` file.

You can define more than one `.simvisionrc` file. By default, SimVision searches for the file first in the following order:

`.`	Current directory
`$CDS_WORKAREA`	User-defined work area
`$CDS_SEARCHDIR`	Tool-defined area; set when certain tools start up
`$HOME`	User's home directory
`$CDS_PROJECT`	Project-specific storage area
`$CDS_SITE`	Site-specific setup area
`$(compute:THIS_TOOL_INST_ROOT/share`	Cadence default setup area

For information on how to change this default search order, see Chapter 3, "Cadence Setup Search File: setup.loc," in the *Cadence Application Infrastructure User Guide*.

SimVision executes the commands in only the first file that it finds. However, you can execute commands in multiple files by using the `source` command. For example, suppose the following `.simvisionrc` file is contained in the current working directory:

```
database open waves.shm/waves.trn
waveform new
waveform add -signals {test_drink.top.clk test_drink.top.vending.current_state}
source ~/.simvisionrc
```

At startup, SimVision executes the commands to open the database and Waveform window, then it executes the commands in the `.simvisionrc` file contained in the user's home directory.

For information on the commands that you can place in a `.simvisionrc` file, see the *SimVision Command Language Reference*.

4

Accessing the Design Source Code

The Source Browser gives you access to the source code for your design. It recognizes the Verilog, SystemVerilog, VHDL, C, C++, and SystemC languages, and it can color-code keywords, comments, and message strings that it detects in the source files.

The Source Browser lets you access design objects and perform simulation functions, such as setting breakpoints and probes. You cannot edit the source code directly in the Source Browser, because it would no longer represent the design snapshot currently loaded into the simulator. However, you can use the Source Browser to invoke a text editor, make changes to the source file, and then create a new snapshot and load it into the simulator.

Opening a Source Browser Window

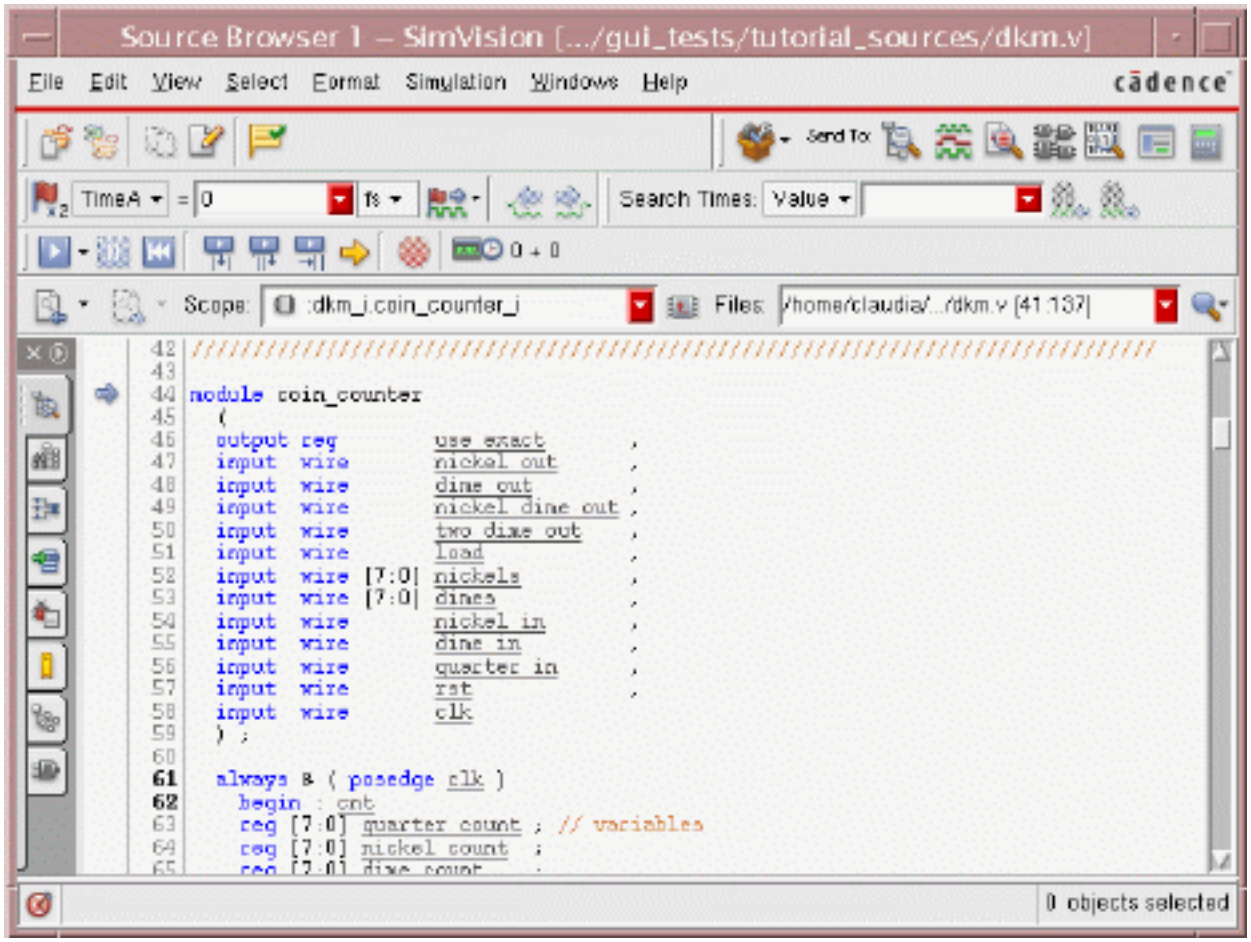
To display a source file for a selected scope or design object:

➤ Select a scope or object in any SimVision window and click *Source Browser* ,  .

If you select nothing before you click the *Source Browser* button, the target Source Browser window opens to the debug scope, if you are attached to a simulator. All other Source Browser windows are empty when you open them, unless you send a scope or object to them.

The Source Browser displays the source code associated with the selected scope, as shown in Figure 4-1 on page 64.

Figure 4-1 Source Browser Window



The standard toolbars and menus perform the common functions available in all SimVision windows, such as controlling the simulation. A navigation toolbar lets you move through the design hierarchy and open source files for the scopes that you select. You can also add a toolbar to search for text strings and line numbers within the source file.

The source file uses the following default colors to show the type of objects in the source file:

Blue	Language keywords
Green	Text strings, and objects that have been traced
Red	Comments

You can change these default colors, as described in “Source Browser Colors” on page 319.

Navigating the Design Hierarchy

From the Source Browser, you can access the design hierarchy and the files associated with each module or block in the hierarchy, including SystemC processes, modules, and instances.

Note: You cannot navigate a UDP with the Source Browser.

To move down the hierarchy one level:

➤ Select a scope from the drop-down list in the *Scope* field. When you select the subscope, the Source Browser displays the source code for that scope.

To move up one level in the hierarchy:

➤ Click the *Scope Up* button,  .

As you move through the hierarchy, the Source Browser maintains a history of the scopes you have visited. By default, the Source Browser history list contains the 20 most-recently visited scopes. You can change the size of the history list in the Preferences form, as described in "Source Browser" on page 318.

To move through the scope history:

➤ Click *Previous Scope*,  , to move back through the scope history. This button also has a drop-down list, so that you can jump to any scope in the list.

➤ Click *Next Scope*,  , to move forward through the scope history. This button also has a drop-down list, so that you can jump to any scope in the list.

As you move through the hierarchy, the Source Browser maintains a list of the source files or design units that are related to the current scope, such as included files and VHDL units.

To access a source file or design unit:

➤ Select it from the drop-down list. This list is labeled *Files* for Verilog designs. It is labeled *Units* for VHDL designs.

Opening a File in the Source Browser

You can open any source file, regardless of whether it is part of your design.

To open a source file:

1. Click *Open Source*,  , or choose *File – Open Source File*. SimVision opens the Open Source File form.

   The path shown in the *Directory* field is determined by the `SIMVISION_WORKDIR` environment variable. If you have not set this variable, the path is the last directory visited by any of the file browser forms. You can navigate to other directories to find the source file that you want to open.

2. Select a file and click *Open*.

 *Tip*

If the source file is going to take a long time to load, the Source Browser adds an *Abort* button to the status bar. Click this button if you do not want to load this source file.

Selecting Text in the Source Browser

The Source Browser defines the following mouse actions for selecting text:

Double-click	Select a word
Triple-click	Select a line
Quadruple-click	Select all
Shift-click	Select all text between two cursor locations

After you select text in the Source Browser, you can paste it into a text file.

When you select an area of text in the Source Browser window, you select the signals and variables contained within that text area. See “Accessing Design Objects and Values” on page 68 for more information.

Searching a Source File

To search for a string or a line number within a source file, you can add a search toolbar to the Source Browser window or use the *Edit* menu.

To add the search toolbar to the window:

➤ Click *Show Search*, . From the toolbar, you can search for a text string or a line within the source file.

To search for a text string:

➤ Enter a search string in the *Text Search* field, and click *Search Up*, , or *Search Down*, , to find the next or previous occurrence of the string within the file.

The string can include any of the following special characters:

*	Match any number of characters
?	Match a single character

Because SimVision keeps a history of the search strings that you enter, you can enter a partial string and press Tab. If the history contains one matching string, SimVision seeds the *Search* field with that string. If the history contains more than one matching string, it displays a list of those strings, and you can choose one.

Tip

You can also initiate a search by selecting some text in the Source Browser, pressing the right mouse button, and choosing *Search for Selected Text* from the pop-up menu.

To search for a line within the source file:

➤ Enter a line number in the *Go To Line* field and click *Go*, . The Source Browser displays a blue arrow in the left column of the source file at the specified line.

To remove the Search toolbar:

➤ Click *Hide Search*, .

If you need more flexibility when searching for text strings or lines, use the *Edit* menu.

To search for a text string from the *Edit* menu:

1. Choose *Edit – Text Search* from the Source Browser menu. SimVision opens the Text Search form.
2. Enter a search string in the *Find what* field, including the special characters, * and ?.

The Text Search form also lets you specify the following options:

- ❑ Disable *Regular Expression* to specify the exact string you want to find. When this button is disabled, special characters (* and ?) are treated like ordinary characters.
- ❑ Enable *Match Case* if you want the search to be case-sensitive.
- ❑ Enable *Up* or *Down* to control the direction of the search.

3. Click *Find Next* to find each occurrence.
4. Press *Close* to end the search.

To go to a line using the *Edit* menu:

➤ Choose *Edit – Go To Line* from the menu and enter a line number in the Go To Line form.

Accessing Design Objects and Values

Design objects are underlined in the Source Browser. You can select objects in several ways:

- ■ Click on the object.
- ■ `Control`-click to select multiple objects.
- ■ Use the *Select* menu to select specific types of objects, such as scopes, signals and variables, and types of ports.
- ■ Drag-select an area of text to select all of the objects in that area.

After you select objects, you can add them to other windows by clicking the window's *Send To* toolbar button.

When you place the cursor over an object, the Source Browser pops up its full hierarchical name. If you have set a probe on the object, the Source Browser also displays the name of the database to which it is probed and its value at the current time. If you have not set a probe for the object, it is labeled `Not probed`, and its value is shown as `Value Not Available` at all times except the last simulation time. If you move the cursor to that time, all objects have a value, regardless of whether they have been probed.

If the value of an object changes during the current clock cycle, the pop-up message indicates the signal transition at that time point. For example, if the signal transitions from 0 to 1 in the current clock cycle, the pop-up message shows `'b0 -> 'b1`. You can turn off the display of signal transitions for all SimVision windows by disabling *Display signal transition values* in the Signal Options tab of the Preferences window.

You can display object values on a separate line in between each line of source code. Signal transitions are displayed with the same notation as in the pop-up message.

To display object values:

Enable *Display or Hide Values* in the Source Browser toolbar, or enable *View – Display Values* in the Source Browser menu.

To hide object values:

Disable *Display or Hide Values* in the Source Browser toolbar, or disable *View – Display Values* in the Source Browser menu.

Note: When *Watch Live Data* is enabled,  , values in the Source Browser are not updated until the simulation pauses or stops.

Displaying ‘define Macros in the Source Browser

When debugging designs that contain `‘define` macros, it is often necessary to see the value of a macro or to see the macro definition. All instances of a macro definition are underlined in the Source Browser, like other design objects.

Note: The Source Browser can display only the values of macro references that are constants. It cannot display values that are complex expressions.

To show the value of a macro:

- Hover the cursor over the instance, and the Source Browser displays a tooltip that contains the value of the macro and its definition.
- Enable *Display or Hide Values*,  , and the Source Browser displays the macro value on a separate line above the line of source code.

To go to the macro definition:

- Double-click the instance, or right-click the instance and choose *Show Macro Definition* from the pop-up menu.

Performing Functions on Design Objects

When you select an object, you can right-click to pop up a menu of functions that you can perform on the object.

The following menu choices are available in the pop-up menu for scopes:

Bookmark This View	Creates a bookmark for the view currently displayed in the window.
Send to	Lets you add the objects in the scope to a window. If you choose one of the window types in the menu, SimVision adds the scope to the target window of that type. If you choose *Send to new*, you can send the scope to a new window of the type you choose.
Set Debug Scope	Displays the source code for the scope in the Source Browser and makes that scope the current scope.
Create Probe	Opens the Set Probe form, so that you can set a probe on the scope. Chapter 8, “Creating and Managing Probes” for more information on setting probes.
Describe	Displays information about the scope, as returned by the `describe` simulator command.

The following menu choices are available in the pop-up menu for signals and variables:

Bookmark This View	Creates a bookmark for the view currently displayed in the window.
Follow Signal	Lets you look at the source code in other scopes where the signal appears. The submenu lists the scopes from which you can choose. If the signal is local to the current scope, the submenu displays *(nothing above)* and *(nothing below)*. **Note:** If your design contains SystemVerilog implicit port connections, the Source Browser displays the port connections as written—that is, with the `.*` notation. The *Follow Signal* menu choice cannot follow these signals. However, the Schematic Tracer and the Trace Signals sidebar display them as if they were explicitly defined.
Trace Driving Logic	Opens the Trace Signals sidebar and traces the selected signal back to its module inputs.
Trace Loading Logic	Opens the Trace Signals sidebar and traces the selected signal forward to its module outputs.
Send to	Lets you add the object to a window. If you choose one of the window types in the menu, SimVision adds the object to the target window of that type. If you choose *Send to new*, you can send the object to a new window of the type you choose.

Break on Change	Creates a breakpoint on the object. See Chapter 9, “Setting and Managing Breakpoints” for more information on setting breakpoints.
Set Force	Opens the Force Value form, so that you can specify the value that you want to give to the object. See “Forcing and Releasing a Signal Value” on page 139.
Release Force	Releases any forces that you have set on the object and returns it to its original value. See “Forcing and Releasing a Signal Value” on page 139.
Deposit Value	Opens the Deposit Value form, so that you can specify the value that you want to deposit. See “Depositing a Signal Value” on page 141.
Create Probe	Opens the Set Probe form, so that you can set a probe on the object. Chapter 8, “Creating and Managing Probes” for more information on setting probes.
Show Value	Displays the current value of the object in the *simulator* tab of the Console window.
Describe	Displays the full hierarchical name and current value of the selected object. This is the same information that pops up when you position the cursor over the object in the Source Browser.

Setting Breakpoints

The Source Browser provides shortcuts for setting line and signal breakpoints. For more information on setting breakpoints, see Chapter 9, “Setting and Managing Breakpoints.”

To set a line breakpoint:

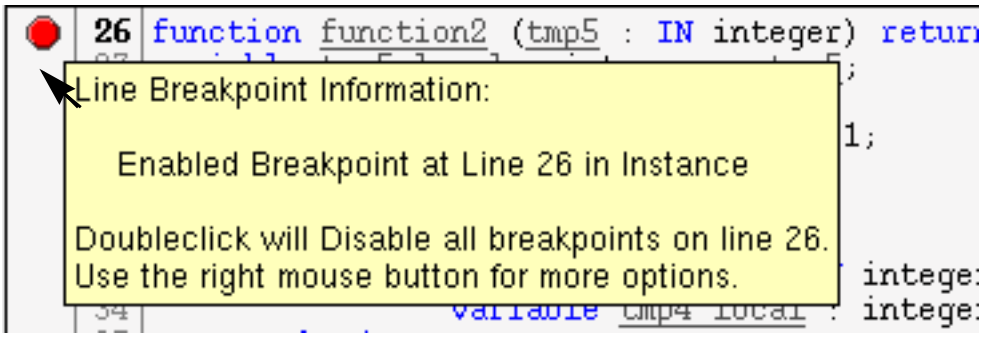
➤ Double-click on the line number, or right-click a line number and choose one of the following pop-up menu entries:

Break at Line	
Break at Line in Instance	Same as double-clicking on the line number
Break at Line in Object	Applies to class objects only

A red stop sign in the left column indicates that a breakpoint is set. You can set more than one type of breakpoint on the same line. However, the Source Browser displays only one

stop sign. Hover the mouse pointer over the stop sign, and a tool tip shows you the breakpoints set on that line, as shown in Figure 4-2 on page 72.

Figure 4-2 Line Breakpoint in the Source Browser

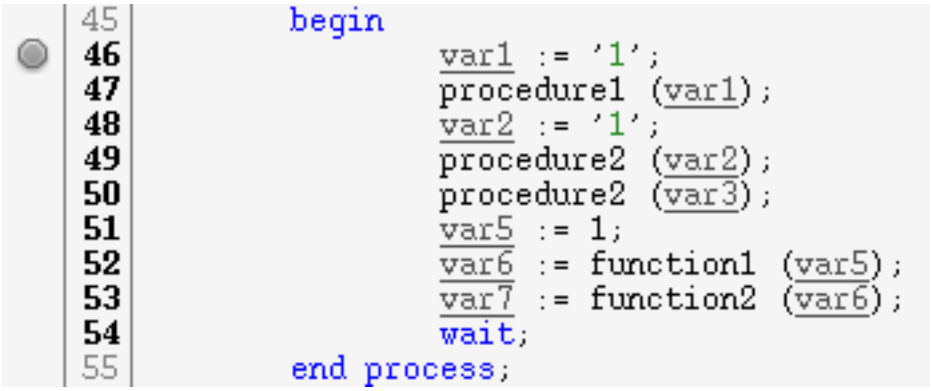


You cannot set a breakpoint on all lines of the source code. The Source Browser displays in black the line numbers for lines that can accept breakpoints; it displays in gray the line numbers for lines that cannot accept breakpoints. See "Preparing Your Design for Simulation" on page 37 for information on making source line numbers available to SimVision.

To disable all breakpoints on a line:

➤ Double-click on the line number or stop sign to disable all breakpoints set on the line. When disabled, the stop sign turns gray, as shown in Figure 4-3 on page 72.

Figure 4-3 Disabled Line Breakpoint



To disable or delete one or all breakpoints:

➤ Right-click the breakpoint and choose one of the following pop-up menu entries:

Delete all Breakpoints on Line <n>	
Disable all Breakpoints on Line <n>	
Delete Line Break in Instance	
Disable Line Break in Instance	
Delete Line Break in Object	Applies to class objects only
Disable Line Break in Object	Applies to class objects only
Delete Line Break	
Disable Line Break	

To set a signal breakpoint:

➤ Select the signal and click *Breakpoint*,  , or right-click and choose *Break on Change*.

When you set a signal breakpoint, no breakpoint icon appears, but you can see the breakpoint definition in the Properties window, as described in "Managing Breakpoints in the Properties Window" on page 138.

Setting Probes

The Source Browser provides the following shortcuts for setting probes:

➤ Select an object and choose *Send to Waveform* from the pop-up menu. This menu choice performs the following operations:

- ❑ Opens the default database, if you have not opened a default database already.
- ❑ Sets a probe on the selected object.
- ❑ Sends the object to the Waveform window. If a Waveform window is not opened, it opens one for you.

➤ Select an object and choose *Create Probe* from the pop-up menu, or choose *Simulation – Create Probe* from the menu bar. SimVision opens the Set Probe form, which you can use to create the probe.

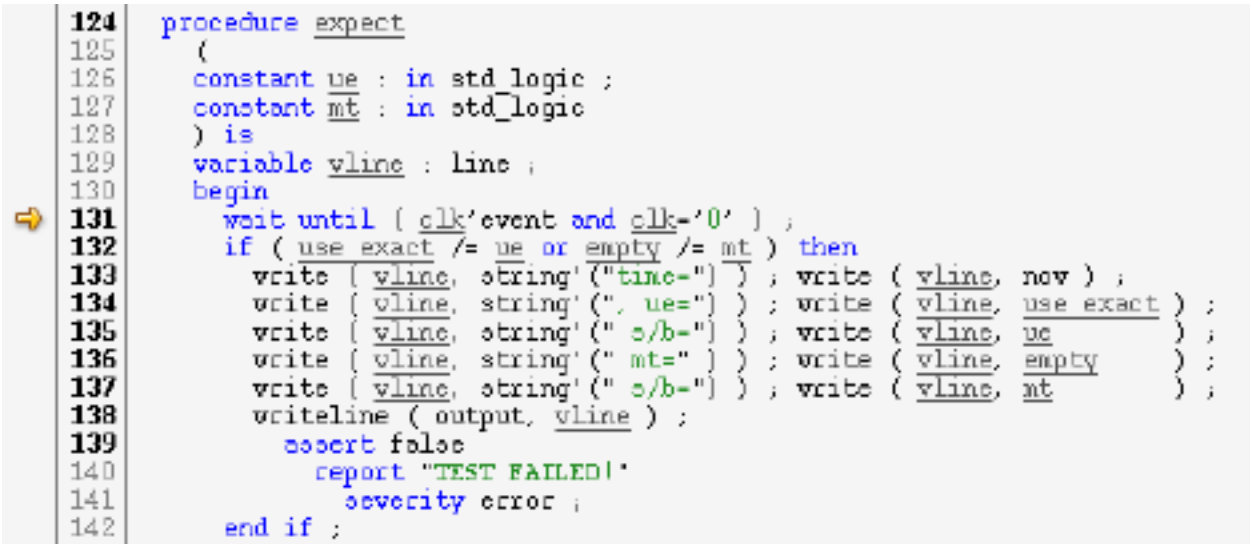
See Chapter 8, "Creating and Managing Probes" for more information on setting probes.

Running the Simulation

You can run the simulation from the Source Browser window, using either the simulation toolbar or the *Simulation* menu. See Chapter 11, “Controlling the Simulation” for more information on using this toolbar and menu.

If the current execution point is known, the Source Browser displays a yellow arrow next to the line of source code that is currently executing, as shown in Figure 4-4 on page 74. Otherwise, the arrow is displayed in gray at the last known execution point.

Figure 4-4 Displaying the Current Execution Point



The Source Browser may not know the current execution point for the following reasons:

- For Verilog designs, it has stopped at the wire resolution phase of the simulation cycle.
- For VHDL designs, it has stopped at the signal evaluation phase.

If the current execution point is not displayed in the Source Browser window, you can quickly move to that point, by clicking *Current Execution Point*, , in the simulation toolbar. If the current execution point is not known, the *Current Execution Point* button is disabled.

Viewing the Call Stack

During simulation, you can view the Verilog and VHDL call stack in the Source Navigation sidebar of the Source Browser window. This tab also functions as a thread manager for

SystemC processes. For information on how to use this tab with SystemC designs, see the *NC-SC User Guide*.

Important

This feature is available only in the target Source Browser window. Make sure you enable the target icon for the Source Browser window in which you want to view the call stack.

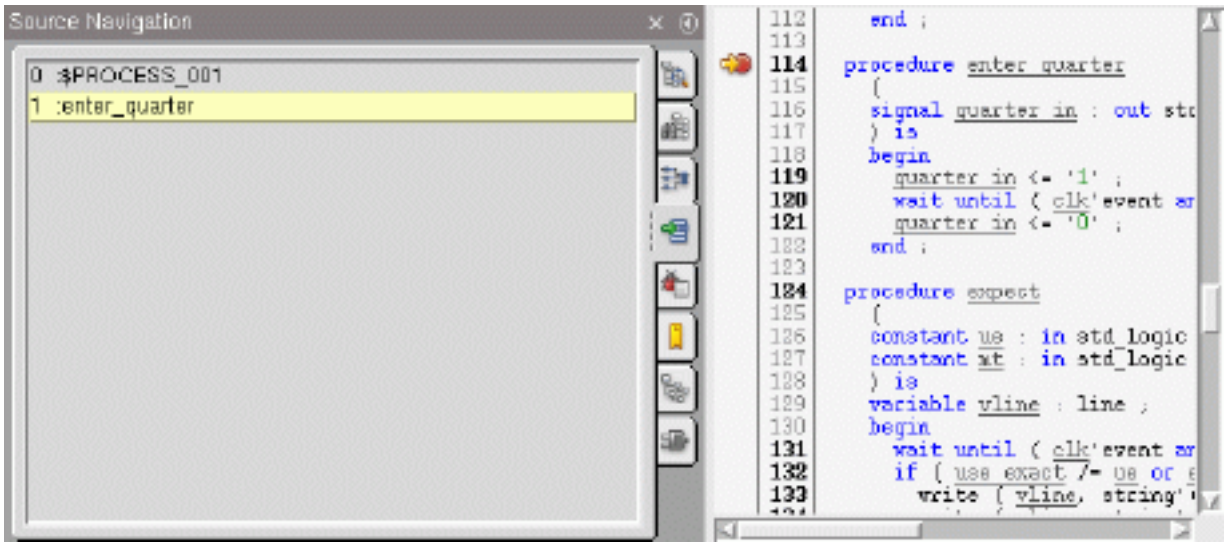
To open the call stack:

Select the *Call Stack* tab in the Source Browser sidebar. The Source Browser displays the call stack, as shown in Figure 4-5 on page 75.

Tip

If the sidebar is not visible, enable the *Sidebar* option in the *View* menu, then click the *Call Stack* tab.

Figure 4-5 Displaying the Call Stack



When simulation enters a Verilog task or function, or a VHDL function, process, or procedure, its name is added to the call stack. When the task, function, process, or procedure exits, its name is removed from the call stack. If there is nothing on the call stack at the current execution point, the Source Browser displays `(call stack empty)`.

When you click on an entry in the call stack, the Source Browser points to its location in the source code.

Note: For Verilog, the sidebar can display the call stack only for the currently executing task or function, and only if the debug scope is set to the scope in which it is executing.

Finding the Cause of a Signal Transition

There might be times when you would like to trace a signal transition to the place in the source code that causes the transition. SimVision is able to go directly to the source code if you are tracing a non-structural object, such as a Verilog register or VHDL variable, and if you have saved statement trace information in the database.

If you do not save statement trace information, or if you are tracing a structural object, such as a Verilog net or a VHDL signal, SimVision can trace the cause of a transition with the Trace Signals sidebar. From the Trace Signals sidebar, you can trace drivers and contributing signals and view them in the Source Browser. For information on the Trace Signals sidebar, see Chapter 16, “Tracing Paths with the Trace Signals Sidebar.”

If you trace a structural object that has no drivers, it is displayed in the Trace Signals sidebar, but no drivers appear below it. When you double-click on the object, a message pops up, stating that no drivers are available.

To save statement trace information:

1. Create a database and enable *Record statement trace information*.

   Although statement trace information can more quickly locate the cause of a problem in your design, it slows the performance of the simulator. The simulator warns you of this when you create the database.

2. Probe the scope in which you want to trace statement information. When setting a probe, you must include all ports, inputs, outputs, and variables. You might also want to include all subscopes.

   You cannot gather statement trace information with an object probe.

3. Simulate the design to generate the simulation data.

To find the cause of a signal transition:

1. In the Waveform window, select a signal and place the primary cursor on the signal transition that you want to trace, or anywhere within the waveform where the signal has the desired value.
2. Choose *Explore – Go To – Cause*. SimVision opens the Source Browser and adds the signal to the Trace Signals sidebar.
3. When *Click and add to source code area* is enabled, you can select a signal in the Trace Signals sidebar and see where it is defined in the source code.

Using Bookmarks in the Source Browser

In the Source Browser, a bookmark saves the source file, line number, and scope displayed in the window. You can use the bookmark to restore the view at a later time. You create, rename, reorder, and delete bookmarks in the Bookmarks sidebar.

Bookmarks are automatically saved when you exit SimVision and restored when you run SimVision again from the same design directory.

 Video

Source Browser Bookmarks

To open the Bookmarks sidebar:

Click *Bookmarks* in the Source Browser sidebar, or choose *View – Manage Bookmarks* from the menu bar.

To create a bookmark:

➤ Click *Add a Bookmark*,  , in the sidebar, or choose *View – Add to Bookmarks* from the menu bar.

The Source Browser assigns a unique name to the bookmark, based on the source file name, line number, and scope.

To rename a bookmark:

➤ Double-click the bookmark to open a text area where you can edit the name; press `Return` to close the text area.

To reorder the list of bookmarks:

➤ Press and hold the middle mouse button over the bookmark, then drag the bookmark to a new location in the list.

To delete a bookmark:

➤ Select the bookmark and click *Delete*, .

To return to a view that you have bookmarked:

➤ Select the bookmark in the sidebar.

The Properties window lists all of the bookmarks that you create. You can use the Properties window to rename and delete bookmarks.

To manage Source Browser bookmarks in the Properties window:

1. Click *Properties*, , in the Send To toolbar and select *Bookmarks* in the lefthand pane of the window, or click *Display Bookmark Properties*, , in the Bookmarks sidebar.
2. Select *Source Browser* in the Bookmarks tab of the Properties window. This displays the Source Browser bookmarks that you have created.
3. You can perform the following operations on bookmarks:
   - ❑ Double-click on a bookmark to edit its name.
   - ❑ Select one or more bookmarks and click *Delete*, , to delete the bookmarks.
   - ❑ Drag and drop bookmarks to change their order in the list.

Editing a Source File

You cannot edit a source file directly in the Source Browser, because it would no longer represent the design snapshot currently loaded into the simulator. However, you can invoke a text editor, either from the Source Browser or outside of the SimVision environment, make the necessary changes, and then create a new snapshot and load it into the simulator.

To invoke a text editor from the Source Browser:

➤ Use the Scope toolbar to display the source code you want to edit, then click *Edit Source*, . The editor that you invoke depends on your preferences setting. See “Source Browser” on page 318 for information on choosing an editor.

After you save the file and reselect the scope for the file, the Source Browser displays the Modified indicator in the Source Browser status bar, as shown in Figure 4-6 on page 79.

Figure 4-6 Modified Indicator in the Source Browser Status Bar

If you have edited the source file after creating the snapshot but before invoking SimVision, the Source Browser displays the Modified indicator, and it displays a message that objects might not line up correctly in the Source Browser as a result of the changes.

To create a new snapshot and load it into the simulator:

➤ Choose *Simulation – Reinvoke Simulator* from the menu bar of any SimVision window. SimVision compiles and elaborates the design, loads the new snapshot into the simulator, and sets simulation time to 0.

For more information, see “Reinvoking the Simulation” on page 147.

5

Accessing Design Objects

The Design Browser and Design Search sidebars let you access the objects in your design. You can use these sidebars to locate the signals and variables that you want to debug. For example, you can use them to select objects to send to the Waveform window or to the Schematic Tracer.

For information on expanding, collapsing, and hiding the sidebars, see “Using the Sidebar” on page 28.

Using the Design Browser Sidebar

The Design Browser sidebar lets you navigate the design hierarchy for the simulation or database that you have loaded into SimVision.

To access the Design Search sidebar:

Click the *Design Browser* tab.

Tip

If the sidebar is not visible, enable the *Sidebar* option in the *View* menu, then click the *Design Browser* tab.

The sidebar contains a scope view, which shows the design hierarchy, plus controls for browsing and filtering the hierarchy, and for displaying signals within the hierarchy. You can display the scope in tree format or list format.

Tree format displays the scopes of the design hierarchically, as shown in Figure 5-1 on page 82. List format displays only the objects at the currently selected level of hierarchy, as shown in Figure 5-2 on page 83.

To switch between tree format and list format:

➤ Enable *View – Show Scope Tree View* for tree format; disable this option for list format.

You can also save this setting as a preference in the Design Browser Preferences tab, as described in “Scope View” on page 317.

Figure 5-1 Design Browser Sidebar in Tree Format

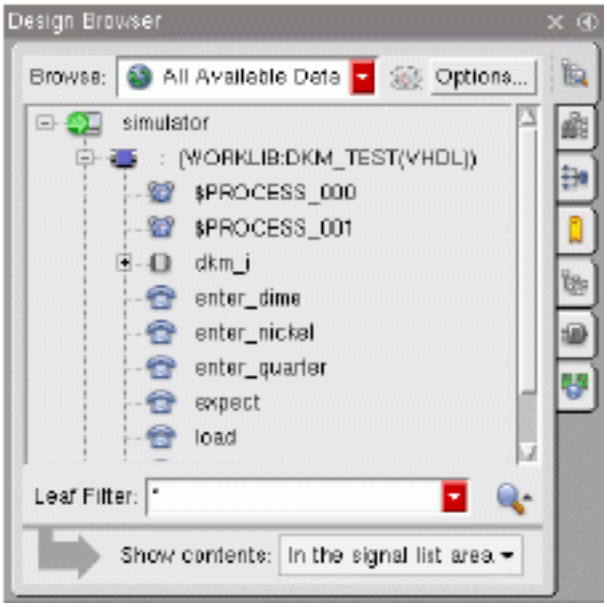
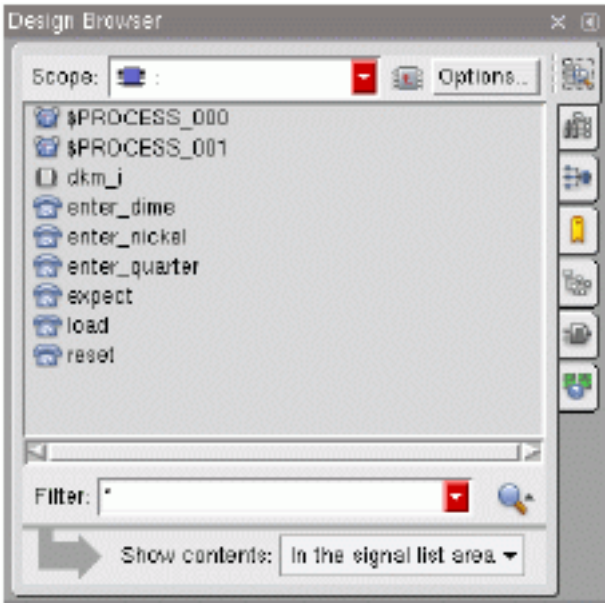


Figure 5-2 Displaying the Scope in List Format



The icons indicate the types of the objects in the scope.

Table 5-1 Scope View Icons

	Verilog module instance
	Verilog package instance
	SystemVerilog interface
	SystemVerilog interface reference
	SystemVerilog modport
	SystemVerilog program block
	SystemVerilog clocking block
	SystemVerilog class
	SystemVerilog class specialization


Verilog or VHDL function


Verilog named block


Verilog task


Verilog generate statement


Procedure statement


VHDL package instance or VHDL block


VHDL block


VHDL component instance


VHDL concurrent assertion


VHDL concurrent procedure call


VHDL concurrent signal assignment


VHDL generate statement


SystemC module


SystemC process


SystemC thread process


OVM instance


e language objects


e language SystemVerilog object


e language generate statement

Selecting a Design Hierarchy to View

When you are connected to a simulation, the top of the design hierarchy is called *simulator*. You can connect to only one simulation at a time.

When you are running a simulation, you might probe objects to one or more databases. These databases all store information about the objects in the simulator design hierarchy. Therefore, by default, the Design Browser sidebar displays a single hiearchy, without regard for the databases where the simulation data is written.

You might also open any number of databases, which contain the results of previous simulations. These databases are not connected to the simulator.

When you invoke the simulator in post-processing mode (by specifying the `-ppe` and `-ppdb` options on the `ncsim` command), the scope view displays the name *design* and the top-level scope of the hierarchy. You can view all of the objects in the database. Probed objects are displayed in regular font; unprobed objects are in italics.

If you want to view the probed objects only, enable the *Show/Hide unprobed objects* button in the signal list.

When you open a database in SimVision (by specifying the *database* argument on the `simvision` command line, or by choosing *File – Open* database from any SimVision menu), the scope view shows the database logical name and the top-level scope of the design hierarchy.

If you load multiple databases from the same simulation session, SimVision automatically overlays the databases over the same design hierarchy.

Thus, at any time you might have one simulator with objects probed to one or more databases, plus any number of databases from other simulations, all loaded into SimVision at the same time. SimVision lets you choose which hierarchies you want to display in the scope view.

To choose the hierarchies that you want to view:

➤ Choose *simulator* from the *Browse* drop-down list to display the hierarchy for the design snapshot you are simulating.

- Choose the name of a database from the *Browse* drop-down list to display the hierarchy for a specific database.
- Choose *All* to display all hierarchies that are loaded into SimVision.

Setting the Root of the Scope View

When you select a hierarchy, the top-level design unit is added to the *Browse* drop-down list. If you select that design unit from the list, its children are added to the list.

To make a subscope the root of the scope view:

- Choose a design unit from the drop-down list. In this way, you can control how much of the design hierarchy is visible in the view.
- Select the design unit in the scope view and choose *Browse from here* from the pop-up menu.

Tip

When in list format, you can double-click on a subscope to make that scope the root of the scope view.

To make a parent scope the root of the scope view:

- Click *Scope Up* ,  . This button moves up the hierarchy one scope at a time.

Tip

When viewing a long list of subscopes in tree format, you can quickly scroll back to the parent scope by selecting the subscope, then right-clicking and choosing *Scroll to parent*.

Expanding and Collapsing a Scope

To expand and collapse the scope:

- Click the + button next to a level of hierarchy, and the hierarchy expands to display the next lower level.
- Click the - button next to a level of hierarchy, and the hierarchy collapses to remove all lower-level hierarchies from the scope view.

➤ Choose *Edit – Explode Scope*, and the hierarchy expands all scopes beginning at the selected scope and ending at the leaf nodes of all subscopes. Locked scopes are not expanded.

When a level of hierarchy does not have a + or - button, it is a leaf node. That is, there are no levels of hierarchy below it. However, if you probe additional scopes during an interactive simulation, a leaf node may turn into a non-leaf node.

Locking and Unlocking a Scope

When a hierarchy is large, you might want to prevent certain scopes from appearing in the scope view. This is done by locking those scopes. When you lock a scope, you cannot expand it, and you cannot search or select any of its subscopes. However, you can still probe its contents, including the contents of its subscopes.

To lock a scope:

➤ Select the scope and choose *View – Lock/Unlock Scope*. The Design Browser sidebar marks the scope with a lock icon.

To unlock a scope:

➤ Select the locked scope and choose *View – Lock/Unlock Scope* from the menu bar, or right-click and choose *Lock/Unlock Scope*.

Filtering the Scope View

When your design hierarchy is large, you might want to filter the scope view to remove those scopes that you are not interested in.

To filter the scope view:

➤ Enter a scope name, partial name, or a glob or regular expression in the *Leaf Filter* field. As you enter filter strings, the Design Browser sidebar adds them to a drop-down list, so that you can quickly switch from one filter setting to another. If you specify multiple strings in the *Leaf Filter* field, separated by spaces, they are ORed.

See Searching for Objects in the *SimVision Command Reference* for information on how to form glob and regular expressions.

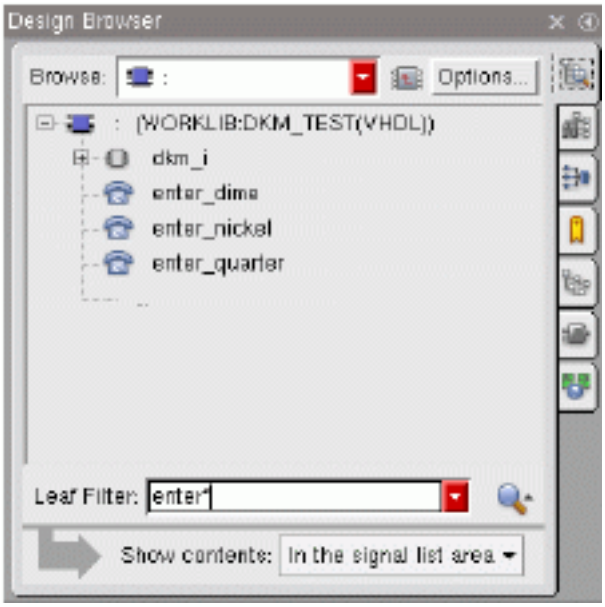
Because SimVision keeps a history of the search strings that you enter, you can enter a partial string and press Tab. If the history contains one matching string, SimVision seeds the *Leaf Filter* field with that string. If the history contains more than one matching string, it displays a list of those strings, and you can choose one.

Important

The *Leaf Filter* field filters only at the leaf-node level because a scope can contain subscopes that match the filter string. Removing higher-level scopes could hide matching subscopes. As a result, the Design Browser sidebar might display scopes that do not match the filter string.

Figure 5-3 on page 88 shows a scope view that has been filtered to show only those scopes whose names begin with the string `load`, plus all higher-level scopes. At those levels of hierarchy where scopes have been removed from the scope view, it displays an ellipsis (`...`).

Figure 5-3 Filtering the Scope View



Searching for Scopes

To add the search toolbar to the window:

➤ Click *Show Search*, . From the search toolbar, you can search for a text string in the scope tree.

To search for a text string:

➤ Enter a search string in the *Text Search* field, and click *Search Up*, , and*Search Down*, , to find the next or previous occurrence of the string.

The string can include any of the following special characters:

*	Match any number of characters
?	Match a single character

Selecting Scopes

To select scopes, use any of the following methods:

- Select a single scope by clicking on it in the scope view.
- Select multiple scopes by `Shift`-clicking on contiguous scopes, or `Control`-clicking on non-contiguous scopes.
- Select a scope and all of its child scopes by clicking on a top-level scope, then do one of the following:
  - Choose *Select – Scope Deep*.
  - Right-click and choose *Select Deep.*
  - Click *Select Scope Deep*,  .

  All of the subscopes of the top-level scope are also selected. *Select Deep* selects all subscopes, regardless of whether they are expanded in the scope view.

Displaying Objects in Selected Scopes

You can choose where you want to display the signals and variables contained in the selected scopes—either in the sidebar or in the window that contains the sidebar.

To display objects in the sidebar:

1. In the *Show contents* field, choose *In the selector below*. Signals are displayed in the sidebar.
2. If the *Click and add to waveform area* button is enabled (the label reflects the type of the containing window), you can send signals to the containing window by clicking on them within the sidebar. If this button is disabled, you can select the signals in the sidebar, then click the *Add* button in the containing window to add them to the window.

3. You can choose which types of signals you want to display by enabling or disabling the following buttons:


Show or hide input ports


Show or hide output ports


Show or hide inout ports


Show or hide internal signals


Show or hide transaction streams


Show or hide assertions


Show or hide unprobed signals

4. To filter signals from the signal list by name, enter the signal or variable name, or partial name, in the *Filter* field. Prefix the search string with ~ to display signals that do not match the filter string.

   Objects whose names match the string are displayed in the signal list; all other objects are removed. As you enter filter strings, the Design Browser sidebar adds them to a drop-down list, so that you can quickly switch from one filter setting to another. If you specify multiple strings in the *Filter* field, separated by spaces, they are ORed. That is, the Design Browser sidebar displays a signal if it matches any of the filter strings.

To display objects in the containing window:

➤ In the *Show contents* field, choose *in the waveform area* (the label reflects the type of the containing window). When you select a scope in the scope view, its signals are displayed in the containing window. If the selected scope contains subscopes, only the signals in the top-level scope are added to the window.

Setting Scope View Options

You can control how scopes are displayed in the scope view, including how they are sorted, the format of their names, and the types of scopes that you want to display.

To set scope view options:

1. Click *Options*. The Design Browser sidebar opens the *Scope View* tab of the Sidebar Options form.
2. In the *General* tab, you can set the following options:
   - Choose how you want to sort scopes in the scope view, either by name or by declaration type.
   - Enable *Show Module/Unit Names* if you want the Design Browser to display module or design unit names.

     Note: *Show Module/Unit Names* does not apply to the ***e*** unit tree. The ***e*** unit tree displays the short names for units at all times.
   - Enable *Show Icons* to display icons that indicate object type, or disable this option to display only the object name.
   - Enable *Show Tree View* to display the scopes of the design hierarchically. Disable this option to display only the objects at the current level of hierarchy.
3. Click the *Verilog* tab and enable the types of Verilog scopes you want to display in the scope view.
4. Click the *VHDL* tab and enable the types of VHDL scopes you want to display in the scope view.
5. Click the *SystemC* tab and enable the types of SystemC scopes you want to display in the scope view.
6. Click the *e* tab and enable the types of ***e*** scopes you want to display in the scope view.

Important

The options you choose affect the sidebar in the current window only. Use the Design Browser Scope View Preferences tab to change these settings for all Design Browser windows, as described in "Scope View" on page 317.

Setting Signal List Options

You can control how signals are displayed in the signal list area of the sidebar, including the types of signals that are displayed, and how they are sorted.

To set signal list options:

1. Click *Options*. SimVision opens the Sidebar Options form.

2. Enable *Signal List* on the left side of the form. SimVision displays the Signal List General options.
3. In the *General* tab, select the signal types that you want to display in the Signal List area of the sidebar.
4. In the *Verilog* tab, select the Verilog signal types that you want to display.
5. In the *VHDL* tab, specify the VHDL signal types that you want to display.
6. In the *SystemC* tab, enable the SystemC signal types that you want to display.
7. In the *e* tab, enable the *e* language scope types that you want to display.

Important

The options you choose affect the sidebar in the current window only.

Using the Design Search Sidebar

The Design Search sidebar lets you search for scopes, signals, and variables across multiple databases without regard for the design hierarchy.

To access the Design Search sidebar:

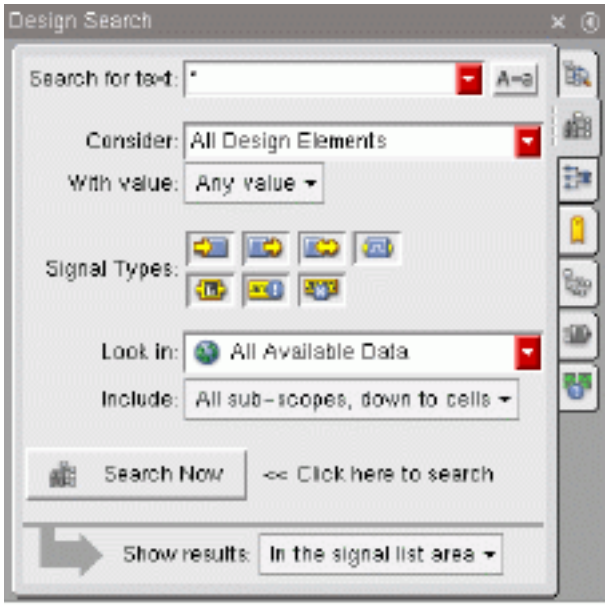
Click the *Design Search* tab.

Tip

If the sidebar is not visible, enable the *Sidebar* option in the *View* menu, then click the *Design Search* tab.

The Design Search sidebar is shown in Figure 5-4 on page 93.

Figure 5-4 Design Search Sidebar



Searching for Objects

To search for objects:

1. Enter a search string in the *Search for text* field. This string specifies the name, or partial name, of the object that you want to find. By default, the search string is *, which matches any name.

   SimVision keeps a history of the search strings that you enter. Therefore, you can enter a partial string and press Tab. If the history contains one matching string, SimVision seeds the *Search for text* field with that string. If the history contains more than one matching string, it displays a list of those strings, and you can choose one.

   You can drag and drop the signal or scope into the *Search for text* field. SimVision adds the simple name of the object, not the full scope path.

Tip

Prefix the search string with ~ to search for signals that do not match the filter string.

2. Enable *Case sensitivity*,  , if you want your search to be case sensitive; disable the button,  , if you want the search to be case insensitive. This button is useful, for example, when debugging a mixed-language design in which the Verilog code is case sensitive and the VHDL code is case insensitive.

3. In the *Consider* field, choose the types of objects you want to search for, as follows:
   - ❑ All Design Elements
   - ❑ Only Signals/Variables
   - ❑ Only Scopes
4. If you want to return only those signals that have a specific value—*X*, *Z*, *0*, 0r *1*—at the current simulation time, choose the value from the *With value* drop-down menu. The default is *Any value*.
5. Enable the types of signals that you want to search for, by enabling or disabling the following buttons:


Show or hide input ports


Show or hide output ports


Show or hide inout ports


Show or hide internal signals


Show or hide transaction streams


Show or hide assertions


Show or hide unprobed signals

6. In the *Look In* field, specify the hierarchy and scopes within that hierarchy that you want to search, in any of the following ways:
   - ❑ From the drop-down menu, choose the simulator or database hierarchy that you want to search, or choose *All Available Data* to search all hierarchies that are loaded into SimVision.

     When you select a hierarchy, its top-level scope is added to the drop-down list. Select this scope to add its sub-scopes to the list. Continue in this way until you reach the scope that you want to search.
   - ❑ Drag a scope from any SimVision window and drop it into the *Look In* field.

- Select a scope in any SimVision window, then right-click and choose *Search from here*.

7. Use the *Include* drop-down menu to further limit how scopes are searched. Choose from the following levels:
   - *Just this level*—Search only the current scope
   - *All sub-scopes*—Search sub-scopes of the current scope, except library cells
   - *All sub-scopes, down to cells*—Ssearch sub-scopes of the current scope, including library cells
8. Click *Search Now* to perform the search. While the search is in progress, the *Search* button is replaced with an *Abort* button. If you want to stop the search before it has completed, click *Abort*.

Displaying the Search Results

You can display the results of the search in the sidebar, or send the signals and scopes to the containing window.

To display the results in the sidebar:

1. In the *Show results* field, choose *In the selector below*. The signals and scopes returned by the search are displayed in the sidebar.
2. If the *Click and add to waveform area* button is enabled (the label reflects the type of the containing window), you can send the results to the containing window by clicking on them within the sidebar. If this button is disabled, you can select the signals in the sidebar, then click the *Add* button in the containing window to add them to the window.

To display results in the containing window:

➤ In the *Show results* field, choose *In the waveform area* (the label reflects the type of the containing window). When you perform a search, the signals and scopes are sent to the containing window.

6

Monitoring Signal Values

The Design Browser lets you monitor signal value changes, during a running simulation or in a simulation database.

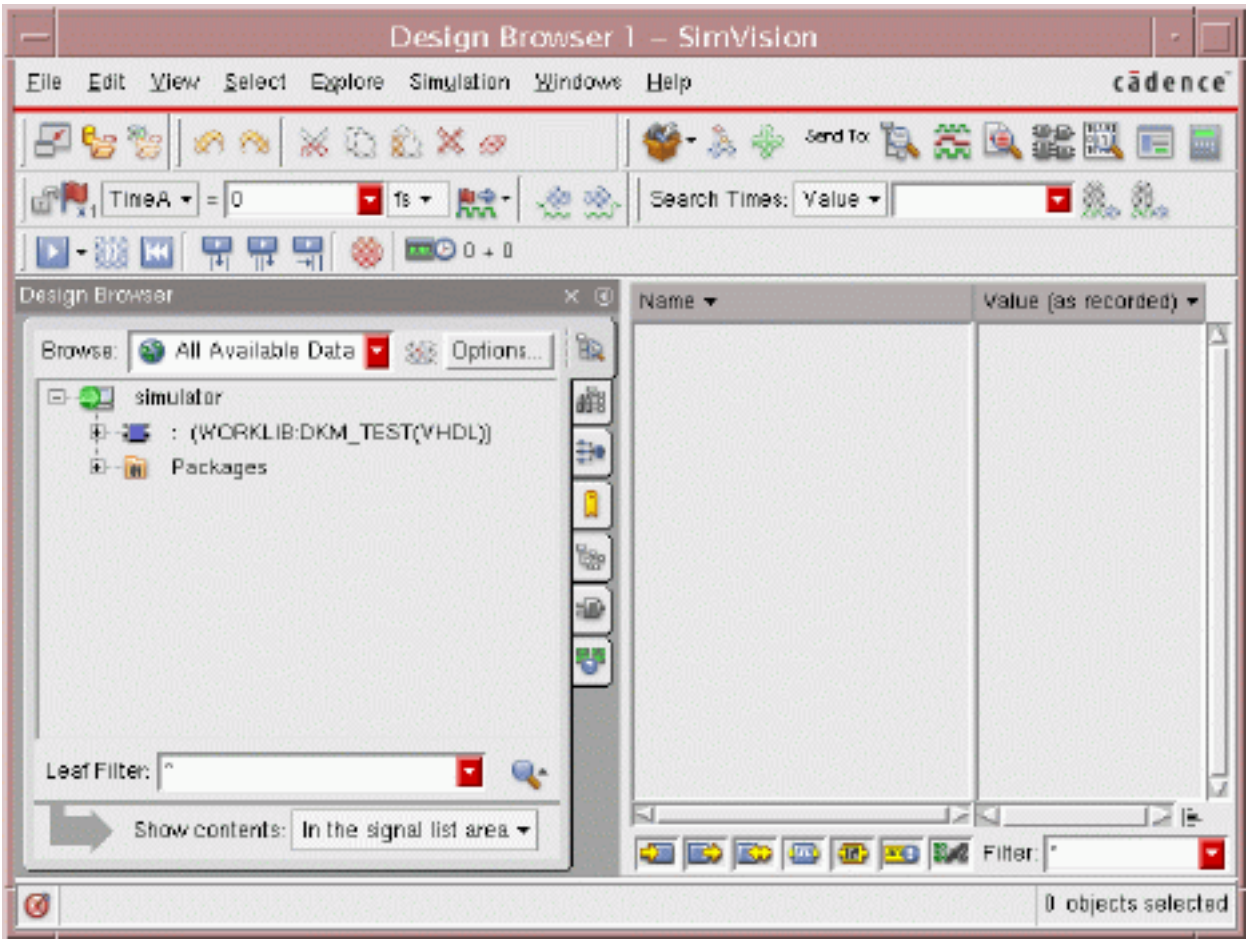
Opening a Design Browser Window

By default, SimVision opens one Design Browser window when it starts up, as shown in Figure 6-1 on page 98.

To open additional windows:

- In the Send To toolbar of any SimVision window, click and hold the *Design Browser* button,  , and choose *New Browser*, or choose *Windows – New – Design Browser* from the menu bar.

Figure 6-1 Design Browser Window



The left side of the window contains the Design Browser sidebar, which lets you select the objects in the design. These are added to the signal list in the right side of the window. You use the signal list to monitor the signal values. See Chapter 5, “Accessing Design Objects” for information on the Design Browser sidebar.

Video

Monitoring Signal Values

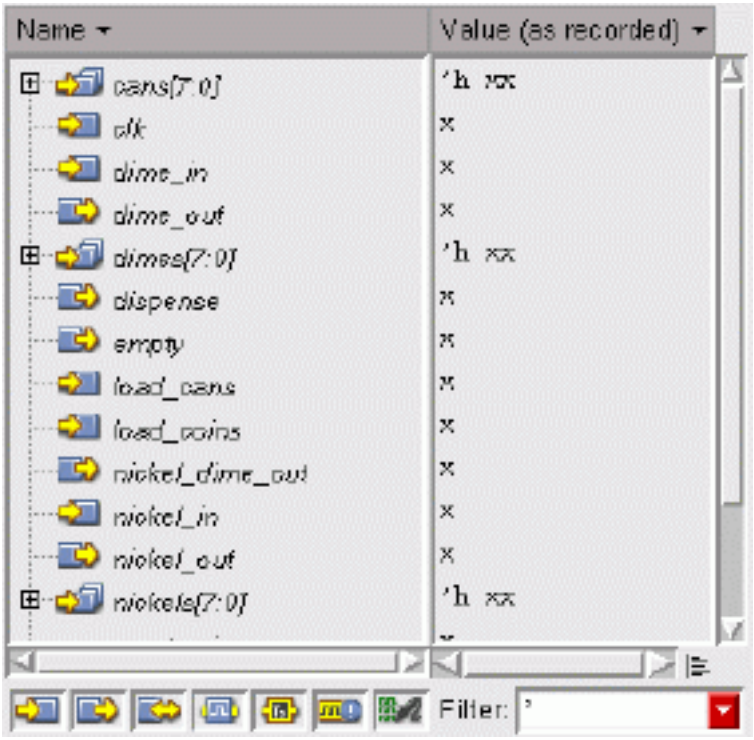
Displaying Signals and Variables

To select the signals that you want to monitor, use either of the following methods:

- In the Design Browser sidebar, set the *Show contents field* to *In the signal list area*. When you select a scope, all of the signals in that scope are added to the signal list.
- In the Design Browser sidebar, set the *Show contents field* to *In the selector below*. When you select a scope, its signals are displayed in the sidebar. If you enable *Click and add to signal list area*, individual signals within the scope are added to the signal list as you select them. If this button is disabled, you can drag and drop signals into the signal list.

The Design Browser window displays the names and values of signals in the signal list, as shown in Figure 6-2 on page 99.

Figure 6-2 Signals and Variables in the Signal List



The left column shows the signal names; the right column shows the signal values at the primary cursor time. Probed objects are shown in regular font; unprobed objects are shown in italics. If the signal value changes at the current cursor time, the transition is displayed with the notation `old-value -> new-value`.

A button sometimes appears below the *Name* column when the hierarchy becomes so deep that the expand and compress buttons (+ and -) reach the right side of the column. You can use this button to scroll through the signal names, as follows:

 Click the right or left side of the pyramid to shift the hierarchy to the left or right.

Note: For probed objects, signal values are available for all simulation times that the probe is in effect. For unprobed objects with read access, signal values are available only for the current simulation time, if *Watch Live Data*,  , is enabled.

Note: When a signal is probed to a database, you can go back in time, and the Design Browser displays the values for the signals at that time. For analog buses, the Design Browser shows the value for the whole bus and the values for the individual bits of the bus. However, bit values in the whole bus representation and in the individual bit representation are different. This is because SimVision interpolates values for individual bits, whereas the whole bus value is taken from the solution point. When you move the cursor in the Waveform window, individual bits show the correct interpolated values.

Sorting the Signal List

Signals appear in the signal list in the order in which you add them.

To sort the signals in the signal list:

➤ Choose *View – Sort – By Name* to sort the signals in alphabetical order.

➤ Choose *View – Sort – By Declaration* to sort the signals by declaration type.

Choosing the Format of Signal Names

You can display signal names in one of the following formats:

- *Name*—Displays only the signal name
- *Path.Name*—Displays the full hierarchical signal name
- *Db::Name*—Displays the database name and signal name
- *Db::Path.Name*—Displays the database name and full hierarchical signal name

To change the format of signal list names:

➤ Select the desired format from the drop-down list above the signal names.

Choosing the Radix for Signal Values

By default, the Design Browser displays signal values in the radix in which they were recorded.

To change the radix:

➤ Choose a radix from the drop-down list in the *Value* column.

Justifying the Signal Values

The button below the *Value* column indicates whether the column is left-justified or right-justified. Click the button to toggle between the two choices. The button changes, as follows:


Indicates that the *Value* column is left-justified.


Indicates that the *Value* column is right-justified.

Filtering Signals in the Signal List

For each object, the Design Browser window indicates whether the object is an input, output, input/output, internal signal, transaction stream, or assertion. You can choose which types of signals you want to display by enabling or disabling the following buttons:


Show or hide input ports


Show or hide output ports


Show or hide inout ports


Show or hide internal signals


Show or hide transaction streams


Show or hide assertions

Show or hide unprobed signals

To filter signals from the signal list by name, enter the signal or variable name, partial name in the *Filter* field. Prefix the search string with ~ to display signals that do not match the filter string. Objects whose names satisfy the search criteria are displayed in the signal list; all other objects are removed.

Tip

If you want to include wildcard characters, you must enable *Filter using regular expressions* in the Design Browser Signal List tab of the Preferences window. See Searching for Objects in the *SimVision Command Reference* for information on how to form a regular expression.

As you enter filter strings, the Design Browser window adds them to a drop-down list, so that you can quickly switch from one filter setting to another. If you specify multiple strings in the *Filter* field, separated by spaces, the Design Browser window displays a signal if it matches any of the filter strings.

Tip

By default, the Design Browser updates the signal list incrementally, that is, after each character that you enter in the *Filter* field. If you want the signal list to be filtered only after you have finished entering the entire search string, terminated by a `Return`, disable the *Incremental Signal Filtering* option in the Design Browser tab of the Preferences window.

SimVision uses the notation `more` to indicate that some objects have been filtered from the signal list.

Viewing Aggregate Signals

When you view aggregate signals in the Design Browser, such as arrays, VHDL records, and SystemVerilog structures, you might want to see just the name of the signal or expand the signal to see the individual elements. The Design Browser provides several ways to expand and collapse aggregate signals. When you expand a large signal with many elements, you can create a scrollable region, so that the object takes up less space in the window. In addition, you can sort the elements of a queue, or a dynamic or associative array in either 0-first or reverse index order.

Video

Viewing Aggregate Signals

Expanding and Collapsing Aggregates

Note: Queues, and dynamic and associative arrays can be expanded only when *Watch Live Data*, , is enabled.

You can expand and collapse objects in several ways.

From the *Edit* menu:

- To expand an object, select it and choose *Edit – Expand Signal – Expand* from the menu bar.
- To collapse an object, select it and choose *Edit – Expand Signal – Collapse* from the menu bar.

When *View – Show Values* is enabled:

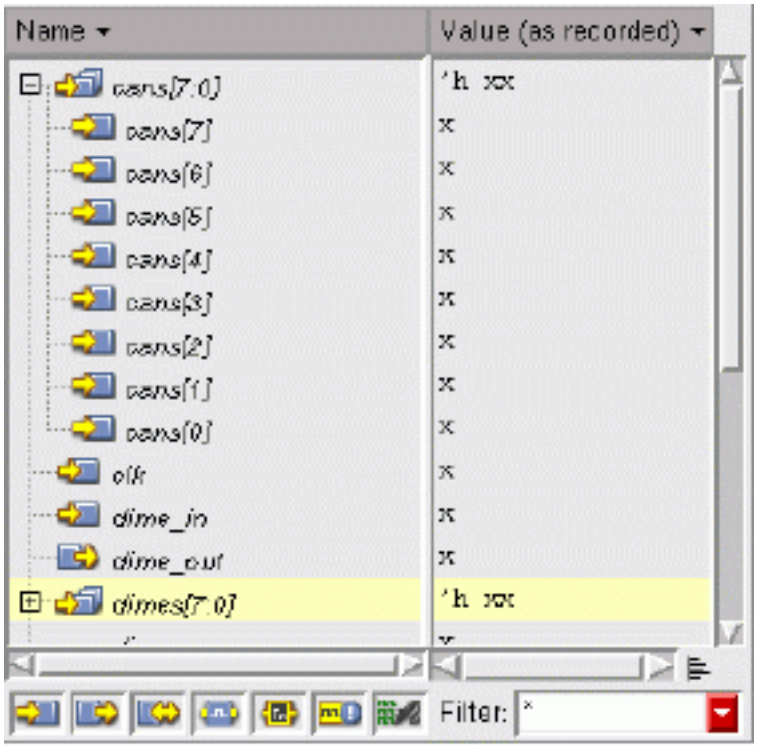
- To expand an object, click on the + button next to the signal, or double-click on the signal name.
- To collapse an object, click on the - button next to the signal, or double-click on the signal name.
- Right-click the signal name and choose *Expand – Expand*.

When *View – Show Values* is disabled:

- To expand or collapse an object, double-click on its name.
- Right-click the signal name and choose *Expand – Expand*.

When a signal is expanded, you see each element of the object. For example, Figure 6-3 on page 104 shows two arrays. The one named `cans` is expanded; you can see each bit of the array. The one named `dimes` is collapsed. In this figure, *View – Show Values* is enabled, so that you can see both the signal names and their values at the current simulation time.

Figure 6-3 Expanding an Array



Some aggregate types give you different options for expanding objects of those types. For example, you can expand a VHDL record to *Record Elements*, and packed structures to low-level elements, such as *Bits* and *Bytes*, as well as higher-level *Structure Elements*.

Splitting a Signal

You can split aggregates composed of bits—such as bit vectors, integers, and SystemVerilog packed structures—into bits and bytes, or a width that you define. Whenever you choose a new width, it becomes the default width for all subsequent expand operations on that signal.

There are several ways to split a signal.

From the *Edit* menu:

➤ Select the object that you want to split, and enable the width in the *Expand Signal* pull-right menu.

Note: If the object you have selected is not expanded, choosing a width does not expand the object. However, if the object is expanded, it is displayed with the new width.

When *View – Show Values* is enabled:

➤ Right-click the + button next to the signal that you want to split, and choose a width.

Note: If the object is not already expanded, choosing a width expands the object at the same time. If the object is expanded to a different width, it is displayed with the new width.

When *View – Show Values* is disabled:

➤ Right-click the signal name and choose the width that you want to use from the *Expand* pull-right menu.

Note: If the object is not already expanded, choosing a width does not expand it. However, if the object is already expanded, it is displayed with the new width.

To define your own width:

1. Right-click the + button next to the signal, and choose *User-Defined Width*. SimVision opens the Enter Desired Width form.
2. Enter the width and click *OK*. SimVision splits the signal to the desired width, and also adds the new width to the pop-up menu.

Creating a Scrollable Region

Aggregates can be quite large, and you might not want to display an entire object at once. You can make an aggregate scrollable and display only a portion of it at a time.

Note: When you expand SystemVerilog queues, and dynamic and associative arrays, the scrollable region is enabled by default.

To make an aggregate scrollable:

1. Enable *View – Show Values* in the menu bar.

   Note: You cannot create a scrollable region when *View – Show Values* is disabled.

2. Right-click the + icon next to the object and enable *Scrollable View* in the pop-up menu, or select the object and choose *Edit – Expand – Scrollable View* from the menu bar. The Design Browser displays a portion of the aggregate.

To adjust the size of the scrollable region:

1. Move the cursor over an edge of the scrollable region, and the Design Browser displays a box around the region. In the lower left corner of the box, notice the sizer button.
2. Drag the sizer up or down to increase or decrease the size of the scrollable region.

To scroll a region:

1. Move the cursor to the right side of the scrollable region. The Design Browser displays a scroll bar.
2. Drag the cursor up and down within the scroll bar. Scrolling speeds up as the cursor moves toward the top or bottom of the scroll bar, and it slows down as the cursor moves closer to the middle of the scroll bar. Click on the arrows at the top and bottom of the scroll bar to scroll one line at a time.

Making the Design Browser Window Compact

After you have added all of the signals you want to see in the Design Browser window, you can reduce the window size. When the Design Browser window is compact, it acts more like a watch window, in which you can monitor signal values as they change during simulation.

To make the window compact:

➤ Click *Compact*,  .

When the window is compact, the Design Browser sidebar is removed, the menu bar is removed, all toolbars are removed except for the cursor control toolbar, and the width of the window is reduced to the size of the signal list.

 *Tip*

You can add toolbars by right-clicking over any blank area or separator in the toolbar, and enabling the toolbar in the pop-up menu.

Using Bookmarks in the Design Browser

In the Design Browser, a bookmark saves the scope displayed in the window. You can use the bookmark to restore the view at a later time. You create, rename, reorder, and delete bookmarks in the Bookmarks sidebar.

Bookmarks are automatically saved when you exit SimVision and restored when you run SimVision again from the same design directory.

 Video

Design Browser Bookmarks

To open the Bookmarks sidebar:

Click *Bookmarks* in the Design Browser sidebar, or choose *View – Manage Bookmarks* from the menu bar.

To create a bookmark:

➤ Click *Add a Bookmark*, , in the sidebar, or choose *View – Add to Bookmarks* from the menu bar.

The Design Browser assigns a unique name to the bookmark, based on the scope.

To rename a bookmark:

➤ Double-click the bookmark to open a text area where you can edit the name; press `Return` to close the text area.

To reorder the list of bookmarks:

➤ Press and hold the middle mouse button over the bookmark, then drag the bookmark to a new location in the list.

To delete a bookmark:

➤ Select the bookmark and click *Delete*, .

To return to a view that you have bookmarked:

➤ Select the bookmark in the sidebar.

The Properties window lists all of the bookmarks that you create. You can use the Properties window to rename and delete bookmarks.

To manage bookmarks in the Properties window:

1. Click *Properties*, , in the Send To toolbar and select *Bookmarks* in the lefthand pane of the window, or click *Display Bookmark Properties*, , in the Bookmarks sidebar.
2. Select *Design Browser* in the Bookmarks tab of the Properties window. This displays the Design Browser bookmarks that you have created.
3. You can perform the following operations on bookmarks:
   - ❑ Double-click on a bookmark to edit its name.
   - ❑ Select one or more bookmarks and click *Delete*, , to delete the bookmarks.
   - ❑ Drag and drop bookmarks to change their order in the list.

Setting Design Browser Preferences

To choose the information you want to display in the Design Browser window:

1. Enable or disable *View – Show Values* to add or remove signal values from the signal list. By default, the Design Browser displays signal values if you are connected to a simulator.
2. Choose *Edit – Preferences* to open the Design Browser tab of the Preferences form.
3. Set Design Browser preferences, Scope View preferences, and Signal List preferences as described in Chapter 25, “Setting Preferences.”

Important

The options you choose in the Preferences window are applied to the current Design Browser window, as well as any new Design Browser windows that you open.

7

Managing Simulation Databases

Simulation databases store information about the signal transitions that occur during simulation. You can use these files to debug your design in post-processing mode, rather than while running the simulation.

SimVision creates databases in SST2 format. An SST2 database is made up of the following files:

- Design file—Contains information about the design hierarchy and connectivity. Design files have the `.dsn` extension.
- Transition file—Records information about signal values and how they change during simulation. Transition files have the `.trn` extension. Each transition file references a design file. You can create multiple transition files for different simulation times; each transition file references the same design file.
- Statement Trace file—Contains information that lets you trace the cause of a signal transition to its location in the source code. This information is not generated unless you specifically request it, because it significantly increases the overall size of the simulation database. Statement trace files have the `.stc` extension.

These files are stored in a database directory that contains only those files. By convention, this directory has the `.shm` extension.

You are not limited in the number of transition files that you can create during simulation. Creating several files, called incremental files, can be useful in some situations. For example, if your design is large, you might want to save simulation data for each component of the design in its own database. Using this method, you can debug the design components separately before debugging the interfaces between them. You can use system tasks to split your transition file in this way.

You can also specify a maximum file size for the transition file. When a transition file approaches the maximum size, the simulator creates a new file. The simulator gives each new file the same base name as the original transition file, and it appends an incremental number to the name. For example, if the original file is named `ncsim.trn`, the first incremental file is named `ncsim-1.trn`, the next is named `ncsim-2.trn`, and so on.

Whenever a new transition file is created, a new statement trace file is also created. It is given the same incremental number as its corresponding transition file, such as `ncsim-1.stc`.

Breaking up large databases into incremental files can improve SimVision's performance and memory usage, because you can load into SimVision only those files that contain the information you want to view. You can open incremental files individually, by specifying their names, or you can open all of them at once, by specifying the directory in which they are stored.

When you load multiple incremental files into SimVision, the data is merged. That is, you see a single waveform for a signal, even though the simulation data is split across multiple files.

You can also merge incremental databases by concatenating the files, such as with the `cat` command on UNIX systems. However, when you merge the files in this way, you lose the ability to load them incrementally.

Note: Creating incremental files by calling system tasks is supported in both Verilog-XL and the Incisive simulator. Creating incremental files automatically by specifying a maximum file size is supported only by the Incisive simulator; it is not supported by Verilog-XL.

As of the IUS5.3 release, the size of an SST2 database has been reduced by an average of 40%. SimVision can still read databases created with earlier versions of SimVision, but earlier versions of SimVision cannot read these new, optimized databases.

If you have a database created by an earlier version of SimVision, you can still load it into later versions of SimVision. However, if you create an optimized database, it cannot be read by earlier versions of SimVision or by other tools that require the larger database format, such as Signalscan.

If you need to create a database to be read by these tools, set the following environment variable to turn off database optimization:

```
setenv CDS_SHM_COMPATIBILITY 5.1
```

Creating an SST2 Database

SimVision creates a database automatically when you set a probe or when you send a design object to the Waveform window. It names the database directory `waves.shm`, and it uses default database options. For example, it does not record event and statement trace information.

You an create a database when you do not want to use the default settings. The options you can set when creating a database differ, depending on the simulator you are using:

Incisive simulator	See "Creating a Database with the Incisive simulator" on page 111.
Verilog-XL	See "Creating a Database with Verilog-XL" on page 112.

Creating a Database with the Incisive simulator

To create a database with the Incisive simulator:

1. Choose *File – Create Database* from any SimVision window. SimVision opens the New Database form.
2. Specify either the logical name for the database, or the name of the database file. During simulation, you refer to the database by its logical name.

   If you do not specify a logical name, SimVision generates the logical name by using the database filename without the extension. If you do not specify a database filename, SimVision generates a filename by adding the `.shm` extension to the logical name.
3. Disable *Open as default database* if you do not want this to be the default database. Objects that you probe are automatically written to the default database, unless you specify a different database in the Probe form.
4. Enable *Record all events* if you want to write all value changes to the database. This is useful if you need to debug race conditions or locate where a signal is assigned multiple values during the same clock cycle.
5. Enable *Record statement trace information* if you want to trace the cause of a transition in the Source Browser. If you do not record statement trace information, SimVision traces the cause of the transition in the Trace Signals sidebar.
6. Enable *Maximum database size* to set a limit on the size of the database.

   By default, there is no size limit. The value that you enter in the text field must be a positive integer that indicates the number of bytes.

   SST2 databases exist as chunks that vary in size from about 2 MB to 4 MB. Thus, the minimum possible size of the database is somewhere between 2 MB and 4 MB. If you set the maximum size to less than this approximate range of sizes, the resulting database size can still be up to 4 MB.

   When the maximum size that you specify is about to be exceeded, the simulator maintains the size limit by discarding an entire chunk (2–4 MB) of the earliest recorded

values. This means that if the maximum size that you specify is greater than 4 MB, the database size will be below the limit, but it might be up to 4 MB below the limit.

7. Enable *Maximum database compression* to compress the SST2 database. Enabling this button produces a smaller database, but it takes more time and memory to write the database.
8. Enable *Incremental database size* to specify the maximum number of megabytes that you want to store in each incremental database file. Because the file size is checked on simulation time changes, and buffered data is subjected to further compression, the maximum database size is not an exact size.
9. Enable *Maximum number of incremental files* to specify the maximum number of incremental files that you want to keep for the database. When you exceed this number, the simulator deletes the oldest file.

Creating a Database with Verilog-XL

To create a database with Verilog-XL:

1. Choose *File – Create Database* from any SimVision window. SimVision opens the New Database form.
2. Enter a database filename. If you do not specify a name, SimVision gives the database the default name, `waves.shm`.
3. Enable the *Record all events* button if you want to write all value changes to the database. This is useful if you need to debug race conditions or locate where a signal is assigned multiple values during the same clock cycle.
4. Enable the *maximum database size* button to set a limit on the size of the database.

   By default, there is no size limit. The value that you enter in the text field must be a positive integer that indicates the number of bytes.

   SST2 databases exist as chunks that vary in size from about 2 MB to 4 MB. Thus, the minimum possible size of the database is somewhere between 2 MB and 4 MB. If you set the maximum size to less than this approximate range of sizes, the resulting database size might still be up to 4 MB.

   When the maximum size that you specify is about to be exceeded, the simulator maintains the size limit by discarding an entire chunk (2–4 MB) of the earliest recorded values. This means that if the maximum size that you specify is greater than 4 MB, the database size will be below the limit, but it might be up to 4 MB below the limit.

5. Enable the *maximum database compression* button to compress the SST2 database. Enabling this button produces a smaller database, but it takes more time and more memory to write the database.

Opening an SST2 Database

When you open a transition file, SimVision also loads in the corresponding design file. If it cannot find a `.dsn` file with the same filename as the `.trn` file, SimVision prompts you for the design filename. SimVision also opens the corresponding statement trace file, if one exists.

To open a database:

1. Click *Open Database,*  *,* or choose *File – Open Database* from the menu bar in any SimVision window.

   The path shown in the *Directory* field is determined by the `SIMVISION_WORKDIR` environment variable. If you have not set this variable, the path is the last directory visited by any of the file browser forms. You can navigate to other directories to find the database that you want to open.

2. Double-click the filename, or select a file and click *Open*.

 *Tip*

The Open Database form can open only one database file at a time. If you have multiple database files in a single directory, you can open them all at once by entering the `database open` command in the *SimVision* tab of the Console window.

Converting a Database to SST2 Format

When you open a database that is not in SST2 format, SimVision converts it to SST2 format.

SimVision can convert any of the following database formats:

- VCD
- HSPICE list
- HSPICE transient output
- Nutmeg

- Epic
- Qsim

To convert a database to SST2 format:

1. Click *Open Database*,  , or select *File – Open Database* from the menu bar.
2. Choose the type of files that you want to display in the Open Database form.

   For example, if you select *VCD Files (*.vcd)*, the form displays any database files with the `.vcd` extension in the current directory.

   The path shown in the *Directory* field is determined by the `SIMVISION_WORKDIR` environment variable. If you have not set this variable, the path is the last directory visited by any of the file browser forms. You can navigate to other directories to find the database that you want to open.
3. Select the database file from the file list and click *Open*.

   SimVision opens the File Translation form. The contents of the form differ, depending on the database format.
4. Select the options that you want for translating the file, as follows:
   - Enter the filename of the translated database in the *Destination* field. By default, the database files are given the same name as the original database, plus the `.trn` and `.dsn` extensions. You can click *Browse* to select an existing file in your directory hierarchy, or enter a new filename in a file selection form.
   - For VCD databases, specify whether you want to translate the entire database or a range of simulation time.
   - Specify whether you want to translate sequence time information, if you have saved this information in your original database. See “Viewing Events in Sequence Time” on page 255 for information on saving and viewing sequence time.
   - Specify whether you want to compress the translated database. Enabling this option produces a smaller database at the expense of taking more time to translate. It also uses more memory during translation, but it does not impair the reading of the database after it is translated.
5. Click *OK* to translate the database and load it into SimVision.

Exporting a Database

Although SimVision reads databases only in SST2 format, you can export a database to any of the following formats:

- SST2
- Value change dump (VCD)
- Comma-separated values (CSV)

Note: Exporting 9-state VHDL `std_logic` or `std_ulogic` values to VCD format with SimVision's *File – Export* feature or the `simvisdbutil` command does not produce the same results as the simulator's `database -vcd` Tcl command. The simulator supports the 4-state values as defined in the IEEE 1364-2001 VCD specification; SimVision retains the 9-state logic values. The differences between the simulator and SimVision are shown in Table 7-1 on page 115.

Table 7-1 Mapping 9-State VHDL Values

	Simulator	SimVision
'U'	X	U
'X'	X	X
'0'	0	0
'1'	1	1
'Z'	Z	Z
'W'	X	W
'L'	0	L
'H'	1	H
'-'	X	-

You can map these 9-state values in the simulator by enabling *Convert VHDL MVL9 to Verilog 4-state* in the Export Variables form, or by running `simvisdbutil` with the `-CVTMVL9` option.

To export a database to SST2, VCD, or CSV format:

1. Choose *File – Export* from the Design Browser or Waveform window menu bar. SimVision opens the Export Variables form.

2. Enter a name for the exported database file in the *Output File* field, or choose a file by clicking *Browse* (`...`). By default, databases are assigned extensions according to their format:

Database Format	Extension
SST2	`.trn`
VCD	`.vcd`
CSV	`.csv`

3. Choose whether you want to export all selected variables or all recorded variables. When you select a scope, all variables under the selected scope are exported.
4. Choose whether you want to export a range of times or all times. You can select the time units from a drop-down list.
5. Select the format of the exported database from the *Database Format* drop-down list.
6. Select other options, depending on the format of the exported database:

Database Format	Other Options
SST2	Choose whether you want to include sequence time information, and whether you want to compress the exported database.
VCD	Choose whether you want to convert 9-state VHDL values to Verilog 4-state values, and whether you want to add indices to memories and VHDL vectors.
CSV	Click *Advanced CSV Options* to specify the options. The Advanced CSV Options form lets you choose when data is sampled in the CSV database—at signal transitions, at specified time intervals, or when a logical expression evaluates to true. It also lets you specify the time units and radix of the data, and a string to use when writing X values. If you enable *Logical Expression Selection*, the drop-down list contains all of the expressions you have created during the SimVision session. You can choose one of these expressions. If the list is empty, first create an expression, as described in Creating an Expression on page 238.

7. Enable *Save simvisdbutil command line* if you want to be able to perform the same export operation at a later time in batch mode. If you want to save the command line without exporting the database, click *Command Line Only*.
8. If you want to save these settings as the default, click *Save*. The settings are saved along with your other SimVision option settings.

Using the Batch Database Translation Utility

The `simvisdbutil` batch database translation utility translates databases through a command-line interface rather than the SimVision graphical user interface. You can place the `simvisdbutil` command in a script file, for example, to translate large databases overnight.

The `simvisdbutil` utility can translate any of the following database formats into SST2 format:

- VCD
- EVCD
- HSPICE list
- HSPICE transient output
- Nutmeg
- Epic
- Qsim

It can also translate an SST2 database into either VCD or CSV format.

Note: Exporting 9-state VHDL `std_logic` or `std_ulogic` values to VCD format with SimVision's *File – Export* feature or the `simvisdbutil` command does not produce the same results as the simulator's `database -vcd` Tcl command. The simulator supports the 4-state values as defined in the IEEE 1364-2001 VCD specification; SimVision retains the 9-state logic values. The differences between the simulator and SimVision are shown in Table 7-1 on page 115.

simvisdbutil Command Syntax

The `simvisdbutil` command has the following syntax:

```
simvisdbutil [options ...] inputFile ...
```

The command accepts the following options:

`-64BIT`	Invokes the 64-bit version of the utility. By default, `simvisdbutil` runs in 32-bit mode.
`-ADDINDICES`	Adds indices to VHDL vectors and memories. This option is for databases in VCD format. **Note:** This option produces a non-standard VCD database.
`-APPEND_LOG`	Appends messages to an existing log file. If you do not use this option, the previous log file is overwritten.
`-COMPRESS`	Compresses the SST2 output file. This option produces a smaller database, but it takes more time and more memory to write the database.
`-CSV`	Generates a database in CSV format. When you use this option, the input file must be in SST2 format. You can also specify this format by including the `.csv` file extension in the argument to the `-OUTPUT` option.
`-CVTMVL9`	Converts 9-state VHDL `std_logic` or `std_ulogic` values to Verilog 4-state values. This option is for databases in VCD format. By default, SimVision retains the 9-state values. For more information, see Exporting a Database.
`-DELAY` *`time`*	Adds the specified delay (or offset) to CSV expression sample times.
`-EXPRESSION` *`expr`*	Uses the specified expression to determine CSV sample points.
`-FILE` *`argfile`*	Reads `simvisdbutil` command-line options from the specified file. To specify multiple argument files, use one `-FILE` option per file.
`-HELP`	Displays a brief description of all command-line options.
`-LOGFILE` *`filename`*	Specifies the name of the file where messages are written during the conversion. If you do not specify `-LOGFILE`, the file is named `simvisdbutil.log`.
`-MISSING`	Specifies that if a signal is missing in the input database, it is ignored.You would use this option with the `-SIGNAL` option, which selects the signals that you want to export. When you use `-MISSING`, an error is reported only if none of the selected signals are present in the database.

`-NOCOPYRIGHT`	Disables printing of the copyright message when `simvisdbutil` starts up.
`-NOLOG`	Prevents messages from going to any log file.
`-OUTPUT` *`filename`*	Specifies the name of the generated database file.
`-OVERWRITE`	Overwrites an existing database file that has the same name as the generated database file. If you do not specify this command and a database file of the same name exists, `simvisdbutil` reports an error and exits without generating the database. It is also an error to try to overwrite the input database.
`-PERIOD` *`interval`*	Samples data at specified time intervals for CSV output.
`-QUIET`	Disables all output to `stdout` and `stderr`. Messages are written to the log file only.
`-RADIX` *`radix`*	Specifies the radix of CSV output values. Allowable *`radix`* values are `dec` or `decimal`, `hex` or `hexadecimal`, `bin` or `binary`, and `oct` or `octal`.
`-RANGE` *`start`*`:`*`end`*	Specifies the starting and ending times to be included in the output database. You can leave out the starting time range to indicate a starting time of 0, or leave out the ending time range to indicate the last recorded time. The keywords `start` and `end` can also indicate the starting and ending times.
`-SEQUENCE`	Includes sequence time information in the generated database.
`-SHM`	Generates an SST2 database and places the `.trn` and `.dsn` output files in a subdirectory with the same name as the database, plus the `.shm` extension. You can also specify this format by including the `.shm` directory path in the argument to the `-OUTPUT` option.
`-SIGNAL` *`signal`*	Exports the specified signal to the output database. To export multiple signals, use one `-SIGNAL` option per signal. If a signal is not present in the input database and you do not use the `-MISSING` option, an error is reported.

`-SST2`	Generates a database in SST2 format and places the `.trn` and `.dsn` output files in the current working directory. This is the default. You can also specify this format by including the `.trn` file extension in the argument to the `-OUTPUT` option.
`-TIMEUNITS` *`units`*	Specifies the time units used for CSV time output. Allowable time units are `s`, `ms`, `us`, `ns`, `ps`, and `fs`.
`-VCD`	Generates a database in VCD format. When you use this option, the input file must be in SST2 format. You can also specify this format by including the `.vcd` file extension in the argument to the `-OUTPUT` option.
`-VERSION`	Displays the version of `simvisdbutil`.
`-XSUB` *`string`*	Specifies a string to use when displaying X values. If you have set the radix to `hex` or `binary`, individual X values are not substituted, so data is not lost.

Generating a Database That Contains a Range of Times

To generate a database that contains simulation data in a range of times, invoke `simvisdbutil` with the `-RANGE` option. For example:

```
simvisdbutil waves.shm -range 100:10000 -output waves.shm/newdb.trn
```

This command translates the SST2 database stored in the `waves.shm` directory into the SST2 files, `newdb.trn` and `newdb.dsn`, and it places the generated database in the `waves.shm` directory. The database contains the simulation data that falls between simulation time 100 and 10,000. Because no time units were specified on the command line, they are the same as the time units in the input database.

Generating a Database for a Specific Set of Signals

To generate a database that contains simulation data for only a specific set of signals, use the `-SIGNAL` option. For example:

```
simvisdbutil waves.shm -output waves.shm/signals.trn -signal
test_drink.top.dispense -signal test_drink.top.vending.current_state
```

Reloading a Database

You may want to reload a database if you have overwritten it from outside the SimVision analysis environment or from a different SimVision session, and you want to view the new data in your current session. (Changes made from within the current session are reflected automatically and do not require you to reload the database.)

To reload a database:

1. Choose *File – Reload Database* from any SimVision window. This opens the Reload Database form.
2. Select the database that you want to reload, and click *OK*.

Renaming a Database

You might want to rename a database as a way to save it before rerunning a simulation. You can then compare the old simulation data against the new data.

To rename a database from the Waveform window, Design Browser window, or Databases tab of the Properties window:

1. Choose *File – Rename Database* from the menu bar. SimVision opens the Rename Database form.
2. Edit the filename in the *NEW SST filename* field.
3. Click *OK*.

Displaying Information about Databases

Databases that you create during simulation are associated with the simulator; existing databases that you load into SimVision are not. Therefore, information about databases can be found in two places.

Displaying Information about Databases Created during Simulation

To display information about the databases that you create during simulation:

1. Choose *Windows – Tools – Simulator* from the menu bar. This opens the simulator tab of the Properties window.

2. Select *Databases*.

   The Properties window shows whether the database is currently enabled for writing, the database's rank order in the list of databases, and its logical name, type, and filename.

3. Enable the check box to allow the simulator to write to the database; disable the check box to stop writing to the database.

Tip

To write information to a database only for a specific period of time:

- ❑ Create a database, then disable the check box in the Properties window.
- ❑ Set two breakpoints—one for the time you want to begin writing to the database, the other for the time you want to stop writing.
- ❑ Start running the simulation. When the first breakpoint occurs, enable the check box.
- ❑ Resume the simulation. SimVision records information in the database.
- ❑ When the second breakpoint occurs, disable the check box. SimVision stops writing to the database.

Displaying Information about Databases Loaded into SimVision

To display information about the databases that you have loaded into SimVision:

➤ Choose *Windows – Tools – Databases* from the menu bar. The Databases tab in the Properties window displays information about all open databases.

The Databases tab displays the logical names of all opened databases and information about the database that is currently selected. This information includes:

- ❑ The logical name of the database. You can change the logical name by editing the name in the text field.
- ❑ The path to the database file and the range of simulation time stored in the file.
- ❑ Information about the database, such as its format and creation date.

Important

If you create a database during simulation, then switch to PPE mode, the database appears in both the Databases tab and in the simulator Databases tab.

Closing a Database

To close a database from any SimVision window:

1. Choose *File – Close Database/Simulation* from the menu bar. SimVision opens the Close Database/Simulator form.
2. Select the database that you want to close, and click *OK*.

In the Design Browser window, you can also close a database as follows:

➤ Select the database in the scope view, then right-click and choose *Close*.

To close a database from the Databases tab of the Properties window:

➤ Select the database name and click *Close Database,*  .

8

Creating and Managing Probes

When you probe signals, the simulator saves every value change for those signals during recorded simulation time. You can later reload the simulation data into SimVision and debug the design without running the simulator.

If your design contains assertions, you can probe them as described in *Assertion Checking in Simulation*.

Setting a Probe

When you add signals and scopes to the Waveform window, SimVision automatically creates a probe before it executes the next simulator command. At that time, SimVision opens a database (if you have not opened one already) and sets a probe on each of the selected objects. The probed information is written to the default database. For scopes, only the ports for the selected scope are probed.

Creating a probe with the Set Probe form gives you more flexibility than letting the Waveform window create one for you. The Set Probe form lets you probe one or more levels of subscope, choose the types of signals that you want to probe within those scopes, and write the probed information to any database.

The Set Probe form differs, depending on the simulator you are using:

Incisive simulator	See “Setting a Probe with the Incisive Simulator” on page 125.
Verilog-XL	See “Setting a Probe with Verilog-XL” on page 126.

Setting a Probe with the Incisive Simulator

To create a probe with the Set Probe form:

1. Select the signals or scopes that you want to probe, then choose *Simulation – Create Probe* from the menu bar, or right-click and select *Create Probe*.

The area of the window labeled *Probe these signals and scopes* contains the names of the objects you have selected. You can add objects to the probe by selecting them in another window and clicking *Add*, , in the Set Probe toolbar, or by entering their names in the *Signal/Scope* field and clicking the *Add* button beside the field.

2. Enter a name for the probe in the *Probe Name* field. If you do not name the probe, SimVision assigns a name to it.
3. Enable the *Include tasks* option if you want to include task scopes in the probe; enable the *Include functions* option if you want to include function scopes. (You do not need to enable these options if the selected scope is a task or function.)
4. If you have probed a scope, you can enable *Include sub-scopes* and choose which subscopes you want to probe. When probing scopes, you can also choose the types of signals that you want to probe within those scopes.

   For information on probing SystemC processes, see the *NC-SC Simulator User Guide*.
5. Choose a database for storing the simulation data. You can choose a database from the list of opened databases, or click *New Database* to create a new database.
6. Disable *Add to waveform display* if you do not want to display the probed signals in the Waveform window when the probe is created. If you disable this button, you can add the signals to the Waveform window at a later time.

   If you include subscopes and enable *Add to waveform display*, only the signals in the top-level scope are added to the window, because recursively adding scopes to the Waveform window could adversely affect performance.
7. Click *OK* to create the probe.

To remove an object from the *Probe these signals and scopes* list:

➤ Select the object in the list and click *Delete*, .

Setting a Probe with Verilog-XL

To create a probe with the Set Probe form:

1. Select the signals or scopes that you want to probe, then choose *Simulation – Create Probe* from the menu bar, or right-click and select *Create Probe*.

   The area of the window labeled *Probe these signals and scopes* contains the names of the objects you have selected. You can add objects to the probe by selecting them in another window and clicking *Add*, , in the Set Probe toolbar, or by entering their names in the *Signal/Scope* field and clicking the *Add* button beside the field.

2. If you have probed a scope, you can choose whether to include subscopes and you can choose the types of objects in those subscopes to include.
3. Click *OK* to create the probe.

To remove an object from the *Probe these signals and scopes* list:

➤ Select the object in the list and click *Delete*, .

Managing Probes in the Properties Window

All probes are listed in the Properties window. From the Properties window, you can enable, disable, and delete probes, or create new probes.

To access probes in the Properties window:

1. Choose *Simulation – Show – Probes* from the menu bar of any SimVision window. This opens the simulator tab of the Properties window.

   The NC Simultor Probes tab contains a check box to indicate whether the probe is enabled or disabled, the probe name and the name of the database associated with the probe. When you have many probes, you can sort them by clicking a column heading. An arrow in the heading indicates whether the data is sorted in ascending order (up-arrow) or descending order (down-arrow).

   The Verilog-XL Probes tab contains a check box to indicate whether the probe is enabled or disabled, and the the names of the probed objects. You can sort the table by clicking a column heading. An arrow in the heading indicates whether the data is sorted in ascending order (up-arrow) or descending order (down-arrow).

To enable and disable a probe:

➤ Click the box in the *Enabled* column, . When the box is checked, the probe is enabled. When it is not checked, the probe is disabled.

Tip

You can enable and disable a probe at different times during simulation. In this way, you save simulation data only for the times that you are interested in debugging. This can make the simulation database smaller, which improves performance when loading data and viewing waveforms.

To create a probe in the Properties window:

➤ Click *Probe*,  . This opens the Set Probe form. Enter the probe information as described in "Setting a Probe" on page 125.

To delete a probe in the Properties window (Incisive simulator):

➤ Select the probe and click *Delete*,  .

9

Setting and Managing Breakpoints

You might want to run the simulation up to a point, then stop so that you can examine the state of the design. To control where simulation stops, you set breakpoints. SimVision lets you set the following types of breakpoints:

- A time breakpoint stops simulation at a specified time or time interval. Delta breakpoints are a type of time breakpoint.
- A signal breakpoint stops simulation, based on the value of a design object.
- A line breakpoint stops simulation when it reaches a specified line in the source code.
- A condition breakpoint stops simulation when a specified condition is detected. You can set condition breakpoints only with the Incisive simulator.
- A process breakpoint stops simulation when a VHDL process begins executing.
- A subprogram breakpoint stops simulation when a VHDL subprogram begins executing.

Setting a Time Breakpoint

Time breakpoints stop simulation at a specified time or time interval. How you set a time breakpoint depends on the simulator you are using:

Incisive simulator	See “Setting a Time Breakpoint with the Incisive Simulator” on page 129.
Verilog-XL	See “Setting a Time Breakpoint with Verilog-XL” on page 131.

Setting a Time Breakpoint with the Incisive Simulator

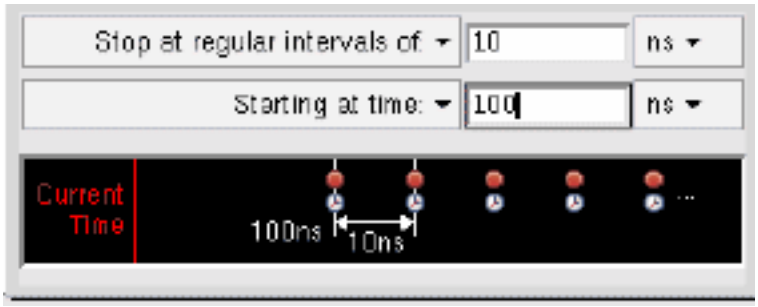
To set a time breakpoint:

➤ Choose *Simulation – Set Breakpoint – Time* from the menu. SimVision opens the Set Breakpoint form.

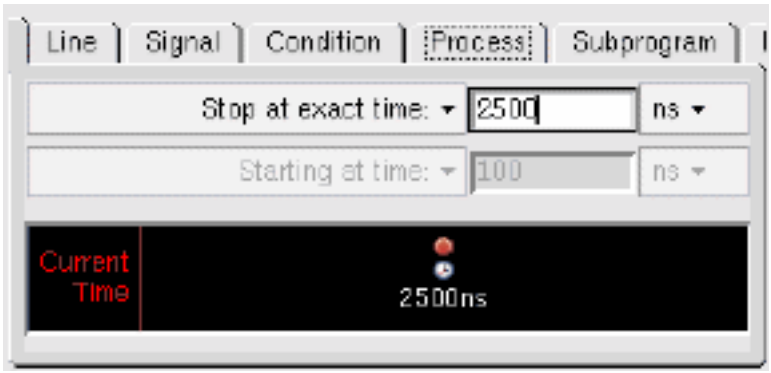
You can set the breakpoint to stop at regular intervals, starting at the current time or at a future time. You can also specify an exact time for a breakpoint that occurs only once.

As you choose settings, the breakpoint display changes. For example:

- To stop every 10 ns starting at simulation time 100 ns, enter the interval, starting time, and time units, as follows:

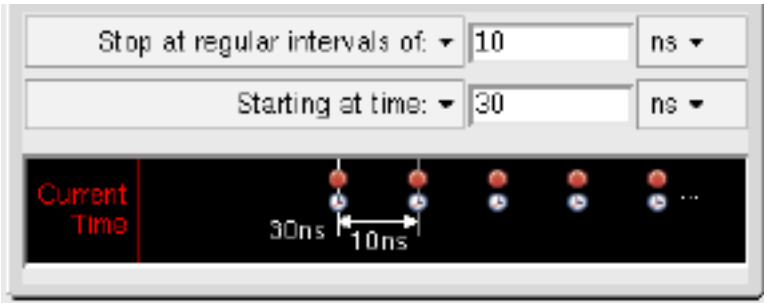


- To stop at simulation time 2500 ns, enter the time and time units, as follows:



- To stop at every 10th delta cycle, starting at the current time, enter the interval and set the time units to *delta*, as follows:

- To stop at delta cycle 10, starting at delta time 30, enter the exact time and the starting time, as follows:



You may also set breakpoint options, as described in Table 9-1 on page 137.

Setting a Time Breakpoint with Verilog-XL

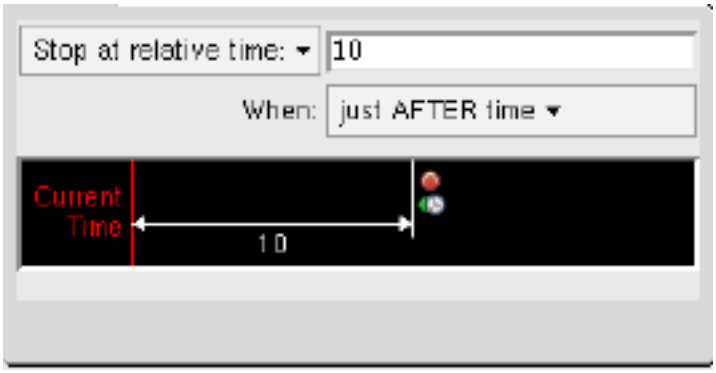
To set a time breakpoint from the *Simulation* menu:

➤ Choose *Simulation – Set Breakpoint – Time* from the menu. SimVision opens the Set Breakpoint form.

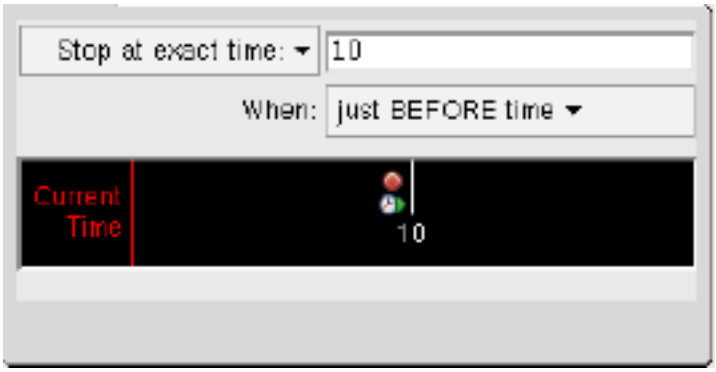
You can set the breakpoint to stop at an exact time or at a time relative to the current simulation time. You can also choose to stop immediately before or after the specified time.

As you choose settings, the breakpoint display changes. For example:

- To stop every 10 ns immediately after the specified time, enter the following:



- To stop once, immediately before 10 ns, enter the following:

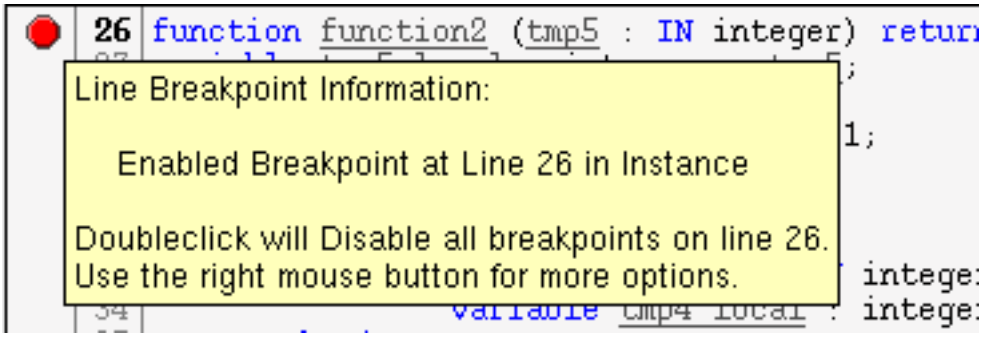


Setting a Line Breakpoint

Line breakpoints stop simulation when it reaches a particular line in the source code. The simplest way to set a line breakpoint is from the Source Browser, as follows:

➤ Double-click on the line number. A red stop sign appears in the left column, as shown in Figure 9-1 on page 132, to indicate that the breakpoint is set.

Figure 9-1 Line Breakpoint in the Source Browser



You can also set a line breakpoint from the *Simulation* menu. How you set an line breakpoint in this way differs, depending on the simulator you are using.

Incisive simulator	See “Setting a Line Breakpoint with the Incisive Simulator” on page 133.
Verilog-XL	See “Setting a Line Breakpoint with Verilog-XL” on page 133.

Setting a Line Breakpoint with the Incisive Simulator

To set a line breakpoint from the *Simulation* menu:

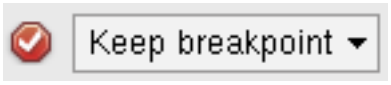
1. Choose *Simulation – Set Breakpoint – Line* from the menu. SimVision opens the Set Breakpoint form.
2. Enter the line number in the *Line* field, and enter the source filename in the *File* field.
3. Select *Stop in one instance at Scope* or *Stop in all instances of Module/Unit* if the design has more than one instance of the module or unit at the specified line. Then specify the scope at which you want to set the breakpoint.

   You can also set breakpoint options, as described in Table 9-1 on page 137.

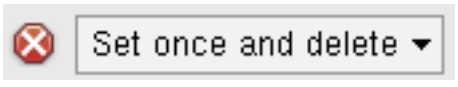
Setting a Line Breakpoint with Verilog-XL

To set a line breakpoint from the *Simulation* menu:

1. Choose *Simulation – Set Breakpoint – Line* from the menu. SimVision opens the Set Breakpoint form.
2. Enter the line number in the *Line* field, and enter the source filename in the *File* field.
3. Select *Stop in one instance at Scope* or *Stop in all instances of Module/Unit*, if the design has more than one instance of the module or unit at the specified line. Then specify the scope, or the module or unit, at which you want to set the breakpoint.
4. Select one of the following options to specify how long you want the breakpoint to exist:



Breakpoint remains in effect until you disable it or until the end of simulation.



Breakpoint triggers only once, and then it is deleted.

Setting a Signal Breakpoint

Signal breakpoints stop simulation based on the value of a signal. The simplest way to set a signal breakpoint is from the simulation toolbar; select the signal and click Breakpoint, .

You can also set a breakpoint from the *Simulation* menu. How you set a signal breakpoint in this way differs, depending on the simulator you are using:

Incisive simulator	See "Setting a Signal Breakpoint with the Incisive Simulator" on page 134.
Verilog-XL	See "Setting a Signal Breakpoint with Verilog-XL" on page 134.

Setting a Signal Breakpoint with the Incisive Simulator

To set a signal breakpoint from the *Simulation* menu:

1. Select the signals that trigger the breakpoint, then choose *Simulation – Set Breakpoint – Signal*. SimVision opens the Set Breakpoint form.

   The *Signal* field contains the names of any signals you have selected. You can add other signals to the breakpoint by selecting the signals in any SimVision window and clicking *Add*,  .

   You can also set breakpoint options, as described in Table 9-1 on page 137.

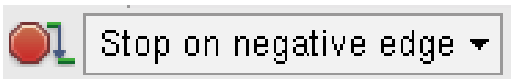
Setting a Signal Breakpoint with Verilog-XL

To set a signal breakpoint from the *Simulation* menu:

1. Select the signals that trigger the breakpoint, then choose *Simulation – Set Breakpoint – Signal* from the menu bar. SimVision opens the Set Breakpoint form.

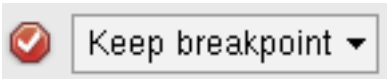
   The *Signal* field contains the names of any signals you have selected. You may add other signals to the breakpoint by selecting the signals in any SimVision window and clicking *Add*,  .

2. Select one of the following options to specify when you want the breakpoint to stop:

	Breakpoint triggers whenever the signal value changes.
	Breakpoint triggers when the signal value goes low.
	Breakpoint triggers when the signal value goes high.

Breakpoint triggers when the signal has the specified value.

3. Select one of the following options to specify how long you want the breakpoint to exist:



Breakpoint remains in effect until you disable it or until the end of simulation.

Breakpoint triggers only once, and then it is deleted.

Setting a Condition Breakpoint

Note: You can set a condition breakpoint only with the Incisive simulator.

A condition breakpoint stops simulation when a condition expression evaluates to true. SimVision evaluates the condition whenever a change occurs to any of the objects referenced by the condition.

A condition expression can check the values of wires, signals, registers, and variables, or it can check the values written to memories. This type of breakpoint is particularly useful when you want to stop simulation the moment an object has been set to an incorrect value.

A condition is a Tcl expression. See the following topics in your simulator online help for more information:

- The description of the `stop` command provides examples of condition breakpoints.
- The section “Tcl Expressions as Arguments” in the chapter “Using the Tcl Command-Line Interface” provides guidelines for writing expressions.
- Appendix A, “Basics of Tcl,” provides general information on Tcl and details on the simulator extensions to Tcl.

To set a condition breakpoint:

1. In any SimVision window, select the objects that you want to include in the condition expression.
2. Choose *Simulation – Set Breakpoint – Condition* from the menu bar. SimVision opens the Set Breakpoint form.

   The *Condition* field contains the names of any objects you have selected, preceded by the # sign and separated by ??.

3. Edit the expression to specify the condition you want to evaluate, as follows:

   - Add objects to the condition expression by selecting them in another window and clicking *Add*,  , by dragging and dropping them into the *Condition* field, or by entering them directly into the field.

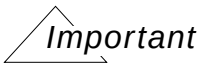
     The expression parser does not recognize names. Therefore, you must precede object names with the # sign or use the `value` command, as in `[value test_drink.top.dime_in]`, so that the value of the object is evaluated rather than the name. If you do not do this, SimVision returns a syntax error.

   - Supply the condition operators for the expression. You must substitute ?? with the appropriate condition operator for any preselected objects.

   You can also set breakpoint options, as described in Table 9-1 on page 137.

Setting a Process Breakpoint for VHDL

For VHDL, you can set a breakpoint that stops the simulation when a named process starts or resumes after a `wait` statement.



You must compile with the `-linedebug` option to enable the setting of process breakpoints.

To set a process breakpoint:

1. Choose *Simulation – Set Breakpoint – Process*. The Set Breakpoint form opens.
2. Enter the name of the process in the *Process* field.

   You can select the process in the Source Browser and then choose *Simulation – Set Breakpoint – Process*. The Set Breakpoint form is seeded with the name of the selected process.
3. Select the options you want and click *OK*. See "Breakpoint Options" on page 137 for details on these options.

Setting a Subprogram Breakpoint

You can set a subprogram breakpoint on a VHDL or Verilog function or procedure, then use the `step` command to step into the function or procedure and view the values of objects declared there.

To set a subprogram breakpoint:

1. Choose *Simulation – Set Breakpoint – Subprogram*. The Set Breakpoint form appears.
2. Enter the name of a function or procedure in the *Subprogram* field.
3. Select any additional options that you want and click *OK*. See “Breakpoint Options” on page 137 for details on these options.

Breakpoint Options

The Incisive simulator let you set breakpoint options, as described in Table 9-1 on page 137.

Table 9-1 NC Simulator Breakpoint Options

Breakpoint Name

Specify an optional name for the breakpoint. By default, breakpoints are numbered sequentially. You can use the breakpoint name in Tcl commands.

Enable this option to set a condition on the breakpoint. The breakpoint triggers only if the condition is true. The condition can be any Tcl boolean expression. See your simulator documentation for information on writing Tcl boolean expressions.

Enable this option to execute a Tcl command when the breakpoint triggers. For example, you might want to examine the value of a set of signals when the breakpoint occurs. See your simulator documentation for information on writing Tcl commands.

Select this option to always stop at this breakpoint.

Select this option to skip a specified number of occurrences of this breakpoint.

Select this option to resume the simulation immediately after executing the breakpoint. For example, you could set a breakpoint to stop every 1000 ns and execute a `value` command, then resume simulation.

Select this option to keep this breakpoint after it has triggered.

Select this option to delete the breakpoint after it has triggered a specified number of times.

Managing Breakpoints in the Properties Window

All of the breakpoints that you define are listed in the Properties window. From the Properties window, you can enable, disable, delete, and create breakpoints.

To access breakpoints in the Properties window:

1. Choose *Simulation – Show – Breakpoints* from the menu bar of any SimVision window. This opens the simulation tab of the Properties window.

To enable and disable a breakpoint:

➤ Click the box in the *Enabled* column,  . When the box is checked, the breakpoint is enabled. When it is not checked, the breakpoint is disabled.

To set a breakpoint:

➤ Click *Set Breakpoint*,  , and fill in the Set Breakpoint form.

To delete a breakpoint:

➤ Select the breakpoint and click *Delete*,  .

10

Changing and Monitoring the Value of an Object during Simulation

Sometimes you might want to change the value of an object during simulation. For example, you might suspect that a signal is not being set correctly, and you might want to test the design behavior when the signal has a different value.

SimVision lets you change an object value in the following ways:

- You can force an object to a particular value. When you set a force, the value change occurs immediately, and future transactions on the object are blocked. That is, the object keeps the forced value until you release it or until you force it to a different value.
- You can deposit a value into an object. When you deposit a value, behaviors that are sensitive to value changes on that object run when the simulation resumes, as if the value change was caused by the design source code. You can deposit a value immediately, at some time in the future, or after a delay. Although you cannot change the object back to its original value, future transactions are not blocked.
- You can create or replace a run-time driver for a VHDL signal.
- If you are running Verilog-XL, you can create a monitor to report the changes in a signal value during simulation.

Important

Any signal being forced or deposited must have write access. That is, you must first elaborate the design with the `-access +w` option, as described in "Preparing Your Design for Simulation" on page 37.

Forcing and Releasing a Signal Value

You can force any type of design object except the following:

- A Verilog memory or memory element

- A bit-select or part-select of a Verilog register or unexpanded wire
- A VHDL variable
- An analog signal
- A digital signal when the analog solver is active

Forces are saved when you save a simulation checkpoint.

To create a force:

1. Select an object in any SimVision window, then choose *Simulation – Create Force* from the menu bar or right-click and choose *Create Force*. SimVision opens the Create Force form. The name of the object and its current value are displayed in the form.

Tip

If you do not select an object before opening the form, you can select it in any SimVision window and click *Add*,  , to add it to the form.

2. Specify the new value for the object. For some objects, such as binary objects, the drop-down menu for the *Value* field lists all values that are valid for the object; you can choose a value from the list. For other objects, such as state machine variables, you must enter the new value in the field.

To display the forces that you have set:

➤ Choose *Simulation – Show – Forces*. SimVision opens the Forces tab of the Properties window.

For each force that is in effect, this tab shows the same information that the `force -show` command displays. That is, it displays the name of the forced object, the value of that object, and where the force originated.

Note: For the Properties window to display forces created by a Verilog procedural `force` continous assignment, you must compile the Verilog source code with the `-linedebug` option, or elaborate the snapshot with the `-show_forces` option.

For more information on the `force -show` command, see the `force` command in your simulator *Help*.

To set a force from the Properties window:

➤ Click *Force*,  . This opens the Create Force form.

To delete a force:

➤ Select a force in the Properties window and click *Delete*, .

To release a force:

➤ Select the object in the Design Browser tab of any SimVision window, then right-click and choose *Release Force*.

Depositing a Signal Value

Note: You cannot set a deposit with Verilog-XL.

You can deposit a value into ports, signals, and variables. If the object is a memory or a range of memory elements, the value is deposited into each element of the memory or each element in a specified range.

You cannot deposit values in the following types of objects:

- An analog signal
- A digital signal when the analog solver is active
- A VHDL variable or signal with multiple sources, if a delay is specified

To deposit a value:

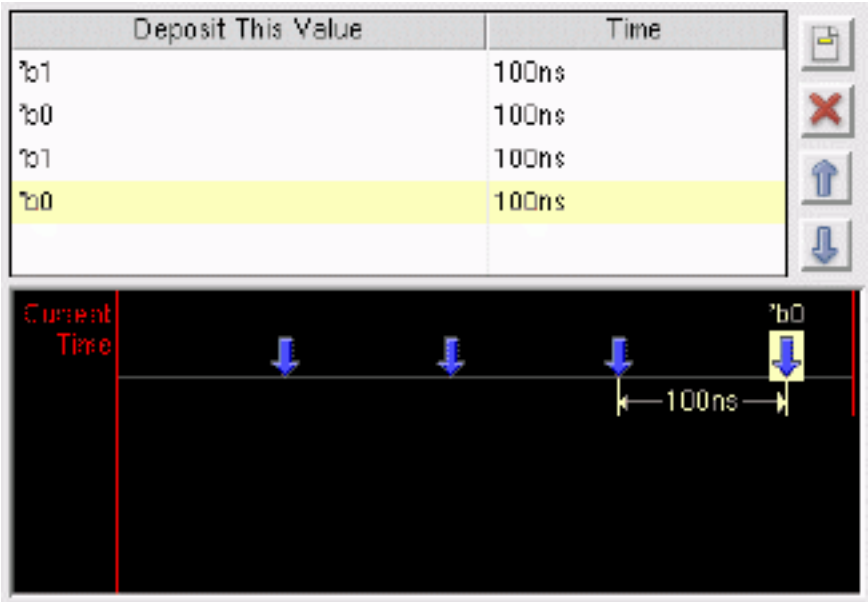
1. Select the object in any SimVision window and choose *Deposit Value* from the pop-up menu or from the menu bar. SimVision displays the Deposit Value form.

   The name of the object and its current value are displayed in the area at the top of the form. You can specify options for the deposit command in the bottom area of the form. As you enter options, the area in the middle of the form changes to show how values are deposited in relation to simulation time.

2. You can deposit other values at different times by clicking *New Value*, . You can then specify the new value and the time when you want to deposit that value.

   For example, Figure 10-1 on page 142 specifies four values to be deposited in the `test_drink.top.vending.reset` signal at 100 ns intervals. The timeline display shows the four points at which a value is to be deposited. The selected value is highlighted along the timeline, showing the interval in which the deposit occurs and the deposited value.

Figure 10-1 Depositing Additional Values



These other buttons let you manage the list of deposited values:

- Deletes the selected entry.
- Moves the entry up one position in the list.
- Moves the entry down one position in the list.

3. Specify any of the following options:

- Choose whether the deposit should occur at relative time intervals or at absolute time points.
- If you want to deposit values at regular time intervals, enable *Repeat* and specify the time interval.
- If you want to stop depositing values after a specified time, enable *Cancel* and enter the time.
- Enable *Release* if you want to release any forces before depositing the value.
- Choose the type of delay that you want to occur before depositing the value—transport delay, inertial delay, or no delay.

The display changes to represent how the value or values are deposited during simulation.

4. Click *OK* to create the deposit command.

Creating, Replacing, and Deleting a VHDL Run-Time Driver

Note: You can set a run-time driver only on a VHDL signal.

To create or replace a run-time driver:

1. Choose *Simulation – Create Driver* from any SimVision window, or select a signal, then right-click and choose *Create Driver*. This menu choice appears in the menus only when your design contains VHDL.

   SimVision opens the Set Driver form. If you select a VHDL signal before opening this form, the signal name and value are entered in the form.

2. In the *Object* field, enter the name of the signal driver that you want to set, or select the signal driver in any SimVision window and click *Add*,  .

3. In the *Value* field, enter the value that you want to give to the driver.

4. Enable *Create* to create a driver, or enable *Replace* to replace the driver.

5. Click OK.

To delete a run-time driver:

➤ Select the VHDL signal that has the run-time driver, then right-click and choose *Delete Driver*.

For more information, see the `driver` command in the *NC-VHDL Simulator Help*.

Monitoring Signal Value Changes in Verilog-XL

If you are running Verilog-XL, you can monitor selected signal values during simulation. When you create a monitor, the simulator displays a message whenever the signal value changes. The monitor messages are displayed on a separate line but interleaved with any other messages that the testbench displays during simulation.

Unlike a breakpoint, simulation does not stop when the value changes. A message is written to the *simulator* tab of the Console window, and the simulation continues to run.

Setting a monitor in SimVision invokes the `$monitor` system task. For more information about this system task, see the *Verilog-XL User Guide*.

To create a monitor:

1. Select the signals that you want to monitor, then choose *Simulation – Create Monitor*. SimVision opens the Create Monitor form.
2. Add a signal by entering its full hierarchical name in the *Signal/Scope* field and clicking *Add*, or by selecting the signal in another window and clicking *Add*, .
3. Specify the radix for the signal value by choosing a radix from the *Format* drop-down menu, or specify a message string by choosing *Use this format string* from the *Format* menu, then enter the string in the text field. The string can contain any of the following substitution strings:

%b	Binary number
%o	Octal number
%d	Decimal number
%x	Hexadecimal number

   For example, if you enter the string `dispense = %b`, the *simulator* tab of the Console window displays the message `dispense = 1` or `dispense = 0`, whenever the value of the signal changes.

11

Controlling the Simulation

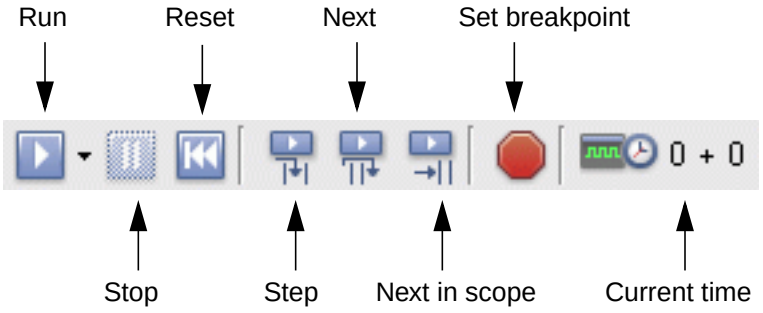
The simulation toolbar lets you run the simulation, stop at breakpoints, and step into and over subprogram calls. The *Simulation* menu gives you menu choices to not only run the simulation, but also to control the simulation in the following ways:

- Reset the simulation to time 0.
- Reinvoke the simulation after you have changed the source code. Reinvoking the simulation automatically recompiles the design and loads it into the same simulator session.
- Saving the state of a simulation, or checkpoint, and restarting the simulation from that point.
- Creating and deleting aliases for simulator or other Tcl commands.
- Setting variables to control the simulation environment.

Running the Simulation

The most common simulation functions are available in the *Simulation* toolbar, as shown in Figure 11-1 on page 145.

Figure 11-1 Simulation Toolbar



The *Run* button runs the simulation until the next breakpoint or the end of simulation. When you click the small down-arrow next to the *Run* button, a text field opens, as shown in Figure 11-2 on page 146. You can enter a duration in that field. When this field is visible, the *Run* button runs the simulation for the specified amount of time.

Figure 11-2 Running the Simulation for a Specified Amount of Time

You can also start, stop, and resume simulation from the *Simulation* menu. Table 11-1 on page 146 shows the *Simulation* menu choices.

Table 11-1 Simulation Menu Choices

Menu Choice	Description
Simulation – Run	Run the simulation until it finishes or until it is interrupted by a breakpoint or a language construct, such as a Verilog `$stop` system task.
Simulation – Stop	Interrupt the simulation.
Simulation – Step	Run one behavioral statement, stepping into any subprogram calls.
Simulation – Next	Run one behavioral statement, stepping over any subprogram calls.
Simulation – Next in Scope	Run to the next statement, maintaining focus in the active scope.
Simulation – Return	Run until the current task, function, procedure, or VHDL subprogram returns. **Note:** Not available in Verilog-XL.
Simulation – Advance	Run the simulation at the delta cycle level. See Chapter 12, “Debugging at the Delta Cycle Level” for more information.

Resetting the Simulation

Note: Resetting the simulation is not available for mixed-signal designs.

When you reset the simulation, you return to time zero. After you reset the simulation, the debugging environment is preserved, as follows:

- SST2 and VCD databases remain open, except for VCD databases created with the `$dumpvars` system task. Those databases are closed.
- All breakpoints and probes remain set.
- The values of Tcl variables remain as they were before the reset.
- All windows remain open.
- Forces and deposits are removed.

Reinvoking the Simulation

When you change your source code, you must reinvoke the simulator to access those changes. When you reinvoke Verilog-XL, SimVision recompiles all of the source files for the design. When you reinvoke the Incisive simulator, SimVision recompiles only the changed design units. It then re-elaborates the design, generates a new snapshot, and loads the updated snapshot.

When you reinvoke, your current setup is automatically saved and restored. This includes the following:

- SST2 databases
- Breakpoints
- Probes
- Windows

To reinvoke the simulator from the *Simulation* menu:

➤ Choose *Simulation – Reinvoke Simulator* from any SimVision menu bar.

By default, the simulator uses the same command line to compile and elaborate your design. However, you can set a preference to prompt you for a command line before reinvoking the simulator.

To set this preference:

- Choose *Edit – Preferences* from any SimVision window.
- In the General Options tab, enable the *Prompt on reinvoke* option.

To reinvoke the simulator from the Properties window:

1. Choose *Windows – Tools – simulator* from the menu bar. SimVision opens the *simulator* tab of the Properties window.

   The top of the form displays information about the current simulator session, plus buttons to disconnect or terminate the simulator, or enter PPE mode. The *Reinvoke* area of the form contains a *Command line* field, which shows the command-line options that you used to invoke the simulator. If you want to change the command-line options, enter the new options in the *Command line* field.

2. Click *Reinvoke*.

Saving and Restarting a Simulation Checkpoint

A simulation checkpoint is the state of a design at a particular simulation time. A checkpoint includes the databases you have opened, breakpoints you have set, and any forces and deposits that are in effect at that time. Verilog-XL saves the simulation state in a file; the Incisive simulator creates a new snapshot.

Creating simulation checkpoints is especially useful for large simulations, where you might want to save the simulation state at regular intervals. For example, you may want to save the simulation state after a circuit has been initialized, so that future simulations can begin at that point rather than from time 0.

You restart the simulation at a checkpoint by loading the file or snapshot into the simulator. You can restart the simulation during the same simulation session, or exit the simulator and start a new session using the saved snapshot.

Note: For mixed-signal designs, you cannot restart another snapshot after simulation has started for the current snapshot.

Saving a Simulation Checkpoint

To save a simulation checkpoint:

1. Select *Simulation – Save Checkpoint*. SimVision opens the Save Checkpoint form.

2. Click *Run Clean* if the simulator is not in a clean state—that is, if it is executing sequential HDL code. The *Run Clean* button runs the simulation until the currently running sequential behavior suspends at a delay, event control, or VHDL `wait` statement. Otherwise, the simulator cannot save the checkpoint.
3. Enter a name for the checkpoint in the *Save this snapshot as* field. If you have created other checkpoints, they are listed in the window. You can replace one of those snapshots by selecting its name in the list.
4. Click *Save* to create the checkpoint.

Restarting during the Same Simulation Session

To restart during the same simulation session:

1. Choose *Simulation – Restart from Checkpoint*. SimVision opens the Restart from Checkpoint form.
2. Select a checkpoint from the list and click *Restart*.

When you restart with a saved snapshot in the same simulation session, your debugging environment is saved, as follows:

- SST2 databases remain open and all probes remain set.
- Breakpoints set at the time that you execute the restart remain set.

  Note: After a restart, periodic breakpoints trigger at the same times they would trigger without a save and restart. This is true even when the save and restart takes place at a time in between the periodic breakpoints. For example, suppose you set a breakpoint that triggers every 10 ns, at times 10, 20, 30, and so on. If you restart with a snapshot saved at time 15, the breakpoint triggers at 20, 30, and so on, not at 25, 35, and so on.
- Forces and deposits in effect at the time you save the snapshot remain in effect when you restart.

Restarting from a New Simulator Session

To restart from a new simulator session:

➤ Invoke the simulator with the name of the checkpoint snapshot or file, as follows:

```
ncsim -gui checkpoint
```

When you invoke the simulator with a saved snapshot, databases are closed. Any probes and breakpoints are deleted. If you want to be able restore the full Tcl debug environment when

you restart, you must save the debugging environment before exiting from the simulator, as described in Chapter 3, “Saving and Restoring Your Debugging Environment.”

Creating and Deleting an Alias in the Incisive Simulator

The Incisive simulator lets you create an alias for any simulator command or series of commands. An alias gives a name to a command, which you can use in place of the simulator command. For example, the Incisive simulator defines `.` as an alias for the `run` command and `quit` as an alias for the `exit` command.

To create an alias:

1. Select *Simulation – Create Command Alias* from the menu bar of any SimVision window. The Command Alias form appears.
2. In the *Alias Name* field, enter the name that you want to give to the command or commands, or choose an existing alias name from the drop-down list.

   Note: You cannot create an alias with the same name as a predefined Tcl command. For example, you cannot use the `gets` Tcl command as an alias. SimVision displays the following error message in the *simulator* tab of the Console window if you try to create an alias named `gets`.

   ```
   ncsim: *E,ALNORP: cannot create an alias for Tcl command gets.
   ```
3. In the *Definition* field, enter the command or commands that you want to alias, then click *OK*.

To display the aliases that are defined:

➤ Choose *Simulation – Show – Aliases* from the menu bar of any SimVision window. SimVision opens the Aliases tab of the Properties window.

To create an alias from the Properties window:

➤ Click *Create Alias*,  , and enter the name and description in the Command Alias form.

To delete an alias:

➤ Select the alias and click *Delete*,  .

Setting Variables in the Incisive Simulator

The Incisive simulator defines a number of variables, which store information about the current simulation. For example, the `snapshot` variable contains the name of the current

snapshot, and the `time_scale` variable contains the time units currently in use. For a list of predefined variables, see the topic called “Setting Variables” in the *NC-Verilog Simulator Help* or the *NC-VHDL Simulator Help*.

The Incisive simulator also lets you define Tcl variables or change the values of a predefined variable.

To create or change a variable definition:

1. Select *Simulation – Create Debug Variable* from the menu bar of any SimVision window. SimVision opens the Debug Variable form.
2. Enter the name of the variable in the *Variable Name* field, or select a variable from the drop-down list for this field.
3. Enter the value in the *Value* field.
4. Click *OK*.

To display a list of all currently-defined variables:

➤ Choose *Simulation – Show – Variables* from the menu bar of any SimVision window. SimVision opens the Variables tab of the Properties window.

The list of variables includes some variables that Tcl sets for itself, such as the path to the Tcl libraries. These variables might change, and new variables might appear with new versions of Tcl. Refer to a Tcl manual for the meaning of these variables.

To change the value of a variable:

➤ In the Variables tab of the Properties window, double-click the variable value and enter the new value. Press `Return` to apply the new value.

To unset a variable:

➤ In the Variables tab of the Properties window, double-click the variable value and press the `Backspace` key. Press `Return` to apply the NULL value.

To create a variable in the Properties window:

➤ Click *Create*, , and enter the variable name and value in the Debug Variable form.

To delete a variable:

➤ Select the variable and click *Delete*, .

12

Debugging at the Delta Cycle Level

Note: The Simulation Cycle Debugger is available only with the Incisive simulator.

When a circuit operates, many events occur simultaneously. A simulator cannot run this way. It must simulate the parallel execution of the circuit by dividing each time point into delta cycles and executing the events, one at a time, within the time point.

Ordinarily, you see only the final state of the signals and variables at the end of the time point. However, you can use the Simulation Cycle Debugger to see what events are scheduled during each delta cycle. Simulating the design at the delta cycle level can help you find the cause of 0-delay errors or errors caused by event ordering.

Simulating at the delta cycle level can be especially helpful when you are debugging VHDL designs. When debugging Verilog designs, you might prefer to use the Waveform window to view events in sequence time. Sequence time shows you the order of events that occur during one time slice, without regard for the delta cycles in which the events occur. For more information on sequence time, see “Viewing Events in Sequence Time” on page 255.

How a Delta Cycle Executes

A delta cycle executes in phases, as follows:

- Phase 1: Signal evaluation or wire resolution

  In VHDL, phase1 is called signal evaluation; in Verilog, it is called wire resolution. Simulation returns to phase 1 after either phase 2 or phase 3 completes. The time it takes to execute a phase and return to phase 1 is called a delta cycle.

  During phase 1, the simulator schedules the processes to be executed during the delta cycle, and it assigns values to signals and wires.

- Phase 2: Process execution or behavior execution

  In VHDL, phase 2 is called process execution; in Verilog, it is called behavior execution. During this phase, the simulator executes all scheduled processes or behavioral statements.

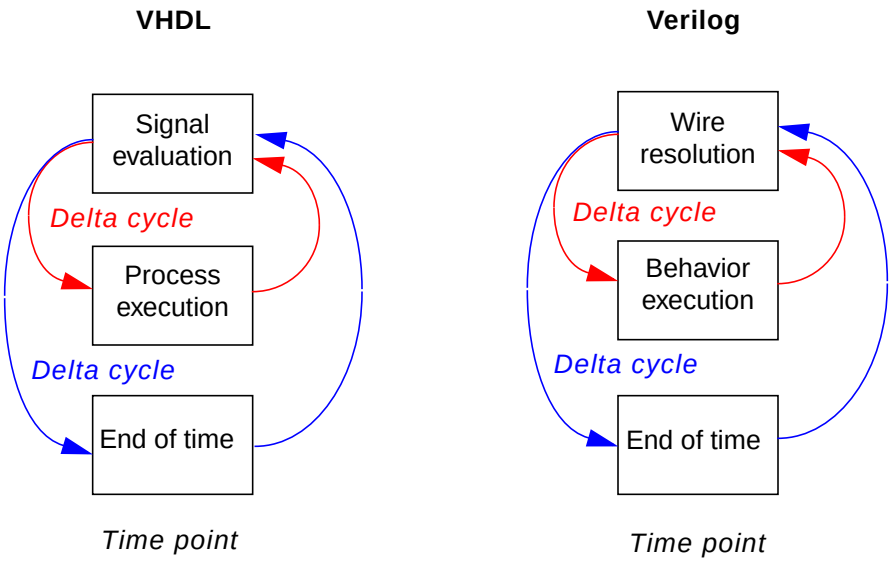
Every process in the same delta cycle is presumed to run simultaneously. Therefore, if one process changes a signal value during phase 2, another process that executes during the same delta cycle does not see the updated value; it uses the value assigned during phase 1.

- Phase 3: End of time

  In VHDL, postponed processes are executed at the end of time; in Verilog, certain tasks, such as `$monitor` tasks, occur at the end of time. After this phase completes, simulation time advances.

Figure 12-1 on page 154 shows the phases of a delta cycle.

Figure 12-1 Simulation Cycles

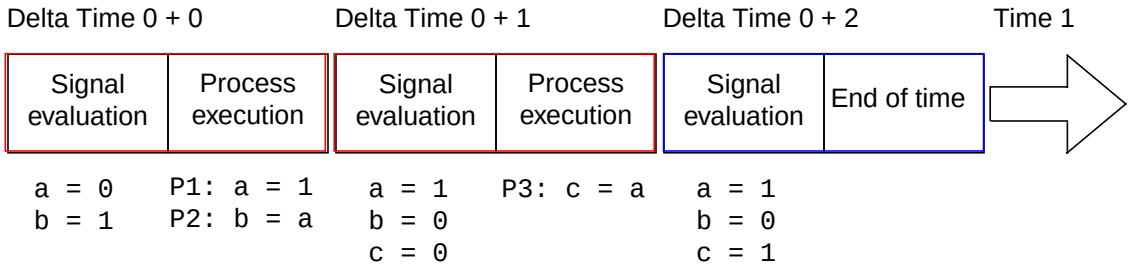


Any number of process execution or behavior execution delta cycles can occur during one simulation time point. For example, if a process is sensitive to a change in a value that gets updated during one delta cycle, the simulator schedules that process to execute in the next delta cycle.

Figure 12-2 on page 155 shows an example of three delta cycles occuring at time 0. During the first delta cycle, the simulator assigns initial values to the signals a and b, and it executes processes P1 and P2. Process P1 assigns the value 1 to a; Process P2 assigns the value of a to b. Notice that the value of b at the end of this delta cycle is 0. That is because the simulator uses the value of a at the beginning of the delta cycle, not at the end of the process P1.

Process P3 is sensitive to the value of a, which changed during the first delta cycle. Therefore, the simulator schedules P3 to execute in the next delta cycle. Process P3 assigns the value of a to signal c. At the end of Time 0, a has the value 1, b has the value 0, and c has the value 1.

Figure 12-2 Three Delta Cycles at Time 0



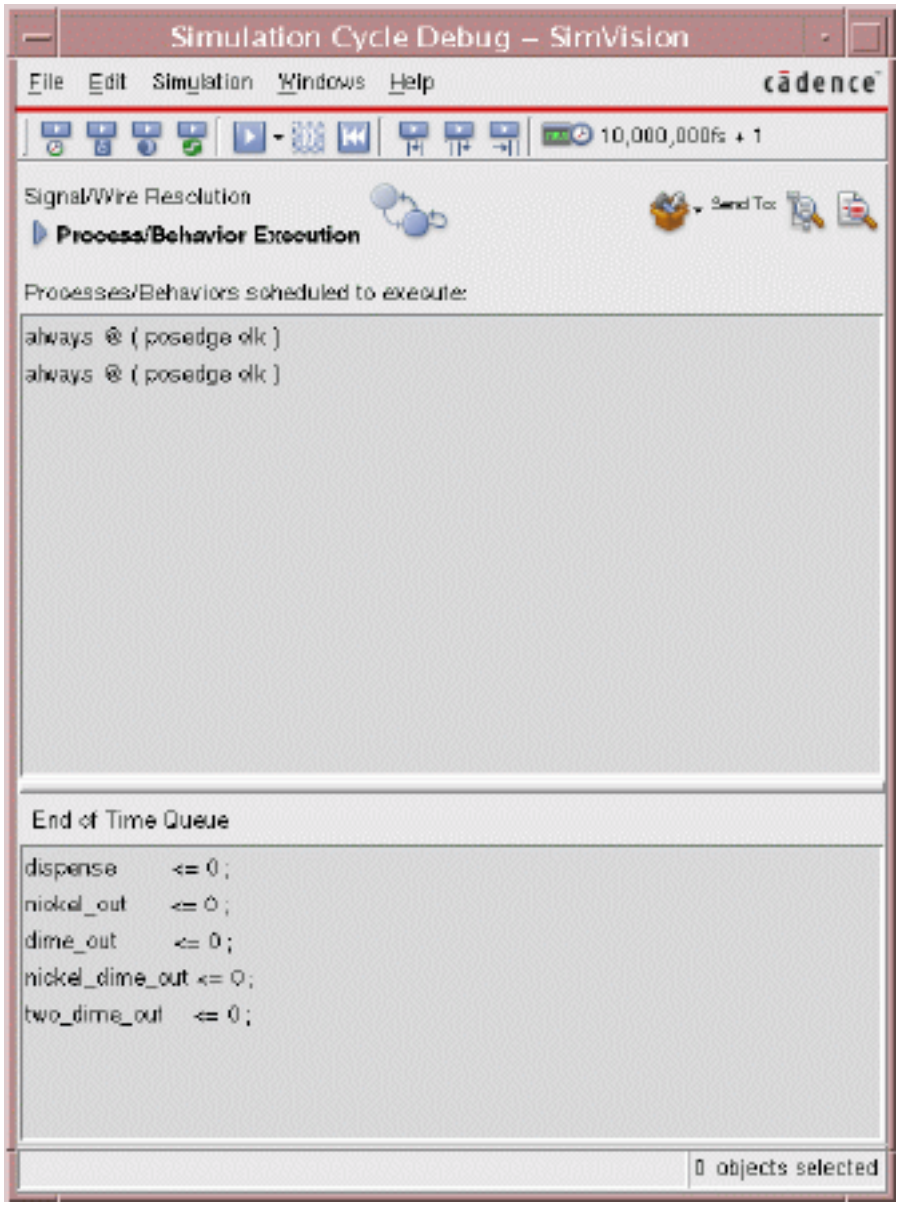
The Simulation Cycle Debugger lets you simulate from one delta time to the next, from one phase to the next, from one process to the next, or from one time point to the next. At every stopping point, it displays the events scheduled to run during the current delta cycle, and the processes or tasks scheduled to run at the end of time.

Opening the Simulation Cycle Debugger

To open the Simulation Cycle Debugger:

➤ Choose *Windows – Tools – Simulation Cycle Debug* from the menu bar of any SimVision window. SimVision opens the window shown in Figure 12-3 on page 156.

Figure 12-3 Simulation Cycle Debugger



The time toolbar contains buttons to let you control simulation at the delta cycle level. Below the time toolbar, a state diagram indicates the current phase that is executing—signal resolution or process execution for VHDL designs; wire resolution or behavior execution for Verilog designs.

Below the state diagram, the Simulation Cycle Debugger shows the list of processes or behavior statements to be executed during the current delta cycle. At the bottom of the window, it lists the processes or tasks to be executed at the end of the current time point.

Running the Simulation at the Delta Cycle Level

To run the simulation, use the menu choices or buttons shown in Table 12-1 on page 157.

Table 12-1 Cycle Debugging Menu Choices and Toolbar Buttons

Toolbar Button	Menu Choice	Description
	*Simulation – Advance – Timepoint*	Run until simulation reaches the next time point.
	*Simulation – Advance – Delta Cycle*	Run until simulation reaches the beginning of the next delta cycle.
	*Simulation – Advance – Simulation Phase*	Run until simulation reaches the next phase of the simulation cycle, either the process execution phase or the signal evaluation phase.
	*Simulation – Advance – Process*	Run until simulation reaches the next scheduled process.

Viewing the Source Code for an Event

When you have stopped simulation, such as at the beginning of a delta cycle, you can select any of the events in the Simulation Cycle Debugger and view the source code associated with that event.

To display the source code for an event:

➤ Select an event in the list of processes, behavior statements, or end of time tasks. SimVision opens a Source Browser window and positions the pointer at the line in the source code where that event is defined.

Printing and Saving the Simulation Cycle Debugger Window

You can print the contents of the Simulation Cycle Debugger window or save the contents to a postscript file.

To print or save the Simulation Cycle Debugger window:

1. Choose *File – Print Window* from the menu bar. The Simulation Cycle Debugger opens the Print form.
2. In the Printer area of the *Options* tab, choose *Command* if you want to print the window, and enter the appropriate print command. If you want to save the window to a postscript file, choose *File* and enter the name of the file.
3. In the *Comments* area, specify a heading that you want to appear on every page. This heading includes a *Title*, *Designer Name*, *Company Name*, and an additional *Note* that you want in the heading.
4. In the *Print paper* area, specify the *Paper size*, *Orientation*, and *Colors* that you want to use. Each of these fields has a drop-down list of values from which you can choose.
5. Click *OK* when you are ready to print.

To view the printed image before saving it or sending it to the printer:

➤ Open the *Preview* tab. This tab shows you the pages as they will appear. You can scale the preview by increasing or decreasing the *Scale* factor at the bottom of the tab.

13

Debugging API Applications

The Incisive simulator support several application programming interfaces (APIs), including VPI, VHPI, DPI, PLI, VDA, CFC, and FMI. These programming interfaces make it possible for you to integrate C, C++, and SystemC applications into the simulator.

API applications can serve a variety of purposes. For example:

- Implementing special-purpose simulator functions to help you debug your design
- Implementing system tasks
- Implementing C-language design models
- Integrating third-party tools into the simulation environment

SimVision provides the capability for debugging API applications directly in the SimVision environment with GDB. This native SimVision debugging mode uses the Source Browser as the debugger display. You can also use DDD as your debugging environment. When you use DDD, the SimVision windows are not available for debugging the API code.

Note: The API debugger provides additional features for debugging SystemC applications. See the *NC-SC User Guide* for information on debugging SystemC applications.


Video

Debugging API Applications

Preparing to Debug an API Application

Cadence recommends that you use `irun` to compile and link your API application for debugging in SimVision. For information on using `irun`, see Chapter 5, Compatibility with Existing Use Models, in the *irun User Guide*.

You may need to use special options when compiling and elaborating the HDL design that calls your API application, depending on the programming interface you are using. For

example, if your API application implements a DPI task, you must use the `-dpiheader` elaborator option and specify a `.h` file that declares the DPI task. See the documentation for the programming interface for information on how to compile and elaborate a design that calls your API application.

Using the Native-Mode GDB Debugger

You invoke the simulator with the `-gui` option to run the simulation in the SimVision analysis environment. When SimVision starts up, it displays the Design Browser and console windows. Simulation is stopped at time 0, and you can begin debugging your HDL code. At this time, the API debugger is not yet active. You must start GDB if you want to debug your API code.

When you start GDB, SimVision adds a *gdb* tab to the Console window. The *gdb* tab acts as the debugger command prompt, where you can enter debugger commands directly, such as `break`, `continue`, `return`, and so on. The *gdb* tab gives you full control over the entire debugging process.

The following buttons are unique to the *gdb* tab of the console window:

Function Breakpoint displays a list of functions in your API code and lets you set breakpoints on the functions you select.

Interrupt C/C++ Debugger sends an interrupt to GDB and places control at the GDB prompt.

Disconnect and Terminate from C/C++ Debugger exits GDB and removes the *gdb* tab from the console window.

When you start GDB, SimVision also adds the following buttons to the Simulation Control toolbar in any Source Browser window that you open:

Set Watchpoint creates an object breakpoint on a SystemC, C, or C++ variable.

Interrupt C/C++ Debugger sends an interrupt to GDB and places control at the GDB prompt.

Open File lets you open the API source files that are compiled with the `-g` option in the simluation process.

Function Breakpoint displays a list of functions in your API code and lets you set breakpoints on the functions you select.

Starting GDB

You can start GDB by setting a breakpoint or by invoking it from the *Simulation* menu.

To start GDB by setting a breakpoint:

1. Click *Send to Source Browser*,  , or choose *File – New – Source Browser* from any SimVision window. SimVision opens a Source Browser window.
2. In the Source Browser window, choose *File – Open Source File* and open the API source file.
3. Set a breakpoint by either double-clicking in the column to the left of the line where you want to set the breakpoint, or by right-clicking over the column and choosing *Set Break* from the pop-up menu.

   The Source Browser displays a red stop sign in the column to show that the breakpoint is set and that it is enabled. SimVision adds the GDB buttons to the Simulation Control toolbar, and it adds the *gdb* tab to the Console window. In the *gdb* tab, an `-exec-continue` command is executed, and control passes to the simulator.
4. Click *Run*,  , and simulation advances to the breakpoint. At that time, control returns to GDB.

To start GDB from the *Simulation* menu:

1. From any SimVision window, choose *Simulation – SystemC/C++/C Debug*. SimVision adds the GDB buttons to the Simulation Control toolbar in any Source Browser windows that are open, and it adds the *gdb* tab to the Console window.
2. In the Source Browser, click *Open File* and choose the API source file from the list. The source file is displayed in the Source Browser window.
3. Set a breakpoint by either double-clicking in the column to the left of the line where you want to set the breakpoint, or by right-clicking over the column and choosing *Set Break* from the pop-up menu. The Source Browser displays a red stop sign in the column to show that the breakpoint is set and that it is enabled.
4. Click *Run*,  , and SimVision executes an `-exec-continue` command in the *gdb* console. Control passes to the simulator.
5. Click *Run*,  , again and simulation advances to the breakpoint. At that time, control returns to GDB.

Viewing Source Code

You can open a source file in any of the following ways:

- Click *Open File* to open the Open C/C++ Source File form. Select a file from the list and click *OK*.
- Choose *File – Open Source File* and choose the API source file from the Open Source File form.

The Source Browser displays the source file, including line numbers. The *Scope* and *Files* fields that you would see when displaying an HDL source file are not available, because the API code is not part of the design hierarchy. In their place, the Source Browser displays the filename, as shown in Figure 13-1 on page 162.

Figure 13-1 Displaying the API Source File Name in the Source Browser

Even though it is not part of the hierarchy, the API source file is added to the scope history.

To move through the scope history:

- Click *Previous Scope*, , or*Next Scope*, , to move backward or forward through the scope history. These buttons also have a drop-down list, so that you can jump to any scope in the list.

In addition, you can use the search function to search for a text string or go to a specified line in the source file. For information on using the search function, see “Searching a Source File” on page 66.

Setting Breakpoints in the API Code

SimVision lets you set breakpoints on any line or function in the API source code. You can also enable, disable, and delete breakpoints in the API code, as you would in the design code. The Properties window keeps tracks of all breakpoints that you set, including those in the API code.

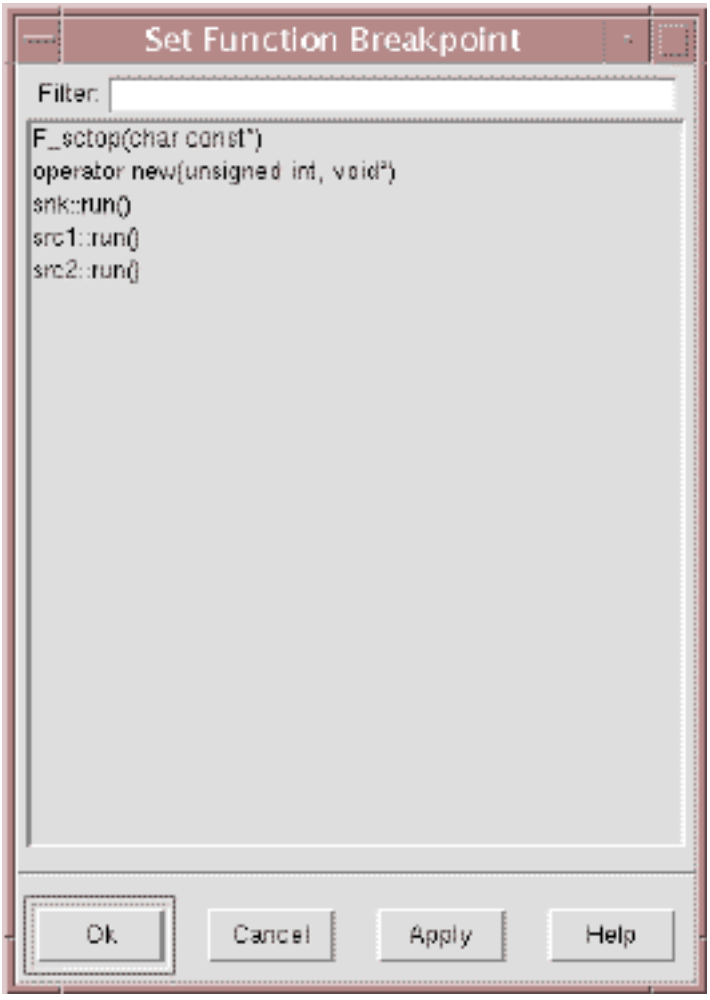
To set a line breakpoint:

1. Open the source file as described in “Viewing Source Code” on page 162.
2. Double-click on the line number where you want to set the breakpoint. SimVision places a red stop sign next to the line number, to indicate that the breakpoint is active.

To set a function breakpoint:

1. Click *Set Function Breakpoint*, , in either the Source Browser or the Console window. SimVision displays a list of the functions in the API code, as shown in Figure 13-2 on page 163.

Figure 13-2 Set Function Breakpoint Form



2. Select the function on which you want to set the breakpoint. If the list of functions is long, you can use the *Filter* field to remove functions from the list, as follows:
   - As you type, SimVision automatically removes the names that do not match the current string.
   - Enter a partial name and press Tab. SimVision completes the name up to the point at which it is no longer unique, and removes the names that do not match that string.

You can continue to enter characters and press `Tab` until you have found the function you are looking for.

- Include wildcard characters, * and ?, anywhere in the string to return all functions whose names match that pattern.

3. When you have selected the function on which you want to set a breakpoint, click *OK*. The Source Browser places a red stop sign next to the first line of the selected function to indicate that the breakpoint is active.

For more information on breakpoints and the Breakpoints Properties window, see Chapter 9, "Setting and Managing Breakpoints."

Setting Watchpoints

SimVision lets you set watchpoints on any C/C++ variable in your design. A watchpoint is like an object breakpoint—simulation stops whenever the value of the variable changes.

To set a watchpoint:

1. Select the variable in the Source Browser.
2. Click *Set Watchpoint*,  , in the API debugger toolbar.

SimVision records all watchpoints in the Breakpoints tab of the Properties form. You can use this form to delete or disable a watchpoint, as you would a breakpoint.

Passing Control between the Simulator and GDB

When simulation reaches a breakpoint in the API code, control is given to the GDB debugger. When simulation stops in the design code, control is given to the simulator.

The *Simulation* menu lets you issue commands, such as `step`, `next`, `return`, and so on, to advance the simulation. You can use these functions with your API code, as well as the HDL code. See Chapter 11, "Controlling the Simulation" for information on the *Simulation* menu and toolbar.

When the simulation is under control of GDB and the simulator is suspended, the `force` and `deposit` menu commands are disabled. If you want to switch context from the simulator back to GDB to continue debugging, set additional breakpoints, and so on, you can interrupt the debugging process.

To interrupt the debugging process:

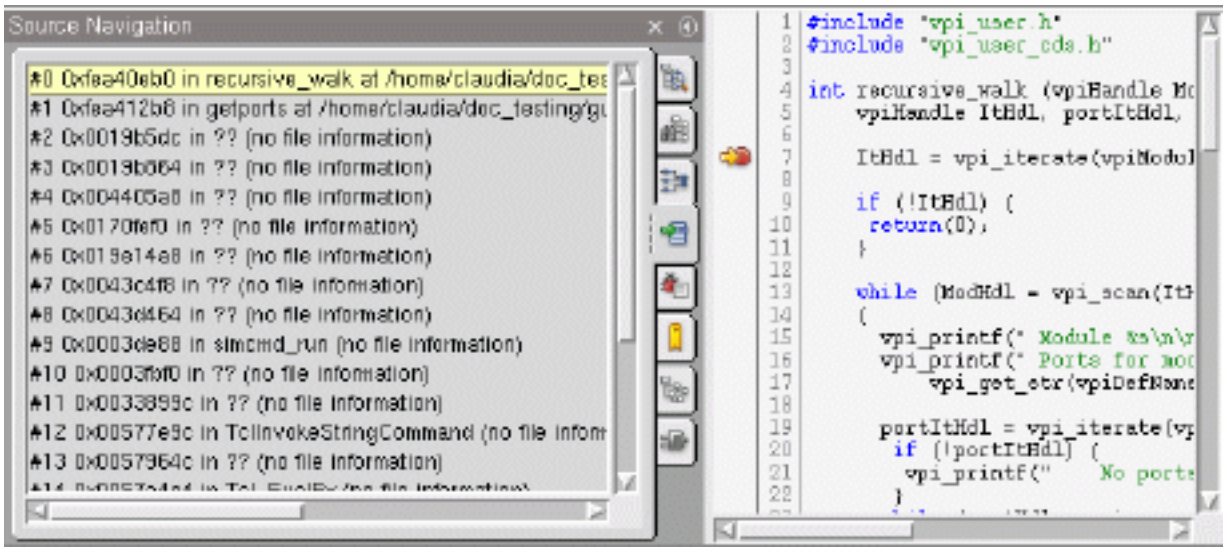
➤ Click *Interrupt C Debugger*, . This sends a `Control-C` to the debugger and returns control to GDB. If GDB is already in control, the button is disabled in the Source Browser window.

You can operate on any HDL object (SystemC, Verilog, or VHDL) while GDB is in control. You can navigate to other portions of your design and set line breakpoints in Verilog or VHDL, create object breakpoints on simulation objects, or issue other simulator Tcl commands. These commands are passed to the simulator through GDB, and their output appears in the simulator console.

Viewing the SystemC/C++/C Call Stack

SimVision can show the SystemC/C++/C call stack when simulation is stopped within GDB. The call stack shows the call stack frame number, followed by the function name and location, as shown in Figure 13-3 on page 165.

Figure 13-3 API Call Stack in the Source Navigation Sidebar



To access the call stack:

Advance simulation into the API code, then open the *Call Stack* tab in the Source Browser window.

When you click on a frame in the call stack, the frame becomes active, thus allowing you to move up and down the call stack. The Source Browser jumps to the selected function in the source file.

When simulation enters an API function, its name is added to the call stack. When the function exits, its name is removed from the call stack. If there is nothing on the call stack at the current execution point, the sidebar displays `(call stack empty)`.

Viewing API Function Parameters and Local Variables

You can view function parameters and local variables in the Source Browser's SystemC/C++/C Variables sidebar. The sidebar displays these objects and their values. You can expand a complex object to see the values of its fields.

To view API function parameters and local variables:

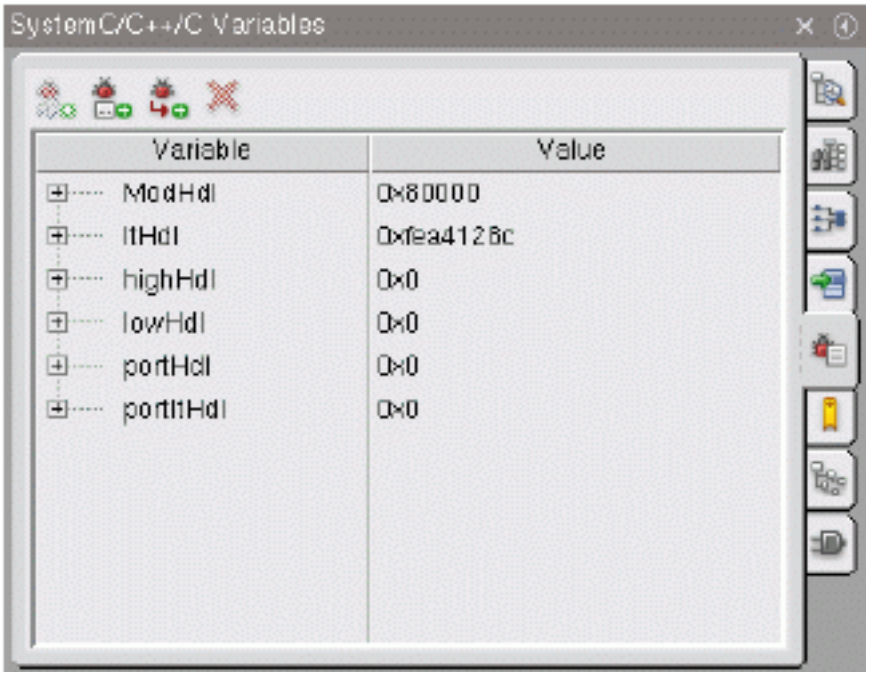
Advance simulation into the API code, then open the *SystemC/C++/C Variables* sidebar in the Source Browser window.

Initially, the sidebar is blank. To add objects to the sidebar:

- Select a parameter or variable in the Source Browser and click *Add Variables*,  .
- Click *Load Parameters*,  , to add all function parameters and their values.
- Click *Load Local Variables*,  , to add all local variables and their values.

Figure 13-4 on page 167 shows the sidebar with parameters and variables loaded into it.

Figure 13-4 The SystemC/C++/C Variables Sidebar



To remove objects from in the sidebar:

➤ Select the object and click *Delete*, ✖ .

SimVision updates the sidebar whenever simulation stops within GDB. If the simulation advances into the HDL code, the values of the local parameters are not updated until control returns to GDB.

Handling Signals When Debugging

The simulator makes use of specific operating system signals to handle unusual simulation conditions or to implement debugging functionality. The simulator can generate and handle the following UNIX/Linux signals during normal operation: `SIGILL`, `SIGALRM`, `SIGSEGV`, `SIGBUS`, `SIGINT`, `SIGFPE`, `SIGEMT`, and `SIGPROF`.

Note: Handling signals as described in this section is necessary only if you start GDB or DBX manually from the command line. Handling signals is not required if you are using SimVision in native debugging mode.

C and C++ debugging tools usually catch UNIX and Linux signals by default. They return control to the debugger user interface when such signals are raised. This is because these signals often indicate a problem with the C++ or C code. Debugging tools also provide

commands to change the default behavior, so that you can continue debugging without interruption by such signals. For example, you can use the `handle` or `ignore` commands for this purpose in GDB or DBX, respectively.

Generally, you should ignore any UNIX signals that are not raised by your C or C++ source code. To do this, you need to set up the debugger accordingly when you start debugging.

Use the following sequence of commands for GDB:

```
(gdb) handle SIGSEGV SIGBUS SIGFPE SIGALRM SIGEMT nostop noprint
(gdb) break nc_signal_raised
(gdb) commands 1
(gdb) handle SIGSEGV SIGBUS SIGFPE stop print
(gdb) cont
(gdb) end
```

Use the following sequence of commands for DBX:

```
(dbx) ignore SIGSEGV SIGBUS SIGFPE SIGALRM SIGEMT
(dbx) when in nc_signal_raised { up 4; down; stop; }
```

When you set a breakpoint at `nc_signal_raised`—a routine in the simulator `main` function—you invoke the simulator functionality that causes the debugger not to stop when one of the listed signals is raised by the simulator or by UNIX, and to stop at signals that are raised by your C++ code.

As part of the SimVision integration with GDB, this set of commands is included in the debugger startup procedure. So when you invoke GDB from SimVision, a breakpoint on `nc_signal_raised` is already set in the simulator `main` function. However, if you want to apply the same methodology of handling signals to debugging the elaborator (the *ncelab* executable) in command-line mode of debugging (GDB or DBX prompt), you must set this breakpoint manually because it is not possible to run elaboration with SimVision.

You can also determine if a signal is generated by your code with the C++ `where` command. After you have compiled your code with the `-g` option, when such a signal is raised, type `where`. If the stack trace includes any function names in your source code, the signal was raised by the code. If you cannot recognize any of the function names, or if there are none in the stack trace, then the signal is a normally generated *ncsim* signal, and you can continue debugging.

If you use GDB, simply type `cont` to continue from the execution point of the debugged process. A simulator internal error on continuing indicates that the signal is generated by your code, and you should consider debugging it.

14

Viewing a Design Schematic

The Schematic Tracer displays abstract RTL models and gate-level designs in schematic form.

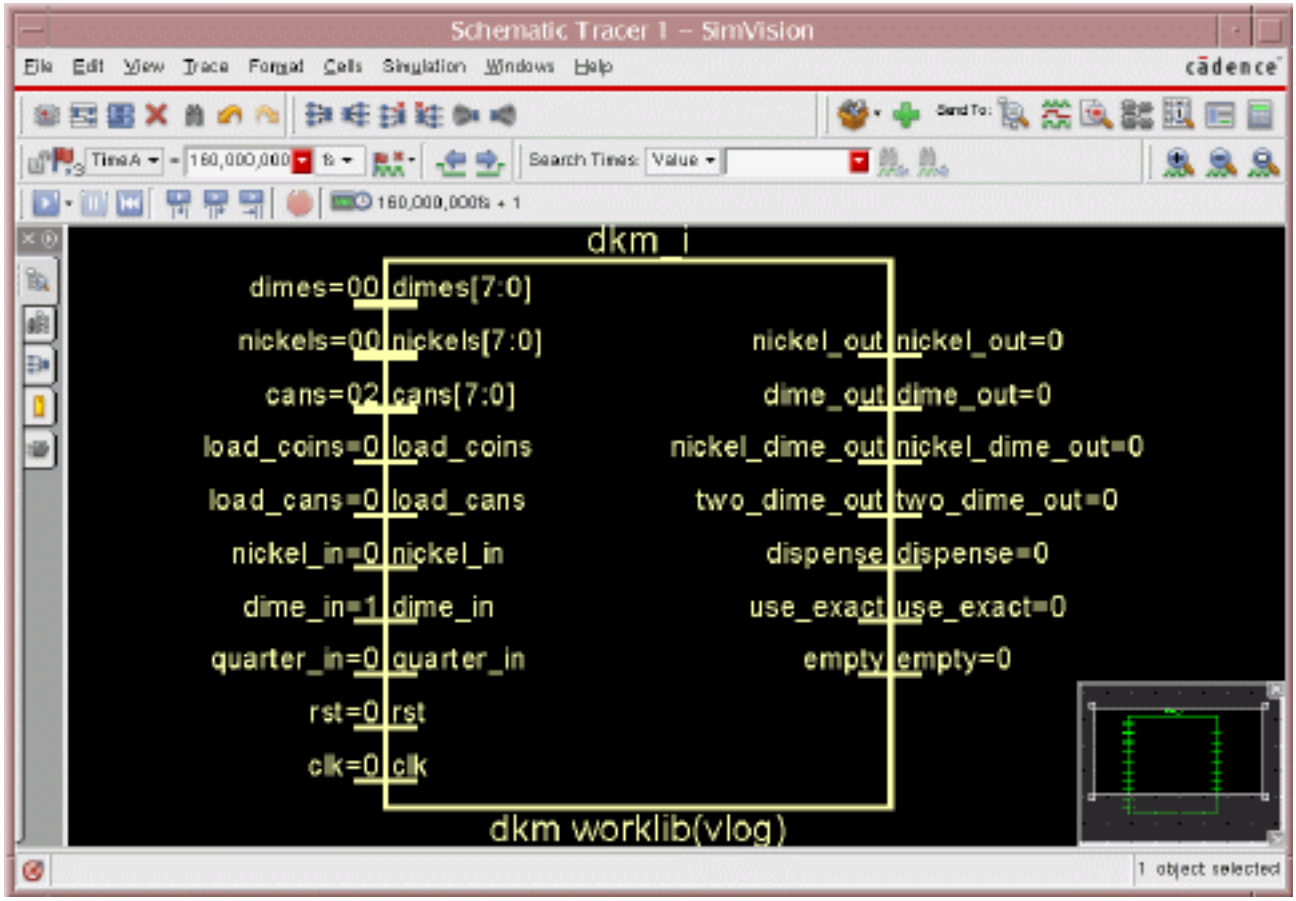
Opening a Schematic Tracer Window

To open a Schematic Tracer window:

➤ In the Send To toolbar of any SimVision window, click *Schematic Tracer*,  , or choose *Windows – New – Schematic Tracer* from the menu bar.

If you select a scope or a signal before opening the window, the Schematic Tracer displays the selected scope or the scope of the selected signal, as shown in Figure 14-1 on page 170.

Figure 14-1 Schematic Tracer Window



Adding Objects to the Schematic

To add objects to the schematic:

- Select scopes or signals in any SimVision window, then click *Schematic Tracer*, , in the Send To toolbar or click *Add*, , in the Schematic Tracer window.
- Open the Design Browser or Design Search sidebar and send scopes or signals to the window from the sidebar, as described in Chapter 5, “Accessing Design Objects.”

Adding Details of a Scope

After you add a scope to the Schematic Tracer, you can add its parent scope, all of its subscopes and signals, and the connections between them.

To add a parent scope:

- Select the child scope, then click *Display Parent Module*, , or choose *Edit – Ascend* from the menu.
- Select the scope, then press and hold the left mouse button and drag the mouse up and to the left. The Schematic Tracer draws a red line labeled *Ascend*. When you release the mouse, the Schematic Tracer adds the parent scope to the display.

To add subscopes and signal connections:

- Select the scope and click *Fill Modules*, , or choose *Edit – Fill Modules* from the menu bar, or right-click the scope and choose *Fill*. This adds any subscopes, blocks, and internal signals defined in the selected scope.
- Select the scope and choose *Edit – Descend into Module*. This fills the selected scope and removes any parent scopes from the display. This is useful when you have a large design and want to focus on only one part of the design.

You can also set preferences in the Schematic Tracer tab of the preferences window to automatically fill instances and modules when you add them to the Schematic Tracer window. See "Schematic Tracer" on page 321 for more information.

Whenever you add objects to the schematic, the Schematic Tracer highlights them. By default, they are displayed in red. You can change the highlight color, as described in "Schematic Tracer Colors" on page 322.

To remove highlighting:

- Choose *Edit – Clear Highlights.*

Selecting Objects in the Schematic

There are several ways to select objects in the Schematic:

- Click to select a single object.
- `Shift`-click to select several objects.
- Press the `Control` key and drag the mouse to select objects within an area of the design.
- Use the *Edit – Select* menu to select all objects in the schematic, or to select all highlighted objects, or to deselect all selected objects.

Removing Scopes from the Schematic

To remove individual scopes from the schematic:

➤ Select the objects that you want to remove and click *Delete*, ,or choose *Edit – Delete Modules* from the menu bar. Signal paths that would pass through a removed scope are displayed with a broken line.

To remove the details of a scope:

➤ Select the scope, then click *Collapse Modules*, , or choose *Edit – Collapse Modules* from the menu bar.

To redisplay the schematic after you have changed its contents, such as removing a module or signal:

➤ Choose *View – Regenerate*.

Using Color in the Schematic Diagram

The Schematic Tracer displays signal names and values in the following colors:

- X values are displayed in red
- Non-X values are displayed in green
- Signals with no value are displayed in white

You cannot change these colors. However, you can change the color of a selected area in the design. This may be useful if you are going to print a schematic diagram and want to draw attention to one area of the design.

To specify the color of an area in the design:

1. Select the scopes and paths that you want to color and choose *Format – Color* from the menu bar. The Schematic Tracer displays the Color menu.
2. Select a color from the palette, click *More* to mix your own color in the Select a New Color form, or click *Default* to return the selected area to its default color.

Displaying Signal Names

You can display either the simple name of a signal or its full name.

To change the format of signal names:

- Enable *View – Show Full Signal Names* to display the full hierarchical name of signals.
- Disable *View – Show Full Signal Names* to display the simple name of signals.

Displaying Signal Values

The Schematic Tracer lets you display signal values in any radix that you choose, and it lets you watch signal values change during a live simulation or as you jump from edge to edge in a simulation database.

If *Display signal transition values* is enabled in the Signal Options tab of the Preferences window, the Schematic Tracer displays two values when the value of an object changes during the current clock cycle. For example, if a signal transitions from 0 to 1 in the current clock cycle, SimVision displays the transition as `'b0 -> 'b1`. If *Display signal transition values* is disabled, the Schematic Tracer displays only the value of the signal at the end of the clock cycle.

To display signal values in the Schematic Tracer window:

- Choose *View – Show Values* from the menu bar. The Schematic Tracer adds signal values to the schematic. If the signal value changes during the current clock cycle, the transition is displayed with the notation *`old-value -> new-value`*.

  When *Watch Live Data* is enabled,  , values in the Schematic Tracer are not updated until the simulation pauses or stops.

To view signal value changes during simulation:

- Select a signal and click *Next Edge*,  . Simulation progresses up to the point where the signal value changes. Simulation is interrupted, similar to setting a breakpoint on the object, and the values of signals in the window are updated.

  You can click *Next Edge* again, or select another signal and simulate to the next edge of that signal.

To view signal values in a database:

- Select a signal and click *Next Edge*,  , to jump to the next signal transition.
- Select a signal and click *Previous Edge*,  , to jump to the previous signal transition.

Setting the Radix of Values

By default, signal values are displayed in the radix in which they are recorded.

To change the radix:

➤ Select a radix from the *Format – Radix* menu. The new radix is applied to all signals in the window. *As recorded* displays signals in the radix in which they were recorded during simulation.

Zooming the Schematic

Several toolbar buttons, menu entries, and keyboard shortcuts let you zoom in and out on the Schematic Tracer window.

Button	Menu Entry	Keyboard Shortcut
	*View – Zoom – In*	`i`
	*View – Zoom – Out*	`o`
	*View – Zoom – Fit*	`f`

You can also zoom in and out by pressing and holding the left mouse button, then dragging the mouse in one of the following directions:

Up and right	Zoom out incrementally
Down and right	Zoom to fit the select area in the window
Down and left	Zoom to fit the entire schematic in the window

Using the Panner

You can use the scroll bars at the bottom and right sides of the window to view areas of the schematic that are not currently displayed in the window. However, the Schematic Tracer also has a panner that gives you more panning and zooming features.

Video

Using the Panner

The panner is a small representation of the entire schematic, with a box sourrounding the area currently displayed in the window. You can use the panner to zoom in and out and display different areas of the schematic.

To show the panner:

➤ Click the down arrow in the upper left corner of the window,  , or enable *View – Panner* in the menu bar.

To zoom in and out:

➤ Press and hold the left mouse button over one corner of the box, then drag the corner to make the box larger (zoom out) or smaller (zoom in).

To display a different area of the schematic:

➤ In the panner, click the area that you want to view, or press and hold the left mouse button and drag the box to the new location.

To hide the panner:

➤ Click the up arrow in the upper left corner of the panner,  , or disable *View – Panner* in the menu bar.

To remove the panner entirely:

➤ Click *Close Panner*,  , in the upper right corner of the panner, or disable *View – Panner* in the menu bar.

Searching for Objects

When the window contains many signals and scopes, you might have difficulty finding a specific instance or signal.

To search for signals and instances:

1. Click the *Search* button,  , or choose *Edit – Search* from the menu bar. The Schematic Tracer opens the Search form.
2. Select the type of object you want to search for, either *Instance* or *Signal*.
3. Enter the name or partial name of the object you want to find. By default, the search string is *, which matches any name.
4. Click *Search,* and the Schematic Tracer displays a list of all instances or signals whose names match the search string.
5. Single-click a name in the list, and the Schematic Tracer highlights the object in the schematic diagram. Double-click a name, and the Schematic Tracer zooms in and highlights the object.

Tracing Paths with the Schematic Tracer

The Schematic Tracer provides several ways to trace a signal path through the design. When a signal has an unexpected value, you can trace the driving logic for the signal back through the design to see the signals that contribute to that value. Similarly, if you suspect that a signal contributes to an incorrect value in another part of the design, you can trace the loading logic from that signal forward through the design. You can also trace a scope to see all of the drivers or all of the loads for that block or module.

The Schematic Tracer toolbar contains the following trace buttons:

Traces a signal back through the design, or traces all drivers of a scope. You can also perform this trace function by choosing *Trace – Driving Logic* from the menu bar or *Trace Driving Logic* from the pop-up menu.

Traces a signal forward through the design, or traces all the loads of a scope. You can also perform this trace function by choosing *Trace – Loading Logic* from the menu bar or *Trace Loading Logic* from the pop-up menu.

Traces the driving logic back to the cause of an X value. You can also perform this trace function by choosing *Trace – X Backwards* from the menu bar or *Trace-X Backwards* from the pop-up menu.

The Schematic Tracer cannot trace the driving logic if there are multiple X values along the path.

Traces the loading logic forward from the X value. You can also perform this trace function by choosing *Trace – Follow-X Forwards* from the menu bar or *Follow-X Forwards* from the pop-up menu.

Traces back to the current debug scope boundary, through any intervening modules or blocks. You can also perform this trace function by choosing *Trace – Logic Cone Backwards* from the menu bar or *Logic Cone Backwards* from the pop-up menu.

Traces forward to the current debug scope boundary, through any intervening modules or blocks. You can also perform this trace function by choosing *Trace – Logic Cone Forwards* from the menu bar or *Logic Cone Forwards* from the pop-up menu.

Trace – Trace Signal Through Hiererarchy

Traces the signal through the entire design (both drivers and loads). This option is available only through the *Trace* menu and the pop-up menu, not as a button in the Trace toolbar.

Important

When the Trace Signals sidebar is open, the trace buttons behave in the same way as the trace buttons in the Trace Signals sidebar. That is, you cannot trace a scope, and when you click the trace buttons, a new trace path is added to the Trace Signals sidebar. This change affects the *Trace* and the *Trace X* buttons. The *Trace Logic Cone* buttons behave in the usual way. For more information, see Chapter 16, "Tracing Paths with the Trace Signals Sidebar."

Tracing Paths from Point to Point

The Schematic Tracer can trace the path a signal between two points, as follows:

- Tracing the Path between Two Pins
- Tracing a Path between a Pin and a Scope
- Tracing the Path between a Pin and a Cell

You can trace either forward or backward through the design—from an input to an output, or from an output to an input.

If the schematic information along the traced path is not displayed in the Schematic window, the Schematic Tracer adds the necessary information to the window.

Important

The point-to-point tracing feature cannot trace a path through a bus.

Tracing the Path between Two Pins

To trace a path between two pins:

1. Select the pin at the beginning of the path that you want to trace, then hold down the `Control` key and select the pin at the end of the path.
2. Choose *Trace – Point to Point*. The Schematic Tracer opens the Point To Point Tracing form.

   The first *Name* field contains the full hierarchical name of the first signal that you selected; the second *Name* field contains the second signal. The *Type* field specifies the type of object selected.

   You can also enter the signal names directly in the *Name* fields.
3. In the *Direction for this tracing* field, choose *Input* to trace back through the design toward the primary inputs, or *Output* to trace forward through the design toward the primary outputs.

   The order in which you select the pins determines the direction that you should choose. For example, if you select an input pin as the first point and an output pin as the second point, you will ordinarily set the direction to *Output*. If you select an output pin as the first point and an input pin as the second point, you will ordinarily set the direction to *Input*. Otherwise, the Schematic Tracer will try to a trace loop condition, such as an output signal driving an input signal.
4. If you want to display the results in a new Schematic window, enable *Perform the trace in a new window*. Otherwise, the current window is updated with the trace path.
5. Click *OK* to trace the path between the two pins. The Schematic Tracer displays the path in pink.

Tracing a Path between a Pin and a Scope

To trace a path between a pin and a scope:

1. Select the pin, then hold down the `Control` key and select the scope.
2. Choose *Trace – Point to Point*. The Schematic Tracer opens the Point To Point Tracing form.

The first *Name* field contains the full hierarchical name of the pin that you selected; the second *Name* field contains the name of the selected scope. The *Type* field contains the type of object selected.

You can also enter the signal and scope names directly in the *Name* fields.

3. In the *Direction for this tracing* field, choose *Input* to trace back through the design toward the primary inputs, or *Output* to trace forward through the design toward the primary outputs.
4. If you want to display the results in a new Schematic window, enable *Perform the trace in a new window*. Otherwise, the current window is updated with the trace path.
5. Click *OK*, and the Schematic Tracer displays the traced path in pink.

   If there are multiple paths between the selected pin and scope, the Schematic Tracer displays all of those paths, if possible.

Tracing the Path between a Pin and a Cell

To trace the path between a pin and a cell:

1. Select the pin, then choose *Trace – Point to Point*. The Schematic Tracer opens the Point to Point Tracing form.

   The first *Name* field contains the full hierarchical name of the pin that you selected. The *Type* field specifies the type, *Pin*.

   You can also enter the signal name directly in the *Name* field for the starting point.
2. In the second *Name* field, enter the cell name and set the *Type* to *Cell*.

   The name can contain wildcard characters. For example, specify `ff*` to trace between the pin and any cell instance whose name contains the string `ff`. The name does not have to be a library cell; it can be a module definition (not a module instance).
3. In the *Direction for this tracing* field, choose *Input* to trace back through the design toward the primary inputs, or *Output* to trace forward through the design toward the primary outputs.
4. If you want to display the results in a new Schematic window, enable *Perform the trace in a new window*. Otherwise, the current window is updated with the trace path.
5. Click *OK*, and the Schematic Tracer displays the path in pink.

Using Bookmarks in the Schematic Tracer

In the Schematic Tracer, a bookmark saves the contents of the window—that is, the scopes and interconnections currently displayed. You can use the bookmark to restore the view at a later time. You create, rename, reorder, and delete bookmarks in the Bookmarks sidebar.

Bookmarks are automatically saved when you exit SimVision and restored when you run SimVision again from the same design directory.


Video

Schematic Tracer Bookmarks

To open the Bookmarks sidebar:


Click *Bookmarks* in the Schematic Tracer sidebar, or choose *View – Manage Bookmarks* from the menu bar.

To create a bookmark:

➤ Click *Add a Bookmark*,  , in the sidebar, or choose *View – Add to Bookmarks* from the menu bar.

The Schematic Tracer assigns a unique name to the bookmark, comprised of the string `Schematic` followed by a unique number.

To rename a bookmark:

➤ Double-click the bookmark to open a text area where you can edit the name; press `Return` to close the text area.

To reorder the list of bookmarks:

➤ Press and hold the middle mouse button over the bookmark, then drag the bookmark to a new location in the list.

To delete a bookmark:

➤ Select the bookmark and click *Delete*,  .

To return to a view that you have bookmarked:

➤ Select the bookmark in the sidebar.

The Properties window lists all of the bookmarks that you create. You can use the Properties window to rename and delete bookmarks.

To manage bookmarks in the Properties window:

1. Click *Properties*, , in the Send To toolbar and select *Bookmarks* in the lefthand pane of the window, or click *Display Bookmark Properties*, , in the Bookmarks sidebar.
2. Select *Schematic Tracer* in the Bookmarks tab of the Properties window. This displays the Schematic Tracer bookmarks that you created.
3. You can perform the following operations on bookmarks:
   - Double-click on a bookmark to edit its name.
   - Select one or more bookmarks and click *Delete*, , to delete the bookmarks.
   - Drag and drop bookmarks to change their order in the list.

Printing and Saving the Schematic Window

To print or save the Schematic window:

1. Choose *File – Print Window* from the menu bar to open the Print form.
2. If you want to print the window, select *Printer* and enter the appropriate print command. If you want to save the window to a file, select *File* and enter the filename or click *Browse* to choose a file. By default, the Schematic Tracer names the file `schematic.ps`.
3. Set the *Scale* to *Full* if you want to print or save the entire schematic (similar to a zoom full operation). Set *Scale* to *View* if you want to print or save only the portion of the schematic displayed in the window.
4. Set *Color Mode* to *Mono* to print or save the schematic in black-and-white; set *Color Mode* to *Color* to print or save the schematic in full color.
5. Select a paper size.
6. Click *OK*.

15

Controlling the Appearance of Schematic Elements

The Schematic Tracer maps a Verilog or VHDL RTL or gate-level object, or a SystemC structural object to one of the following schematic elements:

- Blocks
- SystemVerilog Interfaces and Modports
- Expressions
- Abstracted RTL Elements
- Gate Primitives
- Source

After you have synthesized a design, all of the modules that correspond to flip-flops, latches, muxes, and other special gate types are displayed in the Schematic Tracer as simple boxes. If you want to display these cells using other shapes, you can create a cell map file.

Blocks

A block represents an HDL construct, such as a Verilog module or a VHDL `process` statement. The block is labeled with the name given to the element in the design. For example, a block for a Verilog module is labeled with the module name and the instance name. If the element is not named, the Schematic Tracer generates a label. For example, an unnamed VHDL process is labeled `$PROCESS_`*`NNN`* and a Verilog `always` block is labeled `block`*`N`*, where *`NNN`* and *`N`* are sequential numbers.

Figure 15-1 on page 184 shows the block schematic element. Inputs appear on the left side of the block, and outputs appear on the right.

Figure 15-1 Block Schematic Element

Table 15-1 on page 184 lists the Verilog and VHDL constructs represented by the block schematic element.

Table 15-1 Block Language Constructs

Verilog	VHDL
Continuous assignment	`block` statement
`always` statement	`process` statement
module instantiation	Concurrent procedure call
	Concurrent signal assignment
	Component instantiation statement
	`generate` statement

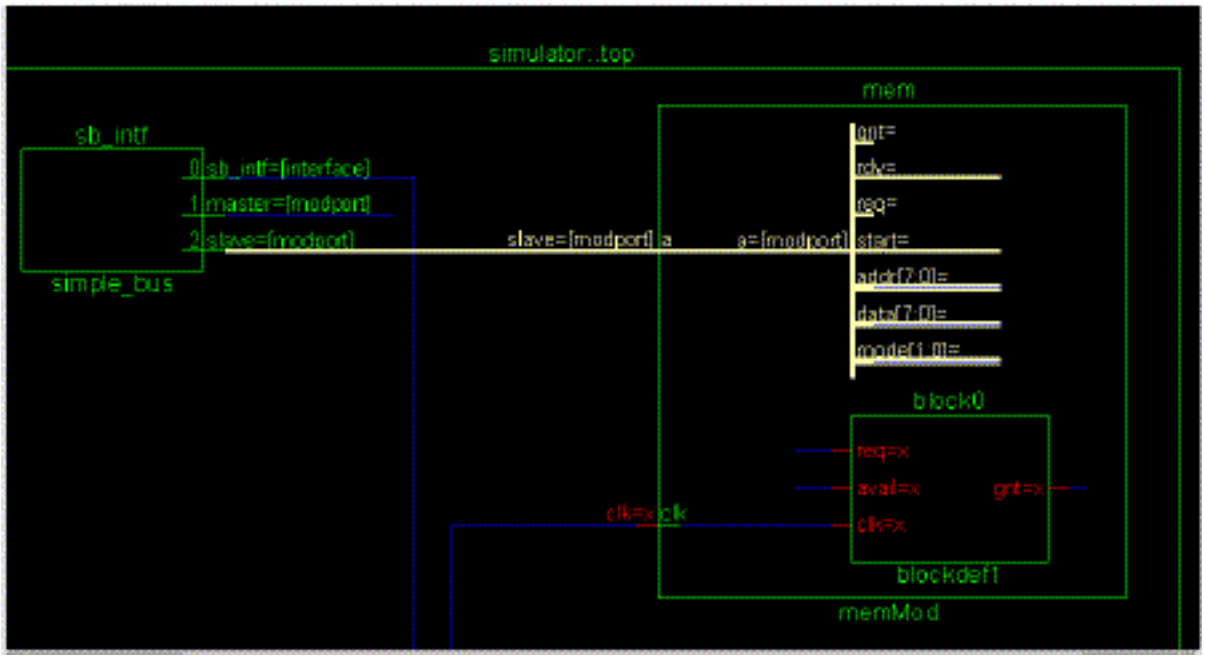
Note: VHDL concurrent assertion statements are ignored by the Schematic Tracer and do not appear in schematic diagrams.

SystemVerilog Interfaces and Modports

Interfaces and modports are displayed as wires labeled with the interface or `modport` name on the outer connection of the block that instantiates it. Inside the block, the wire fans out to show the ports defined in the `interface` or `modport`.

For example, Figure 15-2 on page 185 shows a `modport` named `slave` that is instantiated in the `mem` module. It appears as a wire going into the module, and its ports are displayed within the module.

Figure 15-2 Displaying a SystemVerilog Modport in the Schematic Tracer



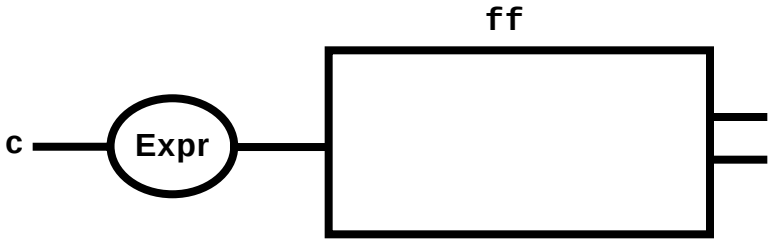
Expressions

An expression schematic element is used when a port is connected to an expression. All of the signals in the expression are shown as inputs to the expression schematic element, and the output feeds the port of the block. For example, the following code defines an expression as an input to a VHDL block:

```
ff: BLOCK (C='1' AND NOT c'STABLE)
  .
  .
  .
```

Figure 15-3 on page 186 shows how the expression is represented in the schematic diagram as an input to the block.

Figure 15-3 Expression Schematic Element



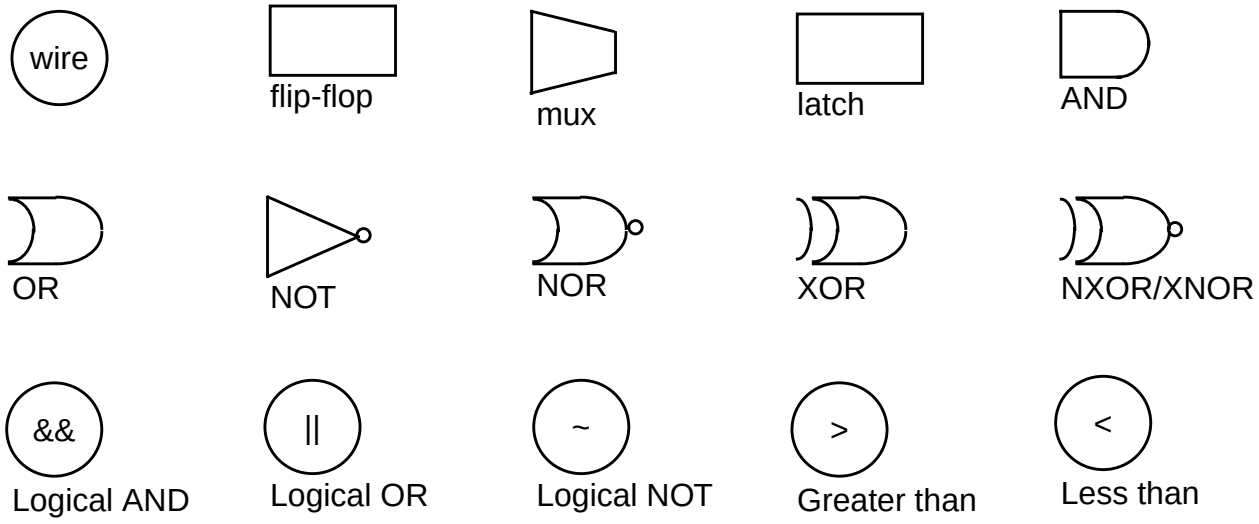
Abstracted RTL Elements

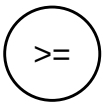
The Schematic Tracer can display logic elements for an RTL design if you have processed the design for Incisive HDL analysis (HAL), or if the design is modeled in such a way that SimVision can extract the logic elements.

You control the display of logic elements, as follows:

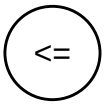
- Enable *View – Show RTL* in the Schematic window. This turns on the display of logic elements in that window only.
- Enable *Show RTL Logic when available* in the Schematic Tracer tab of the Preferences window. This displays logic elements in all new Schematic Tracer windows, by default.

When abstracted RTL information is available, the design is represented using the following logic elements:

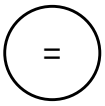




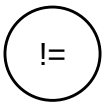
Greater than or equal to



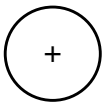
Less than or equal to



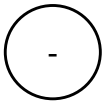
Equal to



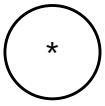
Not equal to



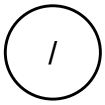
Add



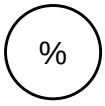
Subtract



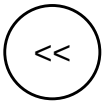
Multiply



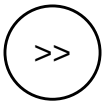
Divide



Modulo



Bitwise shift left



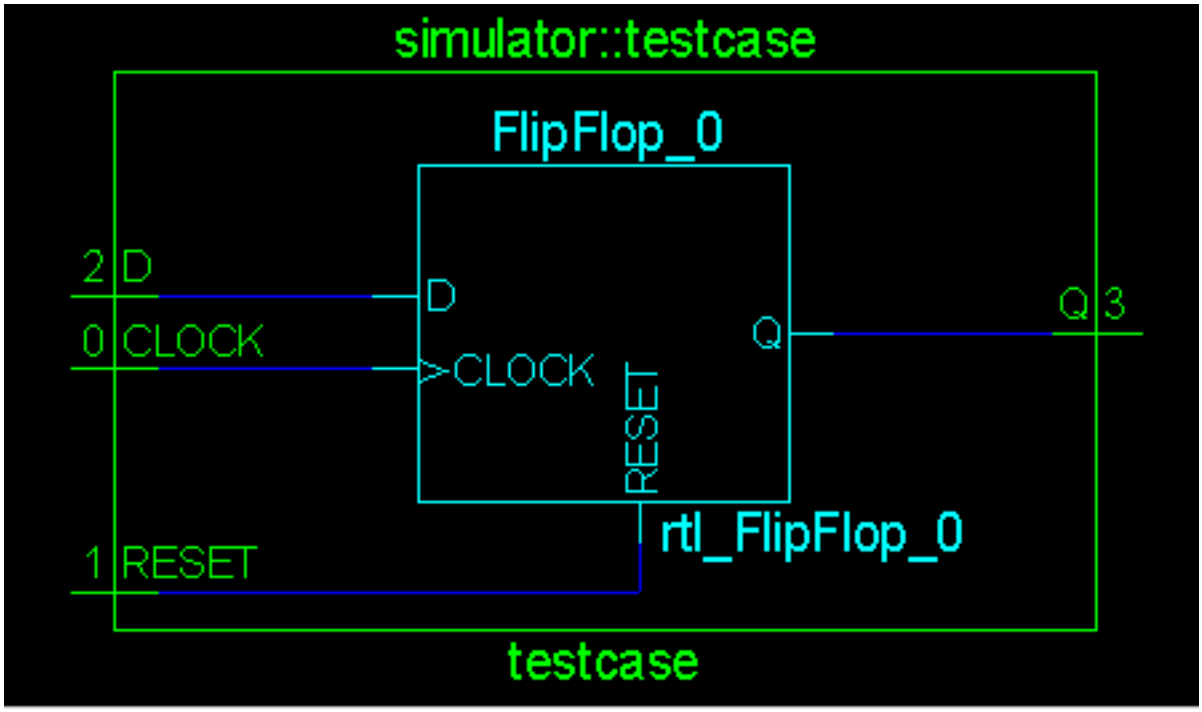
Bitwise shift right

Note: `Triand`, `trior`, `wand`, and `wor` are not supported. They are shown as `Unknown` blocks in the schematic diagram.

Figure 15-4 on page 188 shows the abstracted RTL elements displayed by the Schematic Tracer for the following `always` block:

```
always @(posedge CLOCK or posedge RESET)
    if (RESET)
        Q <= 0 ;
    else
        Q <= D ;
```

Figure 15-4 Displaying Abstracted RTL Elements



Gate Primitives

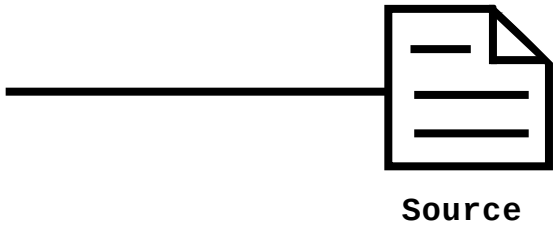
Gate primitives are represented by standard gate schematic elements. The Schematic Tracer supports the Verilog primitives and the following VHDL logic primitives as defined in the IEEE `std_logic_component` package:

ANDGATE	NANDGATE	ORGATE	NORGATE	XORGATE
NXORGATE	XNORGATE	BUFGATE	BUF3S	BUF3SL
WBUFGATE	INVGATE	INV3S	INV3SL	

Source

The Source schematic element, shown in Figure 15-5 on page 189, is used to terminate an unconnected output. You can select this schematic element to locate the output in the Source Browser.

Figure 15-5 Source Schematic Element



Creating a Cell Map File

By default, ASIC library cells, such as AND gates and flip-flops, are displayed as simple boxes. A cell map file defines how you want these cells displayed in the Schematic Tracer.

There are several ways to create a cell map file:

- Enter the cell map definitions in a text file.
- Generate a cell map file template with the Schematic Tracer, then edit the file to specify the shape of each cell type.
- Translate a Liberty (`.lib`) file into a cell map file with the Schematic Tracer.

Format of a Cell Map File

Each line of the map file defines one cell, using the following syntax:

`cell` *`cellname`* `shape` *`shape_type`* `ports [`*`port_list`*`]`

`cellname` Specifies the name of the cell. The name can contain the wildcard characters:

- `*` Matches any number of characters
- `?` Matches a single character

shape_type Specifies one of the following types:

```
flipflop
latch
mux
bufif0
bufif1
and
or
xor
nand
nor
xnor
not
alu
```

port_list Specifies a list of ports in the cell, using the following syntax:

```
{ {name direction attribute} ... }
```

The *name* must match one of the formal ports in the cell.

The valid values for *direction* are:

```
input
output
inout
```

The *attribute* value controls the appearance of the port in the Schematic Tracer. Valid values are:

`normal`, `nnormal`
`set`, `nset`
`reset`, `nreset`
`clock`, `nclock`
`enable`, `nenable`
`selector`, `nselector`

Attributes that begin with the letter n are displayed with a bubble, to imply negation.

If you do not specify a port list, all of the ports are assumed to be `normal`.

You do not need to specify the width of ports; it is determined by the Schematic Tracer when it draws the schematic.

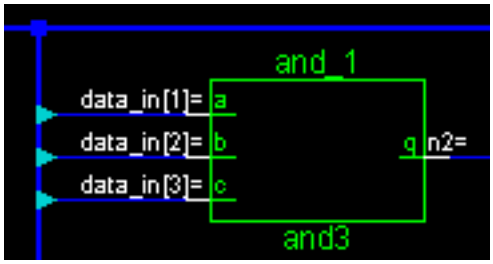
The cell map file can also contain an `include` directive, which specifies the name of a file you want to process along with the file that includes it. The included file can be a SimVision cell map file or a Debussy symlib file.

For example, suppose you define a module, `and3`, as follows:

```
module and3 (a, b, c, q);
input a,b,c;
output q;

    assign q = a & b &c;
endmodule
```

Without cell mapping, instances of `and3` would display as simple blocks in the Schematic Tracer, as shown in Figure 15-6 on page 191.

Figure 15-6 and3 without Cell Mapping



In a cell map file, you can specify that instances of `and3` should be displayed as AND gates, as follows:

```
cell and3 shape and ports
    {{a input normal} {b input normal} {c input normal} {q output normal}}
```

With cell mapping enabled, instances of `and3` are displayed as AND gates, as shown in Figure 15-7 on page 191.

Figure 15-7 and3 with Cell Mapping

Now suppose you have defined the modules, `xor1` and `xor2`, as follows:

```
module xor1(a, b, q);
    input a, b;
    output q;
```

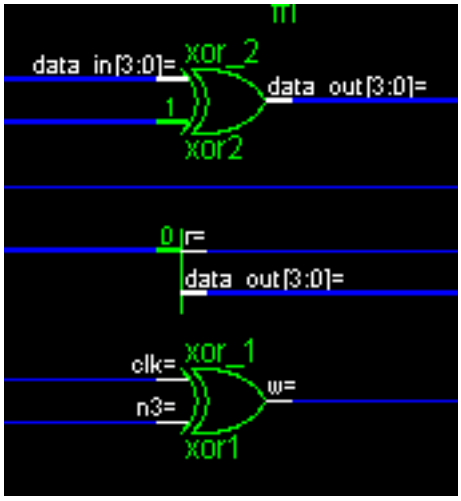
```
    assign q = a;

module xor2(a, b, q);
    input [3:0] a;
    input [3:0] b;
    output [3:0] q;

    assign q = a^b;
```

You can use the * matching character to specify that instances of these modules (and any others that begin with `xor`) are displayed as XOR gates, as follows:

```
cell xor* shape xor ports {{a input normal} {b input normal} {q output normal}}
```

With cell mapping enabled, instances of `xor1` and `xor2` are displayed as shown in Figure 15-8 on page 192.

Figure 15-8 xor1 and xor2 with Cell Mapping



Tip

You do not need to specify all pins in a cell, only special pins, such as clock, set, and reset pins. If a pin is not specified in the cell map file, the Schematic Tracer assumes it is of type `nnormal` with the direction specified in the design.

Creating a Cell Map Template

To create a cell map template:

1. Choose *Cells – Generate Cell Template* from the Schematic Tracer menu bar. The Schematic Tracer opens the Generate Cell Shapes Template form.

2. Enter the name that you want to give to the cell map file, or click *Browse* (*...)* and choose an existing file (this file will be overwritten). Then click *OK* to generate the template file.
3. Open the template file with any text editor. The template contains a cell definition for each cell type in the current snapshot, including the cell name and port list.
4. Remove the lines for any cells that you do not want to map. For each cell that you want to map, replace the string <<unknown>> with the appropriate shape.

For example, the Schematic Tracer creates the following template cell definitions for the and3, xor1, and xor2 modules described in the previous section:

```
cell and3 shape <<unknown>> ports {{a input normal} {b input normal} {c input
normal} {q output normal))
cell xor1 shape <<unknown>> ports {{a input normal} {b input normal} {q output
normal}}
cell xor2 shape <<unknown>> ports {{{a[3:0]} input normal} {{b[3:0]} input normal}
{{q[3:0]} output normal}}
```

For and3, you would replace <<unknown>> with the and shape; for xor1 and xor2, you would specify the xor shape. In addition, because both xor1 and xor2 have the same call map definition, you can remove one of the definitions and use the * wildcard character in the name. This would give you the following cell map definitions:

```
cell and3 shape and ports
    {{a input normal} {b input normal} {c input normal} {q output normal}}
cell xor* shape xor ports {{a input normal} {b input normal} {q output normal}}
```

"Format of a Cell Map File" on page 189 describes the syntax of a cell map definition, including the valid cell shapes that you can use.

Translating a Liberty File into Cell Map File

To translate a Liberty (.lib) file:

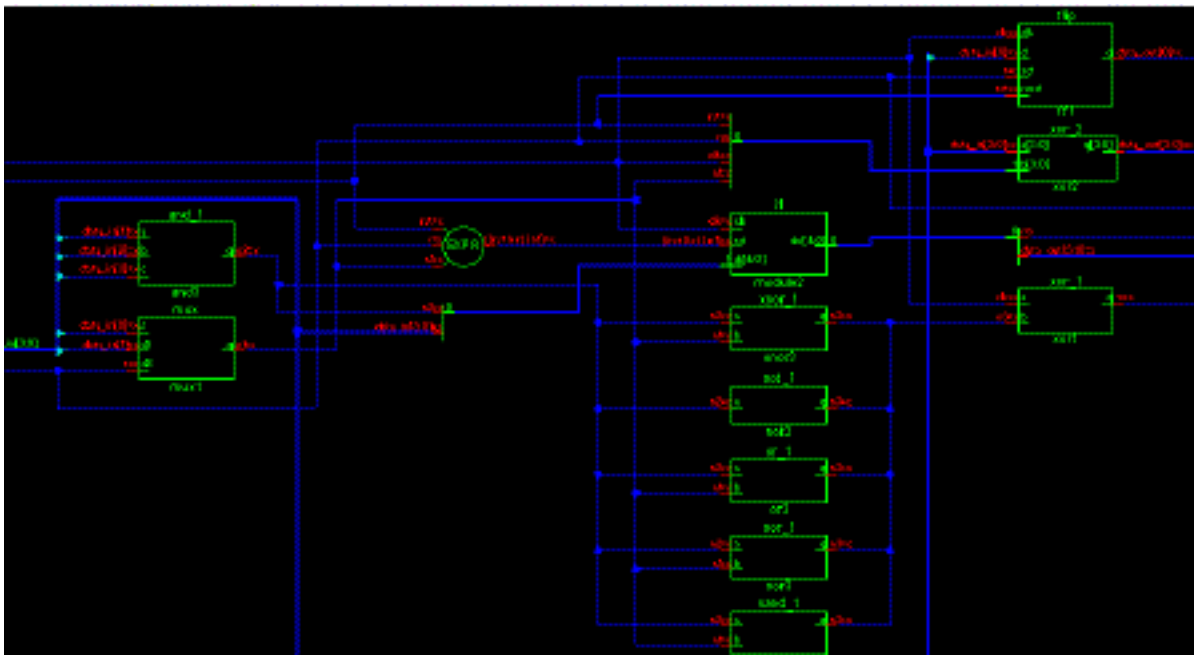
1. Choose *Cells – Translate .lib File* from the Schematic Tracer menu bar. The Schematic Tracer opens the Translate .lib to cell map format form.
2. Enter the name of the .lib file and the translated cell map file, or use the *Browse* (...) buttons to choose these files.
3. Click *OK*.

The Schematic Tracer does not translate all pins in a cell, only special pins, such as clock, set, and reset pins. If a pin is not specified in the cell map file, the Schematic Tracer assumes it is of type nnormal with the direction specified in the design.

Viewing Mapped Cells in the Schematic

Figure 15-9 on page 194 shows a design whose cells are unmapped. The flip-flops, muxes, and other special gates are displayed as simple boxes.

Figure 15-9 Unmapped Cells in a Design Schematic



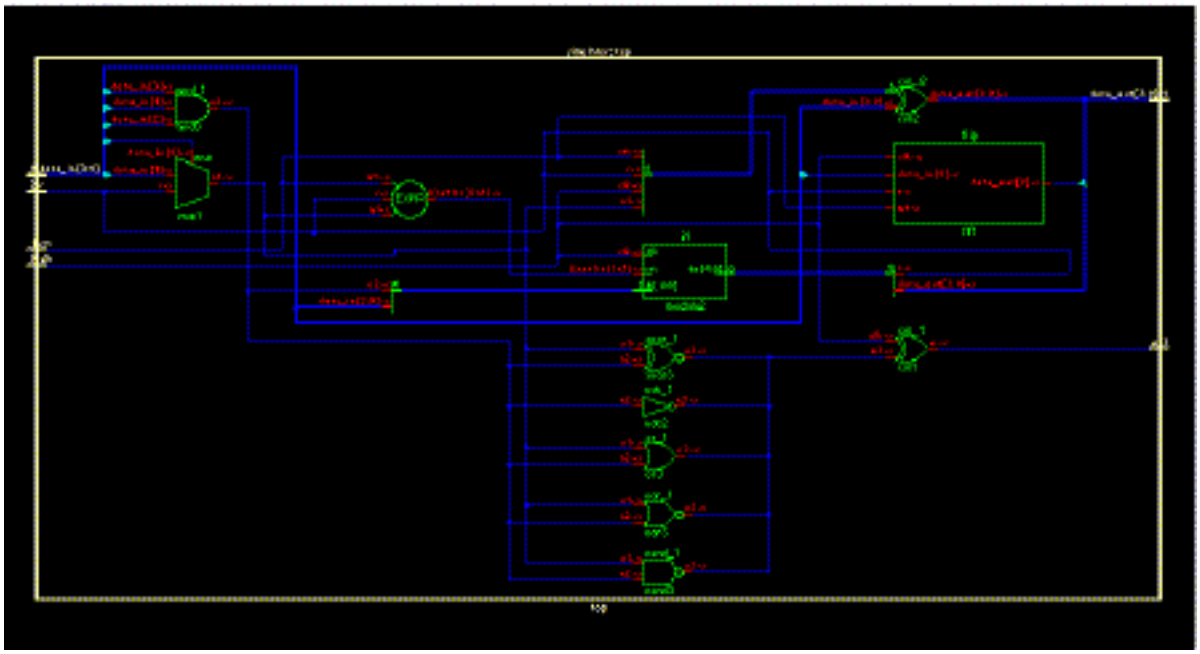
To display these cells with a cell map file:

1. In the Schematic Tracer tab of the Preferences window, enable the *Show Cell Shapes if available* option, and specify the cell map file name in the *Cell Shapes Mapfile* field. Then click *Apply* or *OK*.
2. In the Schematic Tracer window, enable *View – Show Cell Shapes* in the Schematic Tracer menu bar. The schematic is redrawn with the new cell shapes, as shown in Figure 15-10 on page 195.

Tip

The *View – Show Cell Shapes* setting in the Schematic Tracer window lets you turn cell mapping on and off in individual Schematic Tracer windows. However, you cannot turn on cell mapping in the window unless you have also enabled *Show Cell Shapes* in the Preferences tab.

Figure 15-10 Mapped Cells in the Design Schematic



Handling Errors in the Cell Map File

If the cell map file contains syntax errors, the Schematic Tracer renders the schematic diagram as best it can with the information available, as follows:

- If a cell name is missing, no mapping is done.
- If a shape is missing, or has an unknown type, it is displayed with the default RTL symbol—a circle.
- If the port list specifies the wrong type, the type defaults to `normal`.
- If the port list specifies the wrong direction, the direction defaults to the direction defined in the design.

If your design does not appear to be using the shapes that you have specified in the map file, you may have an error in the cell map file.

To locate an error in the map file:

➤ Choose *Cells – Show Cell mapfile Warnings*. The Schematic Tracer opens a Cell Shapes Warning window. For each error in the cell map file, this window displays the file name, line number, and a description of the problem in the cell map definition.

Updating a Cell Map File

To apply a modified map file to your existing Schematic Tracer windows:

➤ Toggle the *View – Show Cell Shapes* option in the Schematic window off and on.

The map file changes take effect immediately in all existing Schematic Tracer windows that are open, and in any new ones that you create.

16

Tracing Paths with the Trace Signals Sidebar

After you have probed some signals and simulated for a period of time, you may see a value that is unexpected. When this happens, you can use the Trace Signals sidebar to trace the signal through the design to see where the value has come from, or to see what effect that value may have on the logic that it drives.

The Trace Signals sidebar lets you trace the driving or loading logic for a signal, and it lets you automatically trace all paths for X values and active drivers to and through registers, cells, and module boundaries.

Accessing the Trace Signals Sidebar

To access the Trace Signals sidebar:

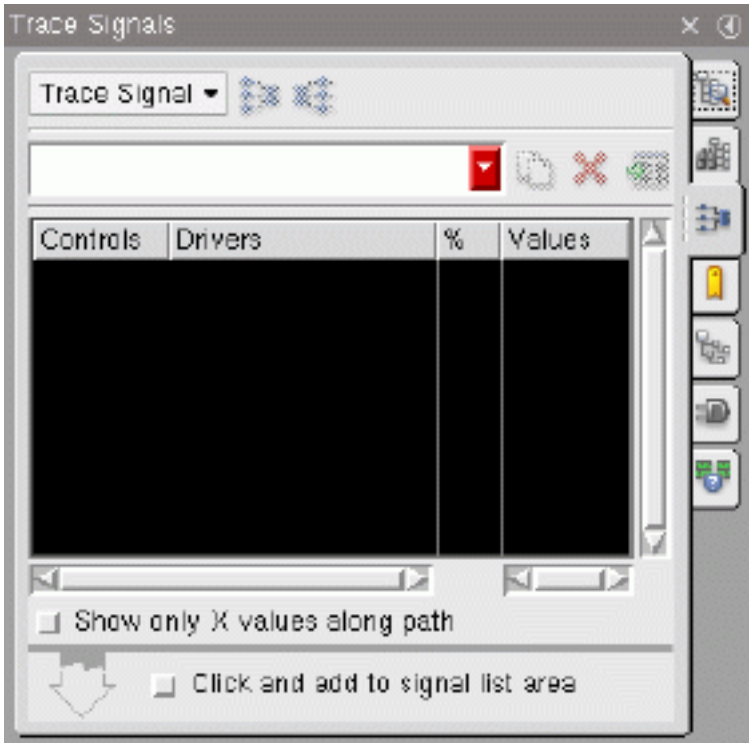
Click the *Trace Signals* tab.

Tip

If the sidebar is not visible, first enable the *Sidebar* option in the *View* menu, then click the *Trace Signals* tab.

The Trace Signals sidebar is shown in Figure 16-1 on page 198.

Figure 16-1 Trace Signals Sidebar



The sidebar contains controls for tracing signals, a drop-down menu of the paths you have traced, and an area to display the signals in the current path, including their values at the current simulation time. If the value of a signal changes during the current point in time, the transition is displayed with the notation *`old-value -> new-value`*.

Tracing the Loading Logic

Video

Tracing Loading Logic in the Trace Signals Sidebar

To trace the loading logic for a signal:

➤ With *Trace Signals* chosen in the Trace Signals toolbar, select a signal in any SimVision window and click *Trace Loading Logic*, .

SimVision traces a path one step at a time.

Click the *Trace* button beside the entry, or double-click the entry label, to continue the trace to the next step.

When it reaches a module boundary, you can either trace into the module or skip over the module, as follows:

Click the *Follow into scope* button to descend into the module. When you descend into a module, the Trace Signals sidebar changes the background of the trace path to gray and indents the signals in the path.

Click the *Trace* button beside the entry, or double-click the entry label, to skip over the module.

When you trace a signal, SimVision adds the signal name to the drop-down list in the *Trace* field. You can use the drop-down list to switch from one path to another. Buttons beside the *Trace* field let you manage the contents of the list, as follows:

Creates another instance of the path, and makes that path the current path.

Deletes a path from the list of trace paths.

Adds or removes signals in the containing window so that the window contains only the signals that appear in the current trace. See “Sending Signals to the Containing Window” on page 204 for more information.

SimVision assigns a unique color to each path that you trace, and each object along the path is on a separate line.

Traced signals are also highlighted in the other windows in which they appear. In the Source Browser, they are displayed in green. In other windows, they are underlined. Choose the *View – Clear Trace Highlights* in any window to clear the highlights in all windows.

When you have chosen the path that you want to trace, SimVision extends the connecting line in the *Controls* column, but keeps the other choices visible, so that you can easily choose a different path.

The *Collapse* button in the *Controls* column shows where there are choice along the path. You can click this button to hide these choices and show only the path taken.

Tracing the Driving Logic

 Video

Tracing Drivers in the Trace Signals Sidebar

To trace the driving logic for a signal:

➤ With *Trace Signals* chosen in the Trace Signals toolbar, select a signal in any SimVision window and click *Trace Driving Logic*,  . SimVision traces the signal back through the logic. When it reaches a module boundary, it traces the signal into the module. When it finds more than one driver, it stops to let you choose which driver you want to trace.

When you trace the driving logic for a signal, SimVision assigns a grade to each driver to indicate the relative probability that the item in the list is the active driver. The higher the grade, the more likely it is that the driver is the active driver for the signal. SimVision uses the following methods to determine each driver's grade:

- If the driver has been forced, SimVision displays the force command as a separate line and gives the force a high grade. SimVision still analyzes and grades the other drivers. If you believe that the force is not active, you can use the grades of the other drivers as an indicator of the active driver.
- Otherwise, SimVision analyzes the contributing logic values. If the logic values have contributed to the selected signal, it is most likely the active driver for the signal.
- If it cannot use logic values to determine the driver's grade, or if multiple drivers were found through logic value analysis, SimVision analyzes the timing of the signal. It assumes that the most likely driver is the one whose value has changed most recently.

Although tracing the driving logic is not an exact process, you can increase the accuracy of the trace results in several ways. The following methods will provide increasing levels of accuracy. However, each increase in accuracy comes with a decrease in simulator and database performance.

- Probe the signals of interest. This method has the smallest impact on performance. It is well-suited for structural designs with delay information. You should try this method first, and use the other methods only if it does not produce the result you are looking for.
- Use the `-event` option when creating the simulation database. This method provides sequence time information, which can help you locate timing problems. However, this method generates a larger simulation database.

- Enable statement tracing when you create the simulation database. This method makes it possible for SimVision to trace the signal back to the exact statement that caused the signal transition, but it slows simulation and creates a larger database.

When you trace the driving logic for a signal, SimVision adds a column, labeled %, to indicate the grade. The list is sorted so that the items with the highest grade are first.

The colored bar indicates the driver's grade. It is sized larger for the most likely drivers, and smaller for the less likely drivers. It is also color-coded, as follows:

- Red—SimVision has determined that the simulated value of the driver might have been overidden by a `force` command.
- Blue—SimVision has determined that the driver might be active, based on timing and other considerations.
- Green—SimVision has a high level of confidence that the driver is active.

When you trace drivers, the Trace Signals sidebar often follows a path back through simulation time. When the driving value changed at a previous time, SimVision adds a jagged line and a small red flag in the trace path where the time shift occurred, and it moves the time back. Above the time shift, the time remains unchanged. Below, the trace path displays the values of the signals at the new time.

To see the times above and below the time shift, roll the mouse over the flag. SimVision pops up a tool tip. If SimVision has selected an unwanted simulation time, you can also manually shift the time, as described in "Shifting Time in the Trace Path" on page 204.

Tracing Active Drivers

If you want to trace the most likely path of the active drivers for a signal, you can continuously trace the drivers with the highest probability, or you can let the Trace Signals sidebar do this for you automatically.

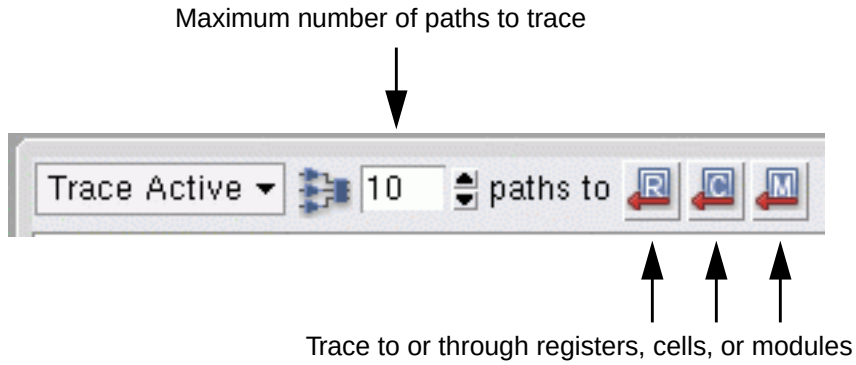
 Video

Tracing Active Drivers in the Trace Signals Sidebar

To automatically trace the most likely active drivers:

1. Choose *Trace Active* from the Trace Signals toolbar. This adds the active driver trace options to the toolbar, as shown in Figure 16-2 on page 202.

Figure 16-2 Active Driver Trace Options



2. Specify the options that you want to use. For example, the options in Figure 16-2 on page 202 specify that SimVision should trace 10 paths through registers, cells, and modules.
3. Click *Trace Driving Logic*, .

Tracing X Values

It is often helpful to trace an X value, either back through the design hierarchy to its origin, or forward through the hierarchy to determine the effect it has on downstream logic. You can trace this X value in the Trace Signals sidebar, by using either the *Trace Signal* buttons or the *Trace X* buttons.

Tracing an X Value with the Trace Signal Buttons

To trace the cause of a signal value, you can use the *Trace Signal* buttons, as follows:

1. Expand the sidebar in the Waveform window and open the Trace Signals sidebar.
2. With the signal selected in the Waveform window, click *Trace Loading Logic*, , or *Trace Driving Logic*, . SimVision follows the signal until it finds multiple paths.

Tip

If there are many contributing signals, it can be difficult to find the contributing signals with X values. In this case, enable *Show only X values along path* to remove from the display all signals except those with X values.

After you have found the signals that have contributed to the X value, you can use other SimVision tools, such as the Source Browser, to locate the cause of this value, or you can display the trace path in the Schematic Tracer or Register window.

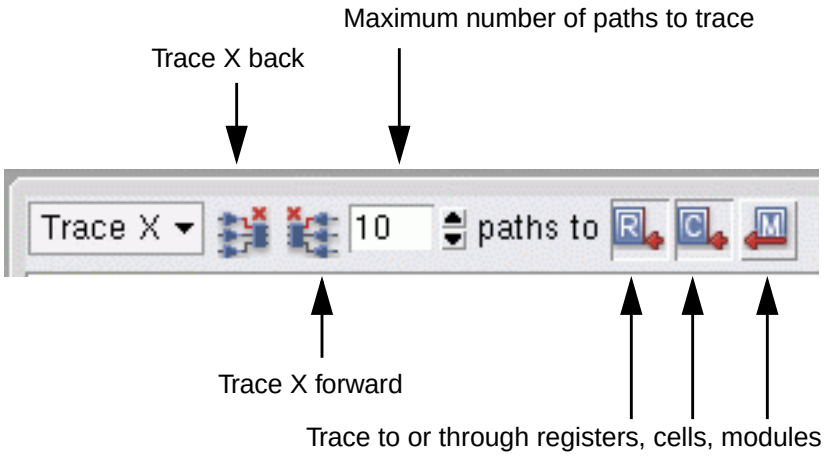
Tracing an X Value with the Trace X Buttons

When you use the *Trace Signal* buttons, tracing stops when SimVision encounters multiple paths. When you use the *Trace X* buttons, SimVision traces all possible paths. That is, it creates one trace path for each possible path. In addition, the trace paths contain only those contributing signals that have X values.

To trace an X value with the *Trace X* buttons:

1. In the containing window, select the signal whose value you want to trace, and open the Trace Signals sidebar.
2. Choose *Trace X* from the Trace Signals toolbar. This adds the X-tracing options to the toolbar, as shown in Figure 16-3 on page 203.

Figure 16-3 X-Tracing Options



3. Specify the X-tracing options that you want to use.
4. Click *Trace-X Backwards,* , or *Trace-X Forwards*, . SimVision creates one trace for each possible path. For each trace path, the pull-down menu indicates the signal name, the trace direction, and the number of the path out of the total number of possible paths.

Shifting Time in the Trace Path

Time shifting lets you examine the values of contributing signals at different times, while still maintaining the time of the signal you are tracing. This can be helpful, particularly with X values, to determine whether an incorrect signal value is caused by a timing problem.

To shift time in a trace path:

➤ Change the time of the primary cursor. There are several ways to do this. For example:

- ❑ Click *Next Edge*,  , in the containing window.
- ❑ Enter a new time in the Time toolbar.
- ❑ In the Waveform window, drag the primary cursor to a new location.
- ❑ Use the Search toolbar to search by *Value*, *Rising Edge*, *Falling Edge*, *Expression*, or *Marker*.

When you change the time of the primary cursor, the Trace Signals sidebar adds a jagged line and a small red flag in the trace path where the time shift occurs. Above the time shift, the time remains unchanged. Below, the trace path displays the values of the signals at the new time. To see the times above and below the time shift, hover the mouse over the flag. SimVision pops up a tool tip.

 *Tip*

You can insert a time shift at any entry in the trace path, and at as many entries as you want.

Sending Signals to the Containing Window

The Trace Signals sidebar lets you send all signals or individual signals in a trace path to the containing window.

To send all of the signals in a trace path to the containing window:

➤ Click *Send Trace to Containing Window*,  . This button adds to the containing window all of the signals that appear in a trace, or removes from the containing window all of the signals that do not appear in the trace.

For example, suppose you have traced a signal in the Design Browser, and you would like to add those signals to a Waveform window. To do this:

1. Open a new Waveform window and display the trace in the Trace Signals sidebar.

2. Click *Send Trace to Containing Window*, . SimVision adds the signals to the Waveform window. These signals are labeled with a comment in the signal list and waveform area of the window, as shown in Figure 16-4 on page 205.

Figure 16-4 Sending a Trace to a New Waveform Window

This feature can also be useful if you have used the Trace Signals sidebar to narrow your field of interest to a few signals, and you want to remove the rest of the signals from the containing window.

For example, you might display a large design in the Schematic Tracer, and use the Trace Signals sidebar to trace a path through one portion of the design. You can then click *Send Trace to Containing Window* to simplify the schematic and display only the portion of the design that relates to the traced path.

To add individual signals in a trace path to the containing window:

➤ Enable *Click to Add*. This button behaves as follows, depending on the window type:

Design Browser	Selects the signal.
Waveform window	Adds the signal to the window. If the signal is already present, it adds a copy of the signal to the window.
Source Browser	Scrolls the source area to where the signal is defined and highlights the signal. Otherwise, it highlights the signal but does not scroll the source area.
Schematic Tracer	Highlights the signal.
Memory Viewer	Adds the signal to the window. If the signal is already present in the window, this button does nothing.
Register	Adds the signal to the window, or selects the signal if it is already present in the window.

Using the Trace Signals Sidebar with the Schematic Tracer

You can use the Schematic Tracer and the Trace Signals sidebar to trace signals in the design. When you trace a path with the Trace Signals sidebar, each path is assigned a unique color, which is also used to display the path in the Schematic Tracer.

You can display multiple paths in the Schematic Tracer. Each is displayed in the color assigned to it by the Trace Signals sidebar.

To display multiple paths:

➤ Enable the check mark in the Trace Signals drop-down menu for each path that you want to display. The Schematic Tracer displays all paths in their assigned colors.

Important

If the selected paths overlap, only one color is displayed for the overlapping area.

To change the color of a trace path:

1. In the Schematic Tracer, display the trace in the Trace Signals sidebar, then click the *Color* button to the right of the signal name. This opens the *Colors* palette.
2. Choose a color from the palette, or click *More* to open the Select a New Color form and mix your own color.

Performing Functions on Objects in the Sidebar

When you select an object, you can right-click to pop up a menu of functions that you can perform on the object, as follows:

Follow (# drivers)	Lets you continue tracing from the selected signal.
Trace Driving Logic	Traces the selected signal back to its module inputs.
Trace Loading Logic	Traces the selected signal forward to its module outputs.
Send to	Lets you add the object to a window. If you choose one of the window types in the menu, SimVision adds the object to the target window of that type. If you choose *Send to new*, you can send the object to a new window of the type you choose.

Break on Change	Creates a breakpoint on the object. See Chapter 9, “Setting and Managing Breakpoints” for more information on setting breakpoints.
Create Force	Opens the Force Value form, so that you can specify the value that you want to give to the object. See “Forcing and Releasing a Signal Value” on page 139.
Release Force	Releases any forces that you have set on the object and returns it to its original value. See “Forcing and Releasing a Signal Value” on page 139.
Deposit Value	Opens the Deposit Value form, so that you can specify the value that you want to deposit. See “Depositing a Signal Value” on page 141.
Create Probe	Opens the Set Probe form, so that you can set a probe on the object. Chapter 8, “Creating and Managing Probes” for more information on setting probes.
Describe	Displays the results of a `describe` command in the simulator Console window. That is, it displays the signal name, type, and current value.

17

Displaying Waveforms

The Waveform window lets you view waveforms that represent the signal transitions during simulation. SimVision lets you view these signal transitions while the simulation is running—this is called “watching live data”—or it can save the transitions to a database that you load into SimVision in post-processing mode.

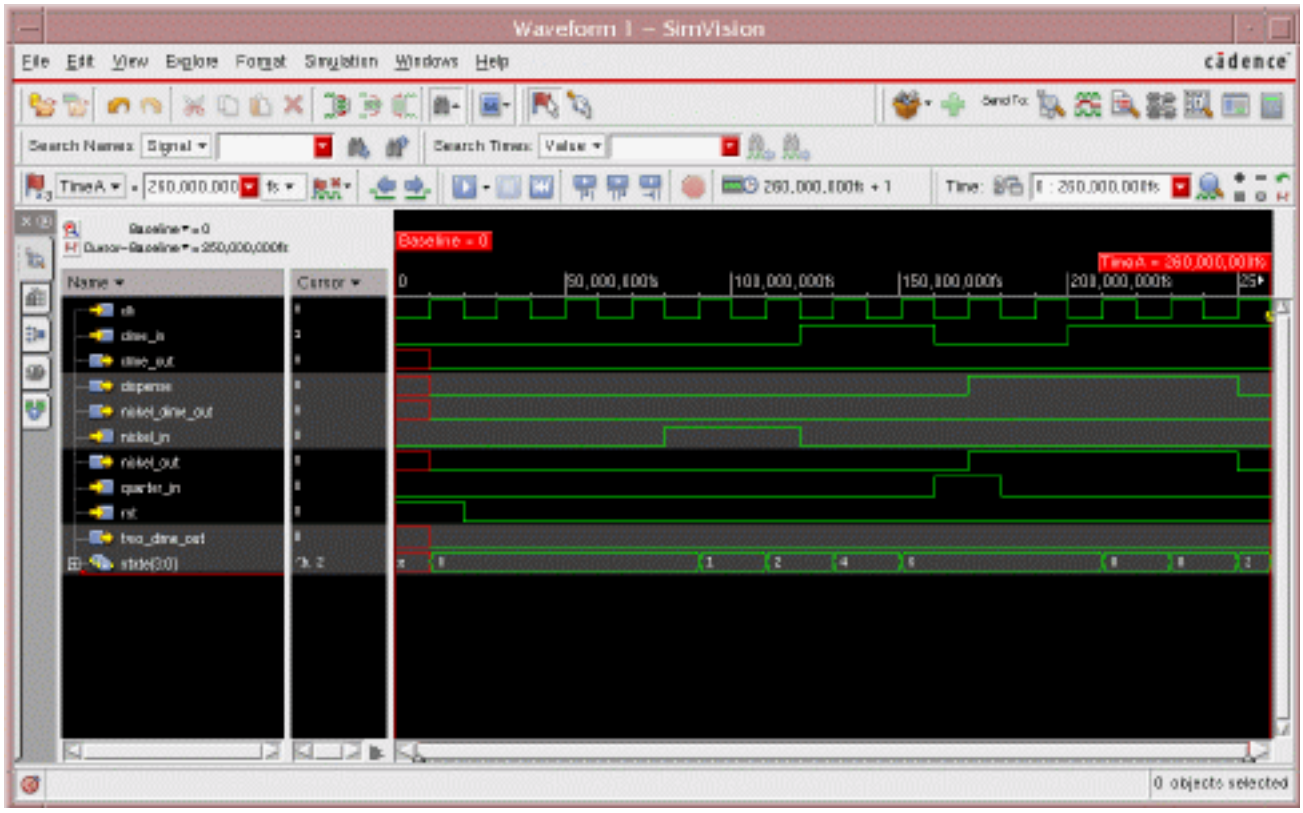
Opening a Waveform Window

To open a Waveform window:

➤ In any SimVision window, click *Waveform*, , in the Send To toolbar, or choose *Windows – New – Wavefrom* from the menu bar.

If you select signals before clicking *Waveform*, , they are displayed in the signal list on the left side of the Waveform window. If simulation data is available, the signal transitions are displayed as waveforms on the right side of the window, as shown in Figure 17-1 on page 210.

Figure 17-1 Waveform Window



The Signal List

The signal list displays the signals and their values, in two independently scrollable columns. The button below the *Value* column indicates whether the column is left-justified or right-justified. Click the button to toggle between the two choices, as follows:

Indicates that the *Value* column is left-justified.

Indicates that the *Value* column is right-justified.

A button sometimes appears below the *Name* column when the hierarchy becomes so deep that the expand and compress buttons (+ and -) reach the right side of the column. You can use this button to scroll the signal names, as follows:

Click on the right or left side of the pyramid to shift the hierarchy to the left or right.

This button does not appear often, but it may if you are displaying the full database and path names for signals in a deep hierarchy.

You can change the size of the signal list and value list to use more or less space in the window.

To resize the signal and value lists:

1. Place the cursor in the bar that separates the signal list from the value list or in the bar that separates the value list from the waveform area. The cursor changes to a double-arrow.
2. Drag the cursor to the right or left. The columns grow wider or thinner as you move the cursor.

The Waveform Area

The waveforms are displayed on the right side of the window. Above the waveforms, a timeline shows the beginning and ending times for the data currently displayed in the window. Below the waveforms, a scroll bar area shows the entire range of simulation time recorded in the database. The scroll bar itself indicates the current view of waveform data displayed in the window.

To scroll the waveform data:

- Drag the scroll bar to the left or right.
- Enable zoom mode,  , then hold and drag the middle mouse button to the left or right.

Maximizing the Waveform Viewing Area

You can maximize the waveform viewing area by enabling and disabling the options in the *View* menu.

For example, to change the window layout so that you can use the maximum amount of space for your waveform display:

1. Disable the *Show Full Signal Names* and *Show Database Names* to show only the simple names of signals in the waveform window.
2. Disable the *View – Sidebar* option or click *Hide Sidebar*,  , in the upper right corner of the sidebar to remove the sidebar from the window.
3. In the *Toolbars* menu, disable any toolbars that you do not need.

4. Click *Show/Hide Search Toolbar*,  , in the standard toolbar to add and remove the search toolbar from the window.

Default Waveform Shapes and Color

The shape of a waveform is determined by the signal type or value, as follows:

 Buses and other aggregate signals

 Events

 Signals with the value `0`

 Signals with the value `1`

 Signals with the value `X`

 Signals with the value `Z`

Icons are sometimes used as special markers, such as when displaying error events.

The color of the waveform is determined by the language in which the source code is written and the signal value. You can change these default colors in the Preferences window, as described in Chapter 25, “Setting Preferences.” These settings apply to all signals in all Waveform windows.

To change the color of an individual signal:

1. Select the signal and choose *Format – Trace Color* to change the color of the trace for that signal in the waveform area, or choose *Format – Name Color* to change the color of the signal name in the signal list.
2. Select a color from the pop-up menu, choose *More* to create your own color, or choose *Default* to restore the trace or the name to its default color.

This setting applies to only the individual signal in the current window. If you send that signal to another window or add it again to the same window, it is displayed in the default color.

Adding Signals to the Waveform Window

There are several ways to add signals to a Waveform window.

- Use the Design Search sidebar. Chapter 5, “Accessing Design Objects” describes how to use the sidebar.
- Use the *Waveform* button:
  - In another SimVision window, select the objects that you want to add to the Waveform window.
  - Use the *Waveform* button, as follows:
    - Click *Waveform*,  , or choose *Send to target* from the button menu to send the objects to the target window.
    - Choose *Send to new* from the button menu to send the objects to a new Waveform window.
- Drag and drop the signals and variables:
  - In another SimVision window, select the objects that you want to add to the Waveform window.
  - Press and hold the middle mouse button, then drag the signals into any area of the Waveform window—signal list, value pane, or waveform area.
  - Release the mouse button to drop the signals into the window.
- Use the *Add* button:
  - In another SimVision window, select the objects that you want to add to the Waveform window.
  - Open a Waveform window , or bring the window to the foreground, then click *Add*, , in that window.

Probing Signals in the Waveform Window

When you add scopes to the Waveform window, SimVision creates a probe for all of the signals in those scopes. The signals are probed to the default database. To probe signals to a different database, use the Set Probe form, as described in Chapter 8, “Creating and Managing Probes.”

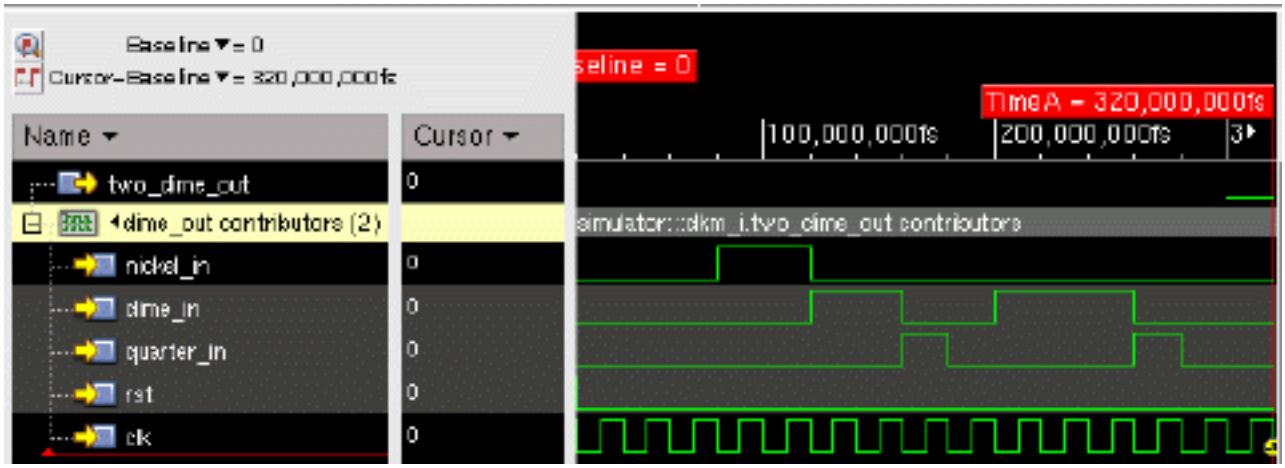
If you have selected a scope and all of its child scopes, only the top-level scope is added to the Waveform window, because recursively adding scopes to the Waveform window could adversely affect performance.

Adding Contributing Signals to the Waveform Window

To add the contributing signals of an object in the Waveform window:

➤ Right-click the object, such as a signal or an assertion, and choose *Add Contributing Signals*. SimVision creates a group that contains all of the contributing signals for the selected object, as shown in Figure 17-2 on page 214.

Figure 17-2 Contributing Signals in the Waveform Window



Displaying Aggregate Signals

When you view aggregate signals in the Waveform window, such as buses, VHDL records, and SystemVerilog packed structures and packed arrays, you can display the entire signal as a single waveform, or expand the signal to see its logic elements or bits. In addition, when you expand a large signal with many elements, the Waveform window lets you create a scrollable region, so that the object takes up less space in the window.

Note: In this release, SystemVerilog queues, dynamic arrays, and associative arrays cannot be displayed in the Waveform window.

Expanding and Collapsing Signals

To expand and collapse signals:

- ➤ Click on the icon next to the signal name. The icon appears as a + sign when the object is collapsed and as a - sign when the object is expanded.
- ➤ Right-click the + or - button and choose *Expand* or *Collapse*.

Some aggregate data types, such as VHDL records or buses, give you different options for expanding objects of those types. For example, you can expand a VHDL record to *Record Elements*, and buses to *Bus Elements*. These choices simply indicate that you can expand these objects to their higher-level elements, rather than the individual bits that make up these objects.

Splitting a Signal

You can split an aggregate signal into groupings in the following widths, or a width that you define.

Bits	Expands to show the individual bits in the signal
Bytes	Expands to show groups of eight bits
Words	Expands to show groups of 16 bits
32-Bit Words	Expands to show groups of 32 bits

Whenever you choose a new width, it becomes the default width for all subsequent expand operations on that signal.

To split a signal:

- ➤ Right-click the + sign next to the signal and choose the width that you want.

To define your own width:

1. Right-click the + button next to the signal, and choose *User-Defined Width*. SimVision opens the Enter Desired Width form.

2. Enter the width and click *OK*. SimVision splits the signal to the desired width, and also adds the new width to the pop-up menu.

Creating a Scrollable Region

Aggregates can be quite large, and you might not want to display an entire object at once. You can make an aggregate scrollable and display only a portion of it at a time.

To make an aggregate scrollable:

➤ Right-click the - button next to the object and choose *Scrollable View* from the pop-up menu. The Waveform viewer displays a portion of the aggregate.

To adjust the size of a scrollable region:

1. Move the cursor over the sizer in the lower left corner of the scrollable region, and the Waveform window displays a box around the region.
2. Drag the sizer up or down to increase or decrease the size of the scrollable region.

To scroll a region:

1. Move the cursor to the left side of the scrollable region. The Waveform window displays a scroll bar.
2. Drag the cursor up and down within the scroll bar. Scrolling speeds up as the cursor moves toward the top or bottom of the scroll bar, and it slows down as the cursor moves closer to the middle of the scroll bar. Click on the arrows at the top and bottom of the scroll bar to scroll one line at a time.

Selecting the Format of Signal Names

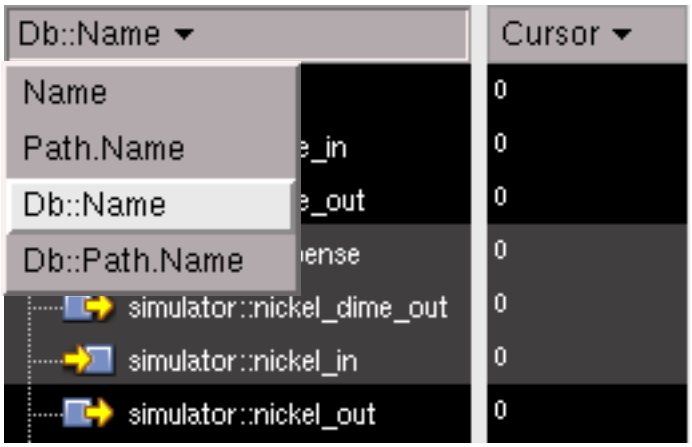
You can choose how you want signal names to appear in the signal list, as follows:

- *Name*—Displays only the signal name
- *Path.Name*—Displays the full hierarchical signal name
- *Db::Name*—Displays the database name and the signal name
- *Db::Path.Name*—Displays the database name and the full hierarchical signal name

To change the format of signal list names:

➤ Select the desired format from the drop-down list above the signal names, as shown in Figure 17-3 on page 217.

Figure 17-3 Choosing a Signal Name Format



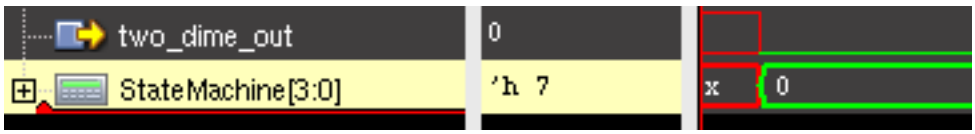
Changing a Signal Name

You can change the name of signal as it appears in the Waveform window. When you change the signal name, it is replaced with a named expression. As a result, its icon in the signal list also changes. You can expand the renamed signal to see the orginal signal.

To change a signal name:

1. Select the signal, then click a second time to open the text field, or choose *Edit Text* from the pop-up menu.
2. Edit the signal name, then press `Return` to close the text field. The icon for the signal changes to the Expression Calculator icon, as shown in Figure 17-5 on page 218.

Figure 17-4 Displaying a Signal Whose Name Has Changed



Note: When you add a structure or array to the Waveform window, the child nodes are shown within the context of that object. Therefore, you cannot change the name of a child node when it is in that context. You must move it out of the structure or array before you can rename it.

Setting the Radix of Signal Values

By default, SimVision displays signal values in the radix in which they are recorded in the database.

To change the radix:

1. Select one or more signals in the Waveform window.
2. Choose *Format – Radix/Mnemonic* from the menu bar and enable the radix in which you want to display the signal value.

You can also display a decimal or hexadecimal value as signed or unsigned.

To display a signal value as signed:

➤ Select the signal, then enable the *Signed* menu choice from the *Format* menu or from the signal value pop-up menu.

To display a signal value as unsigned:

➤ Select the signal, then disable the *Signed* menu choice from the *Format* menu or from the signal value pop-up menu.

Figure 17-5 on page 218 shows a signal, `dir`, displayed twice in the Waveform window. The first is displayed as signed decimal; the second as unsigned decimal.

Figure 17-5 Displaying Signed and Unsigned Values

Note: When you convert a signal between signed and unsigned, you create an expression. This expression is added to the Waveform window in place of the signal. When you add a structure or array to the Waveform window, the child nodes are shown within the context of that object. Therefore, you cannot convert a child node between signed and unsigned when it is in that context. You must move it out of the structure or array before you can convert it.

Using Mnemonic Maps

Mnemonic maps associate signal values with strings, colors, line shapes, and icons. For example, you can define a mnemonic map to associate a name with each state of a state machine, or you can define a mnemonic map to display all `x` values with a red background.

 Video

Creating a Mnemonic Map

Creating a Mnemonic Map

When you create a mnemonic map, you define not only the text that is displayed for a particular signal value, but also the way that value is displayed in the Waveform window, including the shape of the waveform, the color, and icons for any special conditions associated with a value.

To create a mnemonic map:

1. Choose *Edit – Create – Mnemonic Map* from the menu bar. SimVision opens the Mnemonic Maps tab of the Properties window.

   The Mnemonic Maps tab displays the names of all mnemonic maps and the mnemonic map entries for the selected map.

2. Click *New Map* to create a new mnemonic map. By default, mnemonic maps are named `New map`, `New map #2`, `New map #3`, and so on. You can assign a different name to the map by editing the *Name* field.

3. For each entry, provide the following information:

   a. In the *Values Matching* column, choose the default radix from the drop-down list at the beginning of the entry row. When you choose a default radix for the first entry, the same radix is used for all other entries that you add to the map.

   b. Double-click the *Values Matching* field and enter the value that you want to map. Press `Tab` to move to the *Relabel As* field.

   c. In the *Relabel As* field, enter the ASCII string you want to associate with the value. Press `Tab` to move to the *Values Matching* field of the next entry.

   d. The *Preview* field shows what the entry will look like in the Waveform window. You can change its appearance by using the format editing area below the mnemonic map definition, as follows:

Controls the color and shape of the lines drawn around the value.

Controls the color and pattern of the filled area within the lines.

Controls the color and font style of the mnemonic text.

Specifies an icon to display to the left of the mnemonic text.

Tip

You should define all possible values for a signal; undefined values are displayed using the default map.

Applying a Mnemonic Map

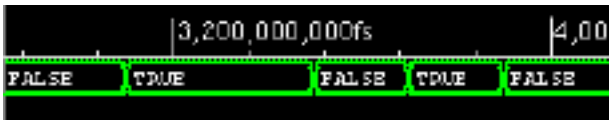
To apply a mnemonic map to a signal, use any of the following methods:

- Select the signal in the Waveform window, then click *Apply To Selected Signals* in the Mnemonic Maps tab.
- Select the mnemonic map name from the list of maps, drag the map into the Waveform window and drop it over a signal. The mnemonic map is applied to that signal.
- Select the signal in the Waveform window and choose the mnemonic map from the *Format – Radix/Mnemonic* menu.

Using a Mnemonic Map to Display Booleans as Logic Signals

Ordinarily, the Waveform window displays VHDL boolean signals as shown in Figure 17-6 on page 220.

Figure 17-6 Displaying a Boolean Signal



However, there may be times when you want to view the boolean as a logic signal, as shown in Figure 17-7 on page 221.

Figure 17-7 Displaying a Boolean as a Logic Signal

To display a boolean as a logic signal:

➤ Select the signal in the Waveform window and choose *Format – Radix/Mmemonic – Boolean as Logic*.

When you use the Boolean as Logic mnemonic map, the values of the signal are displayed as TRUE and FALSE. If you would like to change the way those values are displayed—for example, as `1` and `0`— edit the Boolean as Logic mnemonic map definition in the Mnemonic Maps tab of the Properties window.

Adding Comments to the Waveform Window

You can add comment lines to the Waveform window anywhere in the signal list and waveform area. Comments appear on a single line, and can contain any text that you specify, or can be a single blank line.

There is no limit to the length of a comment. If a comment in the signal list is longer than that area of the window, you can use the scroll bar to see the entire string. If a comment in the waveform area is longer than that area of the window, the text is truncated in the window. However, you can open the text field to see the entire string.

Video

Adding Comments to the Waveform Window

To add a comment:

1. In the signal list, position the insertion bar where you want the comment to appear, or select a signal.
2. Right-click and choose *Create Comment*, or choose *Edit – Create Comment* from the menu bar. A text field opens above the insertion bar or selected signal.
3. Enter the text in the field.
4. If you want the comment to appear in the waveform area, press `Tab`. This opens a text field in the waveform area.

5. Press `Return` to close the text field.

To edit a comment in the signal list:

1. Select the comment, then click a second time to open the text field, or right-click and choose *Edit Text*.
2. Edit the comment, then press `Return` to close the text field.

To edit the comment text in the waveform area:

1. Select the comment, then click a second time to open the text field, or right-click and choose *Edit Text*.
2. Press `Tab` to open the text field in the waveform area.
3. Edit the comment, then press `Return` to close the text field.

To move a comment:

1. Select the comment, then press and hold the middle mouse button.
2. Drag the insertion bar to the new location, and release the mouse button to drop the comment in that location.

To delete a comment:

➤ Select the comment and click *Delete*, , or right-click and choose *Delete*.

Rearranging Objects in the Waveform Window

To rearrange the signals and variables in the Waveform window:

1. In the signal list, select the objects that you want to move.
2. Press and hold the middle mouse button. As you move the cursor up and down the signal list, SimVision displays a horizontal insertion bar.
3. Place the insertion bar where you want the signal to appear, and release the mouse button.

Tip

You can use the insertion bar to determine where additional signals are added to the list. For example, if you want to add a signal in the middle of the list, drag the insertion bar or middle-click at the desired location, then select a signal in any SimVision window, and click *Add*, , in the Waveform window.

Removing Objects from the Waveform Window

To remove objects from the Waveform window:

1. Select the object or objects and click *Delete*,  .

 *Tip*

It can take some time to delete a large object because SimVision is saving *Undo* information. You can improve performance by holding down the `Shift` key while clicking *Delete*. However, when you do this, you cannot undo the delete operation.

Zooming the Waveform Data

There are several ways to zoom the waveform data—with the scroll bar, the mouse, toolbar buttons, the *View* menu, and pop-up menus.

Zooming with the Scroll Bar

- Position your cursor over the pointer at the end of the scroll bar slider, until it turns red.
- Drag the pointer to the left and right. The waveform data zooms in and out to display more or less detail.

Zooming with the Mouse

The Waveform window has two modes for zooming and scrolling the waveform data with the mouse—cursor mode and zoom mode. You select the mode you want by enabling the mode in the Operating Mode toolbar.

In cursor mode, you can zoom between a range of times, as follows:

- `Control`-click and drag the mouse within the waveform area. This highlights a range of times. When you release the mouse, the Waveform Viewer zooms within the selected range.
- Or zoom between a range of times in this way:
  - **a.** Left-click in the waveform area at the beginning of the range. This sets the location of the primary cursor.
  - **b.** Click the middle mouse button at the end of the range. This sets the location of the baseline cursor.
  - **c.** Right-click to zoom between those two times.

In zoom mode, the mouse performs the following zoom and scroll operations:

- Click and drag the mouse horizontally within the waveform area. This highlights a range of times. When you release the mouse button, the Waveform Viewer zooms within the selected range.
- Click and drag the mouse up and to the right to zoom in on the waveform data.
- Click and drag the mouse down and to the left to zoom between the beginning and ending times. This is a zoom fit operation.

Tip

Press the `Control` key to toggle the behavior of the mouse. That is, in cursor mode, hold the `Control` key to perform zoom mode operations. In zoom mode, hold the `Control` key to perform cursor mode operations.

Zooming with Toolbar Buttons and View Menu Choices

➤ To zoom in and out with the zoom toolbar button cluster and *View* menu choices:

Toolbar Button	Menu Choice	Description
	*View – Zoom – In X*	Zoom in incrementally
	*View – Zoom – Out X*	Zoom out incrementally
	*View – Zoom – Previous*	Return to the previous zoom setting
	*View – Zoom – Full X*	Display the waveform data for all simulation times
	*View – Zoom – Full Y*	Display all data along the Y axis (for analog signals)
	*View – Zoom – Cursor-Baseline*	Display the waveform data between the primary cursor and the baseline cursor

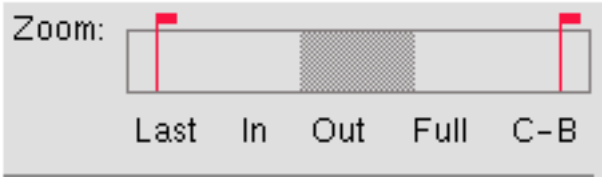
The *Zoom In* function behaves differently, depending on the contents of the window:

- ❑ If the active cursor and baseline are not displayed in the waveform area, the window zooms in toward the center of the waveform data that is currently displayed.
- ❑ If the active cursor is displayed, the window zooms in toward that cursor.
- ❑ If both the active cursor and the baseline are displayed, the window zooms in toward the center of the two cursors.

➤ To zoom in and out with pop-up menus:

Right-click within the waveform area to pop up the menu shown in Figure 17-8 on page 226. The shaded area represents the area currently displayed in the waveform window. As you select one of the choices—*In*, *Out*, *Full*, *C-B*—the pop-up menu displays the corresponding zoom icon. *Zoom Last* returns you to the previous zoom setting.

Figure 17-8 Zoom Command Pop-Up Menu



➤ To zoom between any two markers or cursors:

Right-click a cursor or marker, then choose *Zoom between* and any marker or cursor in the pull-right menu.

➤ To zoom in and out with hotkeys:

Position the mouse inside the waveform area, then press one of the following hotkeys:

Hotkeys	Zoom Command
I	Zoom in
O	Zoom out
=	Zoom out full

Searching the Signal List and Waveform Area

The number of signals and the amount of simulation time displayed in the Waveform window can be large. The Waveform window provides two ways to search for signals and groups in the signal list, and to search for markers, expressions, and values in the waveform area.

To search the signal list, use *Search Names*, as follows:

1. Select *Signal* or *Group* from the *Search Names* menu.
2. In the text field, enter the name or partial name of a signal or group, including the wildcard characters, * and ?. The first occurrence of the string is highlighted as you enter it in the text field.

   The strings you enter are added to a drop-down list, so you can quickly perform the same search again later. Group names are added to the list as you create them. See "Working with Groups" on page 231 for information on creating groups.

Because SimVision keeps a history of the search strings that you enter, you can enter a partial string and press Tab. If the history contains one matching string, SimVision seeds the *Search* field with that string. If the history contains more than one matching string, it displays a list of those strings, and you can choose one.

3. Click *Search Up*, , and *Search Down*, , to find the next or previous object whose name matches the string.

To search the waveform area, use *Search Times*, as follows:

1. Select the type of oject you want to search for from the *Search Times* menu, as follows:
   - ❑ Select *Value* to search for a signal value.
   - ❑ Select *Rising Edges* or *Falling Edges* to search for a specified edge.
   - ❑ Select *Expression* and choose an expression from the drop-down list to search for a specified condition. For information on creating expressions, see "Working with Conditions and Virtual Signals" on page 237.
   - ❑ Select *Marker and* choose a marker from the drop-down list to search for the location of a marker. See "Using Markers" on page 249 for information on creating markers.
   - ❑ Select *Assertion State* and choose a states from the drop-down menu to search for the specified assertion state—`inactive`, `finished`, `failed`, or `disabled`. See *Assertion Checking in Simulation* for information on assertions and assertion states.

   When you choose *Value* or *Marker*, SimVision keeps a history of the search strings that you enter. Therefore, you can enter a partial string and press Tab. If the history contains one matching string, SimVision seeds the field with that string. If the history contains more than one matching string, it displays a list of those strings, and you can choose one.
2. Click *Search Next* and *Search Previous* to move the primary cursor to the next or previous occurrence of the selected object within the waveform area.

Time-Shifting Waveform Data

To shift a waveform:

1. Select the signal from the signal list, then choose *Format – Shift*. This opens the Time Shift form.
2. Enter the amount of time by which you want to shift the waveform.
3. Enable *Relative* to shift the waveform relative to the current location; enable *Absolute* to shift the waveform relative to the original simulated location.
4. Click *OK*. A shifted waveform is indicated in the signal list with the notation:
   `shift(signal_name, total_time)`

Defining and Displaying a Grid

You can display a grid along the X-axis of the waveform data. This can help you see whether signal value changes align as they should. When you first open the Waveform window, the grid is off. You can turn the grid on and off, and define the starting point and interval at which the grid appears.

Turning the Grid On and Off

To turn on the grid:

➤ Enable *Grid* in the Waveform window's *View* menu.

To turn off the grid:

➤ Disable *Grid* in the *View* menu.

The Waveform window can also display a y-axis grid for analog signals. See "Displaying a Y-Axis Grid" on page 276 for more information.

Defining the Grid

By default, grid marks begin at time 0, and they are placed along the X-axis at appropriate time intervals. The size of the interval is determined by how much waveform data you are displaying in the window at the time the grid is enabled. The more data, the larger the interval.

To change the grid settings:

1. Choose *View – Define Grid* from the menu bar. SimVision opens the Grid Definition form.
2. Enter the time at which you want to begin displaying grid marks.
3. Enter the time interval you want to span between the grid marks.
4. Click *OK*.

Saving and Printing Waveforms

To save or print waveforms:

1. Choose *File – Print Window* from the menu bar. The Waveform window opens the Print form.
2. In the *Printer* area of the form, choose *Command* to specify a print command. For example, you can enter the `lpr` command to print the data on the default printer.

   Choose *Print to file* to write the data to a postscript file. You can specify the name of a new file or select an existing file from a file selection form.
3. In the *Print range* area, select the variables and times that you want to print. By default, SimVision prints only those variables and times that are displayed in the Waveform window. However, you can also print selected variables or all variables, and you can specify a range of time, or all times.
4. In the *Comments* area, specify a heading that you want to appear on every page. This heading can include a *Title*, *Designer Name*, *Company Name*, and an additional *Note*.
5. In the *Print paper* area, specify the *Paper size*, *Orientation*, *Colors*, and *Scale* that you want to use. Each of these fields has a drop-down list of values from which you can choose.
6. Click *OK* when you are ready to print.

18

Organizing Signals—Groups, Buses, Conditions, Virtual Signals, and Comparisons

SimVision gives you several ways to organize signals in the Waveform window:

- A group gives a name to a collection of signals, so that you can manipulate the signals as a single entity. A group can also contain other groups.
- A bus concatenates the bits in any number of signals, or a range of bits in an array.
- A condition is defined by an expression that evaluates a set of signals and returns true or false.
- A virtual signal is defined by an expression that produces a value, such as arithmetic or trigonometric functions on a set of signals.
- A comparison combines two or more signals and marks the waveform where their values differ.

Working with Groups

A group gives a name to a collection of signals, so that you can manipulate the signals as a single entity. A group can also contain other groups. There is no limit to the number of groups that can be contained within a group, and the same group can be a member of multiple groups.

Creating a Group in the Waveform Window

To create a group in the Waveform window:

➤ Select the objects that you want to group, and click the *Group* button,  , or choose *Edit – Create – Group*.

By default, SimVision names groups sequentially, `Group 1`, `Group 2`, and so on. To change the name of a group:

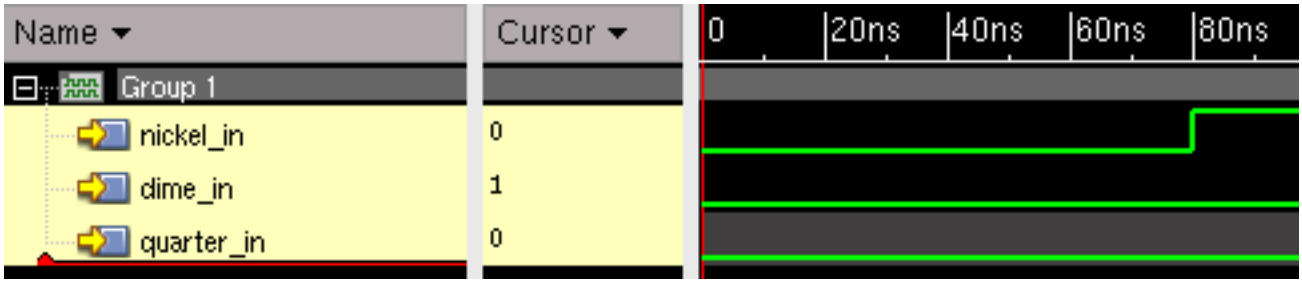
1. Double-click on the group to open the text field, or right-click the group and choose *Edit Text*.
2. Edit the group name, then press `Return` to close the text field.

You can also add a comment in the waveform area on the same line as the group name. To add a comment:

1. Open the text field, then press the `Tab` key. This opens a text field in the waveform area of the window.
2. Enter the text, and press `Return`.

Figure 18-1 on page 232 shows a group made up of three signals: `nickel_in`, `dime_in`, and `quarter_in`. The group is expanded, so that the signals and their waveforms are displayed under the group name, `Group 1`.

Figure 18-1 Creating a Group



Creating a Group in the Properties Window

To create a group in the Properties window:

1. In any SimVision window, select the signals that belong in the group.
2. Choose *Windows – Tools – Groups* from the menu bar. This opens the Groups tab of the Properties window.

   The Groups tab lists any groups that you have already defined and the contents of the currently selected group.
3. Click *Create new group*.  . SimVision creates a new group that contains your selected signals.

4. You can add more signals and groups to the group in any of the following ways:
   - ❑ Drag and drop them into the Groups tab.
   - ❑ Select them in another window and click *Add*, , in the Groups tab.
   - ❑ Copy them in any SimVision window and paste them in the Groups tab.

When you create a group in the Properties window, it does not appear in any Waveform window.

To add a group to a Waveform window:

➤ Select the group in the Properties window and click the *Waveform*, , button.

To change the name of the group in the Properties window:

➤ Select the group, and enter the new name in the *Current Group* field. The name of the group is updated in all windows in which the group appears.

Creating Copies and Instances of Groups

Sometimes you might want to create a new group that contains some or all of the signals in another group. When you create a copy of a group, you can add and remove signals from that group without affecting the original group.

At other times, you might want to create another instance of a group. Any change to an instance of a group affects all instances of that group.

To copy a group:

1. In the Groups tab of the Properties window, select the group and click *Copy*, .
2. Click *Paste*, , and a copy of the group is added to the Groups tab.

The copy of the group contains the same signals as the original group. You can add and remove signals from the copy without changing the contents of the original group.

To create an instance of a group:

1. In the Waveform window, select the group and click *Copy*, .
2. In the same or a different Waveform window, click *Paste*, . An instance of the group is added to that window. It has the same name and contains the same signals as the original group.

If you add or remove signals from an instance of a group, the signals are added or removed from all other instances of that group.

Splitting and Deleting a Group

When you split a group, you remove only the group designation; the signals remain in the window. When you delete a group, you remove the signals from the Waveform window.

To split a group:

➤ In the Waveform window, select the group you want to split, then click *Ungroup*, . The group name is removed from the window, but the signals remain.

To delete a group:

➤ In the Waveform window or in the Groups tab of the Properties window, select the group, then click *Delete*, .

Groups are defined globally; they can appear in multiple windows. Therefore, when you split or delete a group in a Waveform window, it is not deleted from the Properties window or from any other Waveform windows. When you delete a group from the Properties window, however, it is deleted in all Waveform windows in which the group appears.

Performing Other Operations on Groups

You can perform the following operations on groups:

➤ Click the - button next to the group name to collapse the group and make the best use of the available space in the Waveform window; expand it by clicking the + button, to see the individual signals in the group.

➤ Select a group name and drag the group to a new position within the signal list. If you drop the group within another group, it is nested within that group.

➤ Select an individual signal within the group, and drag it to a new position within the group. This rearranges the signals in the group.

➤ Select a signal within the group and drag it outside of the group. This removes the signal from the group.

➤ Select a signal outside of the group and drag it into the group. This adds the signal to the group.

➤ Copy a signal or group and paste it into a group.

When working with analog designs, you can create analog overlay groups. For more information, see "Creating an Overlay Group" on page 273.

Working with Buses

A bus is the concatenation of all the bits in the signal that make up the bus. In the waveform area of the window, the signals that make up the bus are combined into a single waveform. You can expand the bus to see the waveforms for the individual signals.

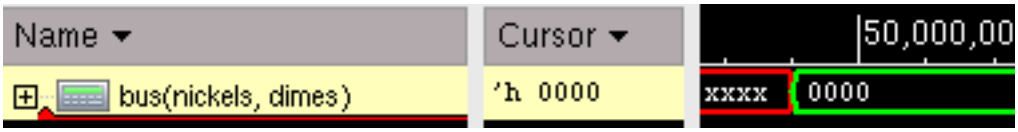
Creating a Bus from Selected Signals

To create a bus:

1. In the Waveform window, select the signals that you want in the bus. You can include individual bits, other buses, or a combination of bits and buses. The order of the bus contents is the order in which the signals are selected.
2. Click *Bus*, ,or choose *Edit – Create – Bus* from the menu bar.

   The bus appears at the location of the insertion bar, as shown in Figure 18-2 on page 235. This figure shows a bus made from two arrays, named `nickels` and `dimes`. Because a bus is the concatenation of all of its signals, all of the signals are combined into one waveform.

Figure 18-2 Creating a Bus



Creating a Bus from an Array Range

If you have an array, you can create a bus from selected bits of the array.

To create a bus from selected bits of an array:

1. Select the array and choose *Edit – Create – Bus from range*. This opens the Create Bus form.

   The default bus name, `bus`, is displayed in the *Name* field. You can edit this field to give the bus a different name.

The name of the array that you selected is displayed in the *Super bus* field.

2. In the *Comma separated bits and/or ranges* field, enter the bits that you want to include in the bus. For individual bits, enter each index into the array, separated by commas. For a range of bits, enter the starting and ending bits in the range, separated by a colon. For example, `2:8`.
3. Click *OK* or *Apply*, and the bus is added to the Waveform window at the location of the insertion bar.

Naming a Bus

By default, a bus is given the default name, `bus`, with the signals that make up the bus enclosed in parenthesis, such as `bus(signal1, signal2)`.

To name a bus:

➤ Double-click the bus in the signal list and enter the new name in the text field.

Buses that you name appear in the Expressions tab of the Properties window, and you can operate on them as described in "Accessing an Expression in the Properties Window" on page 241. Buses with the default name are not displayed in the Properties window.

Splitting a Bus

A bus is made of several operands, and those operands, in turn, can be individual signals or arrays of signals. Therefore, when you split a bus, you have the option of splitting it into its operands or into the bits that make up each operand.

To split a bus into the operands that make up the bus:

➤ Select the bus, then click *Ungroup*,  , or choose *Edit – Ungroup* or *Edit – Split into Operands* from the menu bar.

To split a bus into individual bits:

➤ Choose *Edit – Split into Bits* from the menu bar or pop-up menu.

 *Tip*

If you have time-shifted the bits of a bus, *Ungroup* or the *Edit – Split into Operands* menu choice splits the bus into unshifted bits and the time operand; the *Edit – Split into Bits* menu choice splits the bus into shifted bits.

Moving a Bus

To move a bus:

➤ Select the bus, click the middle mouse button, then drag the bus to a new position within the Waveform window.

Important

You cannot drop a signal into the middle of a bus or drag a signal out of a bus. To add or remove signals, you must edit the bus in the Expression Calculator.

Deleting a Bus

Buses that you name are global objects; if you delete a named bus from the Waveform window, it remains an expression in SimVision. That is, it shows up in the Expressions tab of the Properties window.

To delete a named bus:

➤ Choose *Windows – Tools – Expressions*, select the bus and click *Delete*,  .

To delete an unnamed bus:

➤ Select the bus in the Waveform window and click *Delete*,  .

Working with Conditions and Virtual Signals

A condition is defined by an expression that performs a logical operation on signals; it always evaluates to true or false. It can appear as a digital waveform, which shows where the combination of values occurs during simulation.

A virtual signal is defined by an expression that performs operations on signals, such as arithmetic or trigonometric functions, to produce a new signal value. The virtual signal appears as an analog waveform in the Waveform window, and you can treat it like any other signal.

An expression that you use to create a condition or virtual signal can be named or unnamed. Named and unnamed expressions differ in the following ways:

■ A named expression is a definition of the expression, and you can create as many instances of it in as many Waveform windows as you want. When you change a named

expression, the change affects all instances of the expression. Named expressions also appear in the Expressions tab of the Properties window.

- An unnamed expression is an instance of an expression. You can copy the instance as often as you want in as many Waveform windows as you want, but each instance is unique. That is, if you edit an unnamed expression, you change only that instance; the change does not affect other instances of the expression. Unnamed expressions do not appear in the Properties window.

Important

SimVision cannot display a waveform for an expression unless the signals that make up that expression are probed.

Creating an Expression

To create an expression:

1. Click Expression Calculator,  , to open the Expression Calculator.

   If you have selected signals in the Waveform window, they are displayed in the window. In between each signal, the Expression Calculator inserts operator placeholders for operators and tokens, labeled `Op` and `token`.

2. Edit the *Name* field to give the expression a name, if you want the expression to show up in the Properties window.

3. To simplify the expression definition, enter a default scope in the *Current Scope* field. For example, if you enter `simulator::dkm.top` in the *Current Scope* field, you can write an expression without including the path for the signals, as follows:

   ```
   dispense === ‘b1 && nickel_out === ‘b1
   ```

4. Edit the expression to perform the desired operation. The Expression Calculator provides the following editing functions:

   - Activate a placeholder by clicking on it, or use the `Tab` key to move from one placeholder to the next. Active placeholders are blue.
   - When placeholders are active, you can insert objects in the placeholders by using copy and paste, or by selecting the objects in one window and clicking *Add*,  , in the Expression Calculator. If you select multiple objects, they are inserted into the placeholders from right to left.
   - Position your cursor to the place where you want to insert an operator, and click the desired operator button. There are two tabs of operators, one for digital functions

and one for analog functions. You can also select operators from the *Operations* menu.

If you do not want to see the operator buttons, disable the *Palette* button on the *View* menu.

When you insert an operator, the Expression Calculator inserts placeholders for the arguments to the operator. Hover the cursor over a placeholder for more information about the valid values you can insert in that position in the expression.

- Drag and drop signals, click Add,  , or enter signal names and values for the arguments to operators, or choose another operator to nest a subexpression within an expression. Placeholders are inserted in each subexpression to help you create expressions that are syntactically correct.
- Position the cursor within a subexpression and use the navigation toolbar buttons, as follows:

Select the higher level expression.

Select the first operand in the expression.

Select the next operand in the expression.

The navigation buttons let you select any arbitrary subexpression without requiring you to select the precise range of characters within the expression. They can also help you debug complex expressions without requiring you to count parentheses or memorize operator precedence rules.

- The *Clear* button erases the expression.

5. Click  to apply your edits, or click  to cancel them.

 *Tip*

If you want to create an expression that ANDs the values of a set of signals at the location of the primary cursor:

1. Select the signals that you want to include in the expression.
2. Click the *Next Edge*,  , or *Previous Edge*,  , buttons or move the primary cursor until the signals have a particular value.

3. Choose *Edit – Create – Condition*, or right-click and choose *Create Condition*. SimVision opens the Expression Calculator and creates the expression from those signals and values. For example:

   ```
   dispense == `b01 && dime_out == 'b1.
   ```

4. Edit the *Name* field to give the expression a name, if you want the expression to show up in the Properties window.

5. Click  to save the expression.

Adding an Expression to the Waveform Window

To add an expression to the Waveform window:

- Drag and drop the expression or expression name from the Expression calculator to the Waveform window.
- Select the expression or expression name and click *Waveform*,  . This sends the expression to the target Waveform window, or to a new window if there is no target.

Conditions appear as digital signals. Virtual signals appear as analog signals in the Waveform window. If you want to view a virtual signal as a digital signal, select the expression, then choose *Format – Trace – Digital* from the Waveform window menu bar, or right-click and choose *Trace – Digital*.

Searching for an Expression

You do not need to display an expression as a waveform to search it in the waveform data. Its name appears in the drop-down list above the waveform area in the Waveform window.

To search for an expression:

1. Select *Expression* from the *Search Times* menu, then select the condition or vitual signal from the drop-down list.
2. Click *Next Condition* and *Previous Condition*. The primary cursor moves from one occurrence of the condition to another.

You must display an unnamed expression as a waveform to search for it.

To search for an unnamed expression:

➤ Select the expression in the signal list, then click *Next Edge,*  , and *Previous Edge,*  . The cursor moves from one edge of the expression to another.

If you are connected to a simulation and the primary cursor is at the last recorded simulation time, *Next Edge* advances simulation to the next edge of the expression.

Editing an Expression in the Expression Calculator

Important

You can edit an unnamed expression in the Expression Calculator, but you cannot apply those changes to an instance of the expression.You must drag and drop the expression from the Expression Calculator to the Waveform window and create a new unnamed expression.

To edit an expression in the Expression Calculator:

1. Click Calculator,  , to open the Expression Calculator window.
2. If the expression is not already displayed in the window, choose it from the drop-down list in the *Name* field.
3. Edit the expression to perform the desired operation, as described in “Creating an Expression” on page 238.

Accessing an Expression in the Properties Window

Important

Only named expressions appear in the Expressions tab of the Properties window.

To access an expression in the Properties window:

➤ Choose *Windows – Tools – Expressions* from the menu bar of any SimVision window. This opens the Expressions tab of the Properties window.

To create a new expression:

➤ Click *New*,  , and define the expression, as described in “Editing an Expression in the Expression Calculator” on page 241.

To create a copy of an expression:

1. Select the expression and click *Copy,* .
2. Click *Paste*, . The new expression is named `Copy_of_`*`expression`*, where *`expression`* is the name of the original expression.

To edit an expression:

➤ Select the expression and click *Edit*, . This opens the Expression Calculator window, described in "Editing an Expression in the Expression Calculator" on page 241. You can also edit the expression directly within the Expressions tab of the Properties window.

To delete an expression:

➤ Select the expression and click *Delete*, .

Comparing Waveforms

To perform a simple comparison of waveforms:

1. Select the signals whose waveforms you want to compare, and choose *Edit – Create – Quick Diff*. SimVision adds a new object to the signal list, named `Compare`, followed by the names of the signals you selected, and it adds a waveform that is the combination of the waveforms for the selected signals.
2. Click the + button for the comparison to see the individual signals. SimVision highlights in red the areas of the waveforms that differ, and it displays the values of the individual signals.

To remove the comparison:

➤ Select the comparison and click *Ungroup*, , or choose *Edit – Ungroup* from the menu bar. The individual signals remain in the Waveform window, but the comparison is removed.

To perform complex comparisons on signal transitions, use SimCompare. See the *SimCompare User Guide and Reference* for information on that tool.

19

Managing Time in the Waveform Window

Above the waveform data, the simulation timeline marks off time in increments, by the time units that you have selected. As you work with simulation data, you might want to return to a specific time, mark a range of times, track the current time, or expand sequence time.

SimVision lets you mark points in simulation time, as follows:

- A cursor is a flag that you can position within simulation time, and you can place as many cursors as you want within the waveform data. Every window has one primary cursor, which is used to track the current time and the values of objects at that time, to calculate measurements, and to change your view. You can link primary cursors in several Waveform windows, so that the windows track simulation data together.
- Every window has one baseline, which is used for measurement purposes. In this way, a baseline is like a cursor, but it cannot be linked to cursors or baselines in other windows.
- A marker is a flag that you can position at any point in simulation time. You can place markers anywhere in your waveform data and return to them whenever you want, and you can use them for measurement purposes.
- A time range is a span of simulation time that can be displayed at once in the Waveform window. You can create any number of time ranges.
- If you save event information during simulation, you can expand sequence time to view all events that occur during a time slice.

Using Cursors and the Baseline

A cursor is a flag that you can position at any point in simulation time. A window can have as many cursors as you want, but one cursor in each window is the primary cursor. The primary

cursor is red, and its name and current simulation time appear in the flag and in the text field above the signal list. When a cursor is not the primary cursor, it is gray.

This icon indicates the number of windows in which a cursor is the primary cursor.

Every window also has a baseline. Like the primary cursor, the baseline is red, but you cannot redefine it. The baseline is used in conjunction with the primary cursor for measurement purposes.

Cursors and the baseline appear in three tiers above the waveform area, as shown in Figure 19-1 on page 244. Non-primary cursors are in the top tier. The baseline is in the middle tier. The primary cursor is in the bottom tier.

Figure 19-1 Three Tiers of Cursors

The primary cursor and baseline are used in several ways:

- Setting views—The primary cursor is a focal point for zoom commands. For example, you can set the primary cursor to one point, set the baseline to another point, and zoom the window to display the data between those two points.
- Tracking—As you move the primary cursor, signal values get updated to show their values at the time indicated by the cursor. When you move any of the other cursors, signal values are not updated.
- Measuring—The upper-left corner of the Waveform window shows the cursor time, baseline time, and the distance between them.
- Linking—A cursor can be the primary cursor in more than one window. In this way, you can link Waveform windows together so that they track the same simulation time. When you move the primary cursor in one window, it moves in the other windows to which it is linked.

Synchronizing the Primary Cursor with the Simulation or Database

In the time toolbar, you can synchronize the primary cursor, as follows:

The *Watch Live Data* button indicates whether the primary cursor is synchronized with the simulator or database. When synchronization is on, this button shows a green arrow and the cursor location updates in the Waveform window as simulation time progresses.

The button has a drop-down menu from which you can choose the simulator or database that you want to track when synchronization is on.

When synchronization is off, the *Watch Live Data* button shows a red X and the cursor location does not change. Waveforms get updated periodically, but the view remains stationary.

Important

If you move the primary cursor when *Watch Live Data* is enabled, synchronization is turned off in all windows that share the primary cursor.

To turn synchronization on and off in the Properties window:

1. Place your mouse over the cursor and choose *Properties* from the pop-up menu, or choose *Windows – Tools – Cursors* from the menu bar. This opens the Cursors tab.
2. Click the button in the *Watch Live Data* column.
3. When *Watch Live Data* is enabled and the simulator is writing to multiple databases, you must pick the database or simulator to track during simulation. To associate the primary cursor with a particular live database, double-click on the database in the *Database* column and enter the database name, or select a database from the drop-down list.

You can pick either the primary cursor or the baseline to track the values of signals.

To pick the cursor that tracks signal values:

➤ Select *Cursor* or *Baseline* from the drop-down list above the signal value.

Creating a Cursor

To create a cursor:

➤ In the Waveform window, choose *New Cursor* from the drop-down list beside the *Primary Cursor* field. SimVision creates a new cursor with the default name, `Time`*X*, where *X* is the next available letter in the alphabet.

Changing a Cursor Name

To change the cursor name in the Waveform window:

➤ Double-click on the name in the cursor flag, or right-click the flag and choose *Rename This Cursor*, then enter a new name.

To edit the cursor name in the Properties window:

1. Place your mouse over the cursor and choose *Properties* from the pop-up menu, or choose *Windows – Tools – Cursors* from the menu bar. This opens the Cursors tab.
2. Double-click on the name in the *Cursor Name* column and enter the new name.

Choosing a Primary Cursor

To choose a primary cursor:

➤ Choose the cursor from the drop-down next to the *Primary Cursor* field. The location of that cursor appears in the *Cursor Location* field.

Tip

If the cursor you have chosen is not visible in the waveform area, choose *View – Center on Cursor* from the menu bar.

Setting the Location of a Cursor

To set the location of a cursor in the Waveform window:

➤ For the primary cursor, click within the waveform area at the new location.

➤ For the baseline, click the middle mouse button.

➤ Drag the cursor to the new location. When you drag a cursor beyond the edge of the Waveform window, the window scrolls in that direction.

- Double-click on the time in the the cursor flag, or right-click the flag and choose *Set This Cursor Time*, then enter the time.
- Enter a simulation time in the *Cursor Location* field. Cursor times that you enter are added to the drop-down list, so you can return to that time by selecting it from the list.

  Because SimVision keeps a history of the times that you enter, you can enter a partial time and press Tab. If the history contains one matching time, SimVision seeds the *Cursor Location* field with that time. If the history contains more than one matching time, it displays a list of those times, and you can choose one.

To set the location of a cursor in the Properties window:

1. Place your mouse over the cursor and choose *Properties* from the pop-up menu, or choose *Windows – Tools – Cursors* from the menu bar. This opens the Cursors tab.
2. Double-click on the time in the *Value* column and enter a new time.

Tracking Signal Changes with the Primary Cursor

You can move the primary cursor from edge to edge of one or more signals, as follows:

- Select one or more signals in the Waveform window, then click *Next Edge*,  ,to move to the next rising or falling edge; click *Previous Edge*,  , to move to the previous rising or falling edge of the selected signal or signals. If you have selected multiple signals, the cursor jumps to the next or previous edge of all the selected signals.

When you are connected to a simulation and the primary cursor is at the last recorded simulation time, *Next Edge* can advance simulation to the next change for the selected signals, if you have enabled *Watch Live Data*,  , in the Time toolbar. If you try to advance beyond this point when this button is disabled, SimVision returns the following message:

```
To advance simulation to the next edge of the selected signals, you must enable the
watching of live data from the simulator.
```

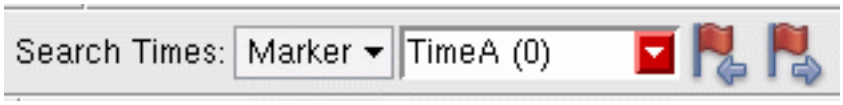
To move the primary cursor from edge to edge of a condition:

1. Select *Expression* from the *Search Times* menu, then select the condition from the drop-down list.
2. Click *Next Condition* and *Previous Condition*. The primary cursor moves from one occurrence of the condition to another.

Finding a Cursor

Sometimes a cursor may be outside of the displayed waveform area. At other times, you might place more than one cursor at the same location. In this case, the cursors are stacked on top of each other, and you might have trouble finding the cursor you are looking for.

To find a cursor that is no longer visible, use any of the following methods:



- Choose *Marker* from the *Search Times* menu, then choose a cursor from the drop-down list of markers and cursors. If the list of cursors and markers is long, and you know the name of the cursor you are looking for, enter its name; or enter a partial name and press Tab. SimVision displays a list of the cursors and markers that match the string, and you can choose one.
- Choose *Marker* from the *Search Times* menu, then click *Next Marker* and *Previous Marker*.
- Choose *View – Center on Cursor* from the menu bar to display the area that contains the primary cursor.
- If the cursor appears in the Delta area above the signal list, click the button beside its name.
- Every cursor appears in the scroll bar below the waveform data as a red line. Position the slider over the red line.

Linking a Waveform Window to a Cursor

You can link a Waveform window to a cursor, so that if the cursor moves in one Waveform window, it will remain visible in all linked windows.

To link a Waveform window to a cursor:

- Choose the cursor name from the drop-down menu attached to the *Link* button in the *Time* field,  .

 *Tip*

If you do not see the *Link* button, choose *View – Toolbars – Customize*, select the *Zoom* toolbar and enable *link*.

Using Markers

Like a cursor, a marker is a flag that you can position at any point in simulation time. Unlike a cursor, the values of the signals are not updated as you move a marker along the simulation timeline, and you cannot use the marker for measurement purposes. However, you can set several markers along the timeline and jump from marker to marker. Markers can also be locked, so that they are fixed at a particular simulation time.

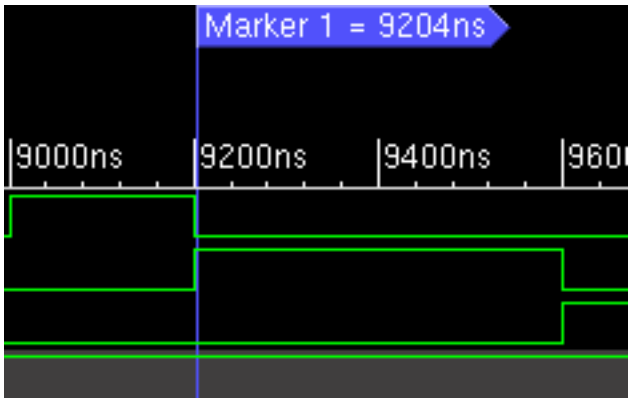
Creating a Marker

To create a marker:

1. Place your mouse in the waveform area where you want the marker to appear, then right-click and choose *Create Marker*, or choose *Edit – Create – Marker* from the menu bar.

   The marker is named `Marker` *`N`* by default, where *`N`* is the next consecutive number. Markers are blue, and their names and locations appear in the flags over the waveform area, as shown in Figure 19-2 on page 249.

Figure 19-2 Creating a Marker



Changing the Name of a Marker

To change the name of a marker in the Waveform window:

➤ Double-click on the name in the marker flag, or right-click the marker and choose *Rename This Marker*, then enter a new name.

To edit the name of a marker in the Properties window:

1. Choose *Windows – Tools – Markers* from any menu bar, or right-click the marker and choose *Properties*. This opens the Markers tab of the Properties window.
2. Double-click in the *Marker Name* column and enter a new name.

Changing the Location of a Marker

To change the location of a marker in the Waveform window:

➤ Double-click on the time in the marker flag, or right-click the marker flag and choose *Set This Marker Time,* then enter a new time value

To change the location of a marker in the Properties window:

1. Choose *Windows – Tools – Markers* from any menu bar, or right-click the marker and choose *Properties*. This opens the Markers tab of the Properties window.
2. Double-click in the *Time* column and enter a simulation time.

Changing the Color of a Marker

To change the color of a marker in the Waveform window:

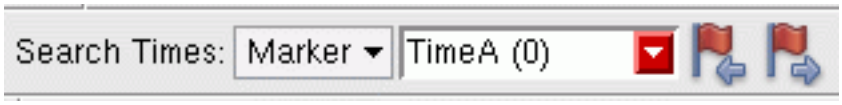
➤ Right-click the marker flag and choose *Marker Color*, then choose a color from the pull-right menu, or choose *More* to create a custom color. You can also change the marker color in the Markers tab of the Properties window.

To change the color of a marker in the Properties window:

1. Choose *Windows – Tools – Markers* from any menu bar, or right-click the marker and choose *Properties*. This opens the Markers tab of the Properties window.
2. Click  and set the color, in one of the following ways:
   - ❑ Choose a color from the menu.
   - ❑ Click *Default* to reset the marker to the default color, blue.
   - ❑ Click *More* to create a custom color.

Finding a Marker

To find a marker in the Waveform window, use any of the following methods:



- Choose *Marker* from the *Search Times* menu, then choose a marker name from the drop-down list to jump to a specific marker.
- Click the *Next Marker* and *Previous Marker* buttons to jump from marker to marker.
- Markers appear in the scroll bar below the waveform as a blue line. Position the slider over the blue line to go to the location of the marker.

Linking a Waveform Window to a Marker

You can link a Waveform window to a marker, so that if the marker moves in one Waveform window, it will remain visible in all linked windows.

To link a Waveform window to a marker:

- Choose the marker name from the drop-down menu attached to the *Link* button in the *Time* field, .

Tip

If you do not see the *Link* button, choose *View – Toolbars – Customize*, select the *Zoom* toolbar and enable *link*.

Locking and Unlocking a Marker

When a marker is unlocked, you can move it to any location in simulation time. When you lock a marker, it is fixed at the locked time and cannot be moved.

To lock and unlock a marker in the Waveform window:

- Right-click the marker flag and choose *Lock Marker*.

To lock and unlock a marker in the Properties window:

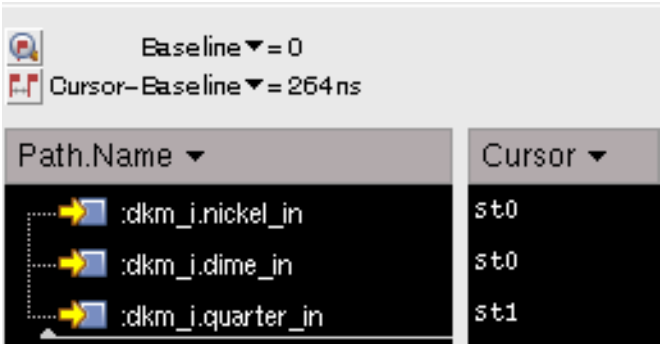
1. Choose *Windows – Tools – Markers* from any menu bar, or right-click the marker flag and choose *Properties*. This opens the Markers tab of the Properties window.

2. Click the lock in the left column of the Markers table. The button changes to indicate when it is locked, , and when it is unlocked, .

Measuring Time in the Cursor Delta Area

Above the signal list is an area in which you can track the location of any cursor or marker, and measure the time between any two cursors or markers. This is called the Cursor Delta area. By default, it shows the location of the baseline and the difference between the primary cursor time and the baseline time, as shown in Figure 19-3 on page 252.

Figure 19-3 Cursor Delta Area of the Waveform Window



You can select any cursor or any two cursors to track in this area. Each measurement appears on a separate line.

Video

Measuring Time in the Cursor Delta Area

To add, change, or delete a line in the Cursor Delta area:

1. Open the drop-down menu for one of the lines:
   - Choose *Cursor* or *Baseline* to track the location of the primary cursor or the baseline.
   - Choose *Delta* to open the Setup Delta form, and select two cursors or markers from the drop-down menus. SimVision uses those two cursors or markers to measure the time between them.
   - Choose a cursor name to track that cursor in the Cursor Delta area.
   - Choose *New Line* to add a cursor or marker measurement to the Cursor Delta area.

- Choose *Delete This Line* to remove a cursor or marker measurement from the Cursor Delta area.

Click the buttons next to a delta line to change the waveform display, as follows:

 Centers the waveform area on the specified cursor or marker.

 Zooms the waveforms between the two cursors or markers.

Using Time Ranges

After you have zoomed in or out to display a particular range of times, you can save that range, so that you can quickly return to it at any time. You can also link other Waveform windows to a time range, so that each linked window tracks the same range. Time ranges appear in a separate tab of the Properties window, where you can create, edit, and delete them.

 Video

Using Time Ranges

Saving a Time Range

To save a time range:

1. Choose *Keep this range* from the *Time* drop-down menu. SimVision opens the Create a Time Range form.
2. If you want, you can change the default name, which is made up of the start and end times separated by a colon, and you can change the start and end times.
3. Click *OK* to create the time range. SimVision adds the range to the *Time* menu, so that you can return to it at any time by choosing it from the menu.

Linking Waveform Windows to a Time Range

You can link two or more Waveform windows, so that they always display the same range of simulation data.

Initially, all windows are unlocked, and each displays its own range of simulation data. The Time toolbar shows a broken *Link* icon.

Tip

If you do not see the *Link* icon, choose *View – Toolbars – Customize*, select the Zoom toolbar and enable *link*.

To link one Waveform window to another:

1. Click the *Lock/Unlock* icon to display a list of Waveform windows. (The list also shows the cursors and markers that you have created. However, you cannot simultaneously link a window to another window and also to a cursor or marker.)
2. Select the window to which you want to link.

When you do this, the Time toolbar in the linked windows show an unbroken *Link* icon.

Now, when you change the range of simulation data displayed in one window, the range also changes in the other windows.

Managing Time Ranges in the Properties Window

Time ranges that you save in the *Time* menu are added to the Time Ranges tab of the Properties window. You can create, edit, and delete ranges in that window.

To open the Time Ranges tab of the Properties window:

➤ Click *Properties*,  , in the toolbar and choose *Time Ranges* from the list of tabs, or choose *Windows – Tools – Time Ranges* from the main menu, or *Edit time ranges* from the *Time* menu.

To create a time range:

➤ Click *New Time Range*,  . SimVision adds a range to the list with the default name `0:0`, and start and end times set to `0`. You can edit any of these values.

To edit a range:

➤ Click on the value that you want to change, then enter the new value.

To delete a range:

➤ Select the range that you want to delete, then click *Delete*, .

Viewing Events in Sequence Time

Ordinarily, the simulation timeline is collapsed so that it appears as if all events in a single clock cycle occur simultaneously. However, this is not how the simulator operates. It lumps together events that occur during the same clock cycle into a single time slice, and it executes those events one at a time. You see only the final state of the signals and variables at the end of a time slice.

If you save event information during simulation, you can expand sequence time and see the order of events that occur during each time slice. Viewing sequence time is useful when you want to determine whether a signal or variable is assigned a value more than once during the same time slice, or when you need to debug a race condition.

Important

Expanding sequence time shows you the order of events, not the delta cycles in which the events occur. For example, if two signals change within the same delta cycle, they are numbered 1 and 2, in the order in which they changed. These numbers do not indicate the delta cycle in which the events occurred. Viewing events in delta time can also be helpful, especially when you are debugging VHDL designs. For information on delta cycles, see Chapter 12, “Debugging at the Delta Cycle Level.”

Video

Viewing Events in Sequence Time

You can view the events that occur in sequence time only if you first save event information in the database during simulation, as follows:

1. Compile and elaborate your design, as you ordinarily would, and invoke SimVision.
2. Choose *File – Create Database* from any SimVision window.

3. In the Open Database form, enter the database logical name and database filename, and then enable the *record all events* button. Enabling this button indicates that you want to include event information in the simulation database.
4. Set your probes and run the simulation to create the simulation database.

To view sequence time in the Waveform window:

1. Populate a Waveform window with the signals you want to observe.

   Initially, sequence time is collapsed, and you see all events as if they occurred simultaneously. However, if a signal or variable has multiple values assigned to it during the same time slice, SimVision displays a yellow dot at the time where the multiple assignments occur.

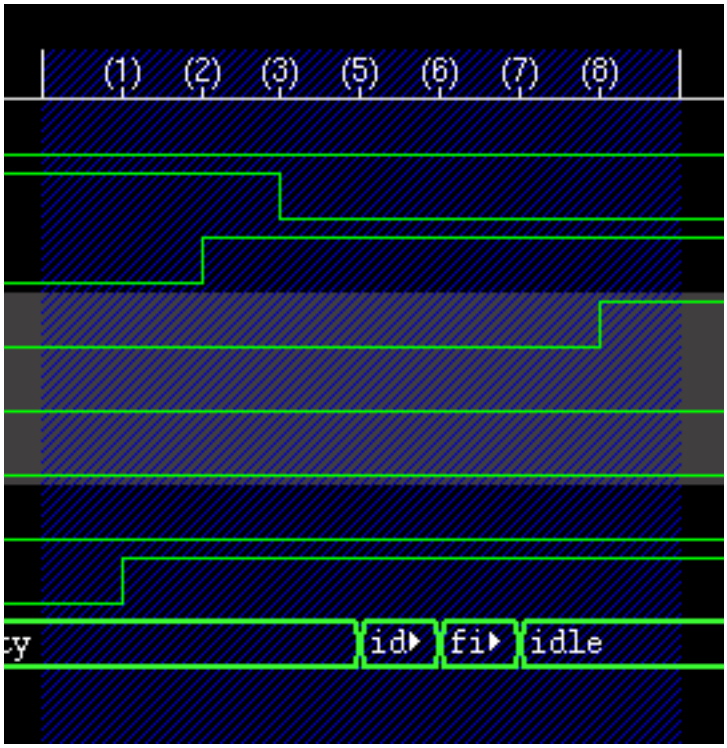
   For example, in Figure 19-4 on page 256, you can see a dot where a variable changes from `fifty` to `idle`. The dot indicates that the variable has multiple values during that time slice.

Figure 19-4 Locating Multi-Value Variables in the Waveform Window

2. Place the primary cursor over the yellow dot, and choose *Expand Time Sequence – At Time* from the pop-up menu.

   The expanded area is colored blue, and numbers are added to the timeline inside the area. These numbers indicate the order in which the signals are assigned values.

Figure 19-5 Displaying Sequence Time



Notice that the events are numbered `(1)`, `(2)`, `(3)`, `(5)`, `(6)`, `(7)`, and `(8)`. This indicates that the signal transition associated with event `(4)` is not displayed in the Waveform window. You can also see that the variable with the yellow dot is assigned a value three times: `idle` at event `(5)`, `five` at event `(6)`, and `idle` at event `(7)`.

3. Pop up the menu over the waveform area of the window and choose *Collapse Sequence Time – All Times*. The sequence time information is removed from the waveform.

To expand sequence time for a selected area of the waveform display:

1. Right-click over the waveform at the beginning of the simulation time that you want to expand. This places the primary cursor at that time.
2. Click the middle mouse button over the end of the simulation time that you want to expand. This places the baseline at that time.
3. Pop-up the menu over the waveform area and choose *Expand Sequence Time – Cursor-Baseline*.

You can also expand sequence time for the entire simulation, by choosing *Expand Sequence Time – All Time*.

20

Debugging Memories

The Memory Viewer lets you observe changes in the internal state of a memory, such as Verilog arrays, and VHDL and SystemC memory constructs.

Important

To view SystemC memories, you need to create a derived class and probe the memories with the `probe` command. For more information, see Chapter 9, “Using the Memory Viewer,” in the *NC-SC Simulator Reference*.

In addition, Memory Viewer support for SystemC has the following limitations:

- You cannot stop on a breakpoint in the memory module, or deposit new values to the memory module.
- Probing memory to the screen (`probe -screen`) is not supported.
- You must instantiate debuggable memory within SystemC; it may not be instantiated directly in HDL.
- The right-click pop-up menu command *Send to Schematic Tracer* is not supported for SystemC memories.
- The menu commands *Edit – Modify*, *File – Save Memory*, and *File – Load Memory* are not supported for SystemC memories.

The Memory Viewer can display the value of each memory cell during simulation. During simulation, you can set breakpoints, and force and deposit values to memory cells. When you probe the memory, the simulator writes these values to a simulation database. Later, you can run SimVision in post-processing mode and view the memory values in the database, but you cannot set breakpoints, or deposit or force values to the memories.

Note: You must probe the memory if you want to be able to view its contents at times prior to the current simulation time. That is, if you move the primary cursor back in time, no values are available to display unless the memory has been probed.

Video

Debugging Memories

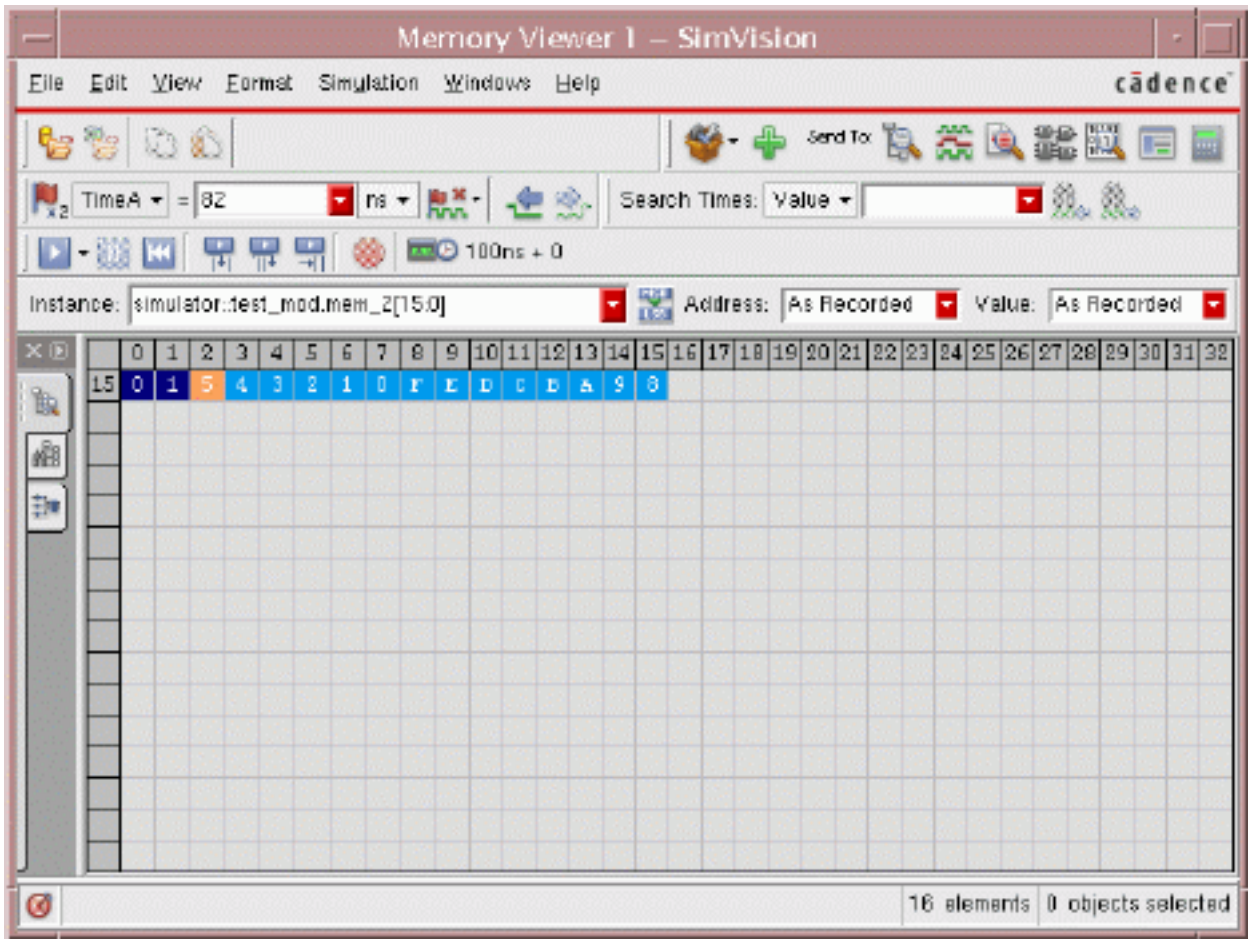
Opening a Memory Viewer Window

To open a Memory Viewer window:

➤ In the Send To toolbar of any SimVision window, click *Memory Viewer*,  , or choose *Windows – New – Memory Viewer* from the menu bar.

If you select a memory before opening the window, the memory is displayed in the window, as shown in Figure 20-1 on page 261. If you do not select a memory first, the window is empty.

Figure 20-1 Memory Viewer Window



In this figure, the Simulation Control toolbar is visible because the memory is being viewed during simulation. When you run in post-processing mode, the Simulation Control toolbar is not visible.

Adding Memories to the Window

There are several ways to add a memory to the window:

- Expand the sidebar to select the objects you want to add to the window. See Chapter 5, “Accessing Design Objects” for information on using the sidebar.
- Drag them from another SimVision window and drop them into the window.
- Select the objects in another window and click *Add*, , in the Memory Viewer window.

➤ Click *Copy*, , to copy selected objects in one window, then click *Paste*, , in the Memory Viewer window.

The Memory Viewer displays only one memory at a time. However, SimVision keeps a history of the memories that you view. Therefore, you can enter a partial memory name and press Tab. If the history contains one matching instance, SimVision seeds the *Instance* field with that instance. If the history contains more than one matching instance, it displays a list of those instances, and you can choose one.

To see more than one memory at the same time, open several Memory Viewer windows and view one in each window.

Color Coding Memory Value Changes

The Memory Viewer color codes the memory cells to show how their values change over time, as follows:


Black on a white background indicates that the value is stable during the current sampled time.


White on a gold background indicates that the value is changing in the current sampled time.


White on a light blue background indicates that the value changed during the current sample time, but its value is the same as its starting value.


White on a dark blue background indicates that the value changed during the current sample time, and its ending value is different from its starting value.

You can change these colors in the *Memory Viewer Colors* tab of the Preferences window. For more information, see “Memory Viewer Colors” on page 323.

Viewing a Range of Memory Cells

To view a range of memory cells:

➤ Expand the memory in a Design Browser or Waveform window, and select the range that you want to view, then add the range to the Memory Viewer window.

➤ Edit the *Instance* field. For example, if the memory module has the range 0:1023, and you want to view the values between 0 to 511, you edit the subrange in the *Instance* field. For example:

```
my_mem[0:511]
```

The range is added to the *Instance* menu. Because SimVision keeps a history of the ranges that you enter, you can enter a partial range and press Tab. If the history contains one matching range, SimVision seeds the *Instance* field with that range. If the history contains more than one matching range, it displays a list of those ranges, and you can choose one.

Changing the Order of Memory Addresses

To change the order in which memory addresses are displayed, toggle the *LSB/MSB* button, as follows:

Displays cells from most significant to least significant bit.

Displays cells from the least significant to the most significant bit.

Changing the Radix of Addresses and Cell Values

To change the radix of memory addresses:

➤ Choose a radix from the *Address* drop-down menu.

To change the radix of cell values:

➤ Choose a radix from the *Value* drop-down menu.

Changing the Number of Columns

To change the number of columns:

➤ Choose *View – Size to Fit*, and the Memory Viewer adjusts the number of columns according to the size of the window.

If you make the window wider, the Memory Viewer adds more columns; if you make the window narrower, it removes columns. If the size of the window is narrower than the number of columns in your memory, the Memory Viewer wraps each row, so that you can see all of the memory cells.

➤ Choose *View – Number of Columns*, and specify the number of columns you want to display in the window. The Memory Viewer displays the fixed number of columns that you have specified.

If you make the window wider, the Memory Viewer increases the size of the columns. If you make it narrower, it decreases the size. If you make the window too narrow for the memory, the Memory Viewer adds a scroll bar, and you can scroll to see the memory cells that are beyond the display area.

Searching for a Value in Memory

To search for a value in memory:

1. Choose *Edit – Search* from the Memory Viewer menu bar. This opens the Text Search form.
2. Enter the value in the *Search for* field, then click *Up* or *Down* to indicate the direction of the search.

Tip

Hover the mouse over the *Search for* field to see examples of valid search constructs.

3. Click *Search* to highlight occurrences of the string one at a time; click *Find All* to highlight all occurrences of the string.

Going to a Memory Cell

To go to a memory cell:

1. Choose *Edit – Go To Address* from the Memory Viewer menu bar. This opens the Go To Address form.
2. Enter the address and click *OK*. The Memory Viewer highlights the address in red.

Viewing a Memory during Simulation

During simulation, there are additional functions that you can perform on memories:

■ Probing Memories

- Setting Breakpoints
- Setting the Value of a Memory Cell
- Saving and Restoring a Memory State

Probing Memories

When you probe a memory, its value changes are written to a database during simulation. You can later load this database into SimVision and view the memory in post-processing mode.

Memories are automatically probed when you send them to the Waveform window, and you can probe a memory from the Design Browser window or sidebar. See Chapter 8, “Creating and Managing Probes” for information on creating probes.

Setting Breakpoints

To create an object breakpoint on a memory cell or range:

➤ Select a memory cell or range, then click *Breakpoint*,  , in the Memory Viewer toolbar or select *Break on Change* from the pop-up menu.

To set a breakpoint on a memory cell based on its value:

➤ Select the memory cell and choose *Simulation –Set Breakpoint – Condition* from the menu bar. See Setting a Condition Breakpoint on page 135 for more information.

The breakpoints that you set appear in the *Breakpoints* tab of the Properties window.

To open the *Breakpoints* tab:

➤ Choose *Simulation – Show – Breakpoints* from the menu bar in any SimVision window.

Setting the Value of a Memory Cell

You can set the value of a memory cell with either a `force` or a `deposit` command.

To set the value of a memory cell:

1. Choose *Edit – Modify* from the menu bar. The Memory Viewer opens the Modify Memory form.

2. In the *Address Range* area, you can choose to modify all addresses or a range of addresses.
3. In the *Fill Data* area, specify the way you want to modify the memory, as follows:
   - Enter a value in the *Value* field.
   - Enable *Increment* and specify the amount by which you want to increment the selected addresses.
   - Enable *Decrement* and specify the amount by which you want to decrement the memory addresses.
   - Enable *Random* to change the seed value, so that you can generate new memory values.
   - Enable *Load from File* to load values into memory from a file. Enter a filename, or click *Browse* to locate and select the file that you want to load. See "Saving and Restoring a Memory State" on page 266 for information on how to create a file that contains memory values.
4. In the *Type* area, choose whether you want to set the memory value with a `deposit` or a `force` command.

For more information on forcing and depositing values, see Chapter 10, "Changing and Monitoring the Value of an Object during Simulation."

Saving and Restoring a Memory State

The Memory Viewer lets you save and restore the contents of a memory at a particular simulation time.

Note: You cannot save and restore SystemC memories.

The memory contents are stored in a text file, which you can later load into the memory during another simulation run. This can help you to restore a memory to a predetermined state during a debugging session.

The memory is saved in a memory image file using the simulator's `memory` command. The file contains directives for specifying the address and data formats, and for specifying a default value for unspecified addresses. The memory and its values are written in *`address/data`* format.

For example, the following file contains the contents of a saved memory. The memory is represented in two columns. The first column shows the address of the memory cell, and the

second column shows the value of the cell. The memory in this example is 16 bits, and the values are in hexadecimal:

```
# Memory Dump.
# Version - TOOL:       ncsim   05.83-d002
# Memory Object - test_mod.l1.mem_2

$ADDRESSFMT H
$DATAFMT h

f/7
e/6
d/5
c/4
b/3
a/2
9/1
8/0
7/f
6/e
5/d
4/c
3/b
2/e
1/f
0/0

# End of Memory Dump
```

For more information on the format of a memory image file, see the documentation of the `memory` command in your simulator *Help*.

To save a memory:

1. Choose the memory that you want to save from the Memory Viewer's *Instance* drop-down menu.
2. Choose *File – Save Memory* from the Memory Viewer menu bar. This opens the Save Memory form.
3. In the *Address Range* area, specify whether you want to save all addresses or a range of addresses within the memory.
4. In the *Output Format* area, choose the radix in which you want to save the memory addresses and values.
5. In the *Save File* field, enter the name for the file. By default, the file has the same name as the memory element, plus the `.txt` file extension. If you want to overwrite an existing memory file, click *Browse* to locate and select the file.

To restore a memory state:

1. Choose *File – Load Memory* from the Memory Viewer menu bar. This opens the Load Memory form.

2. In the *Address Range* area, specify whether you want to save all addresses or a range of addresses within the memory.
3. In the *Load File* field, enter the name for the memory image file. By default, the file has the same name as the memory element, plus the `.txt` file extension. If you want to load a different file, enter the filename in the text field or click *Browse* to locate and select the file.

Printing a Memory

To print the Memory Viewer window:

1. Choose *File – Print Window* from the Memory Viewer menu bar. This opens the Print form.
2. In the *Printer* area, choose *Command* to print the memory, then enter the appropriate print command. Choose *Print to file* to save the memory as a postscript file, then enter a filename or click Browse (`...`) to choose a file. You do not need to enter a file extension; the file is given the `.ps` extension by default.
3. In the *Print range* area, choose to print all memory values or only the values that are visible in the window.
4. In the *Comments* area, enter a title and any other information that you want to appear on the top of the page.
5. In the *Print paper* area, choose your paper size, orientation, and color options. Each option has a drop-down list of choices.

   If you click *Fit to page*, the memory is enlarged or reduced to fit within the margins of the selected paper size.

21

Viewing Analog Data

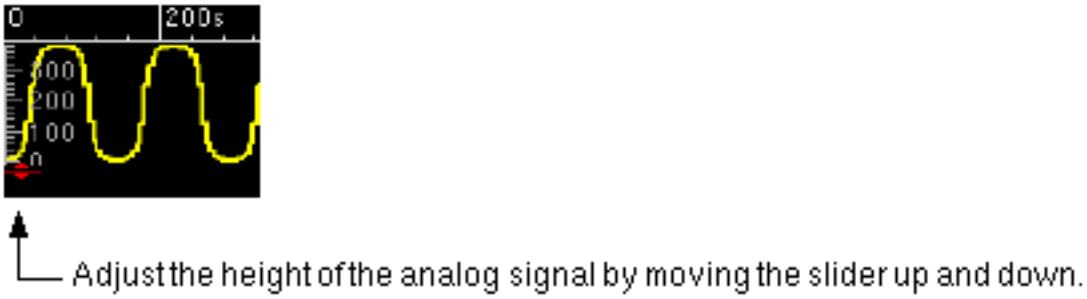
When you open a database that contains analog data, the analog signals appear in the Design Browser and Design Search sidebars. You add analog signals to the Waveform window in the same way that you add digital signals. However, you have more options for viewing analog signals. These additional options are added to the menu bar, pop-up menus, and waveform display when you work with analog designs.

For information on simulating an analog design, see the *Virtuoso AMS Designer Simulator User Guide*.

Displaying Analog Signals in the Waveform Window

To add an analog signal to the Waveform window, select them in the Design Browser or Design Search sidebar, as described in Chapter 5, “Accessing Design Objects.”

When you add an analog signal to the Waveform window, the waveform area contains Y-axis markers, in addition to the X-axis markers displayed for digital signals. The Y-axis is added to the left side of the waveform area within the region occupied by each analog signal.



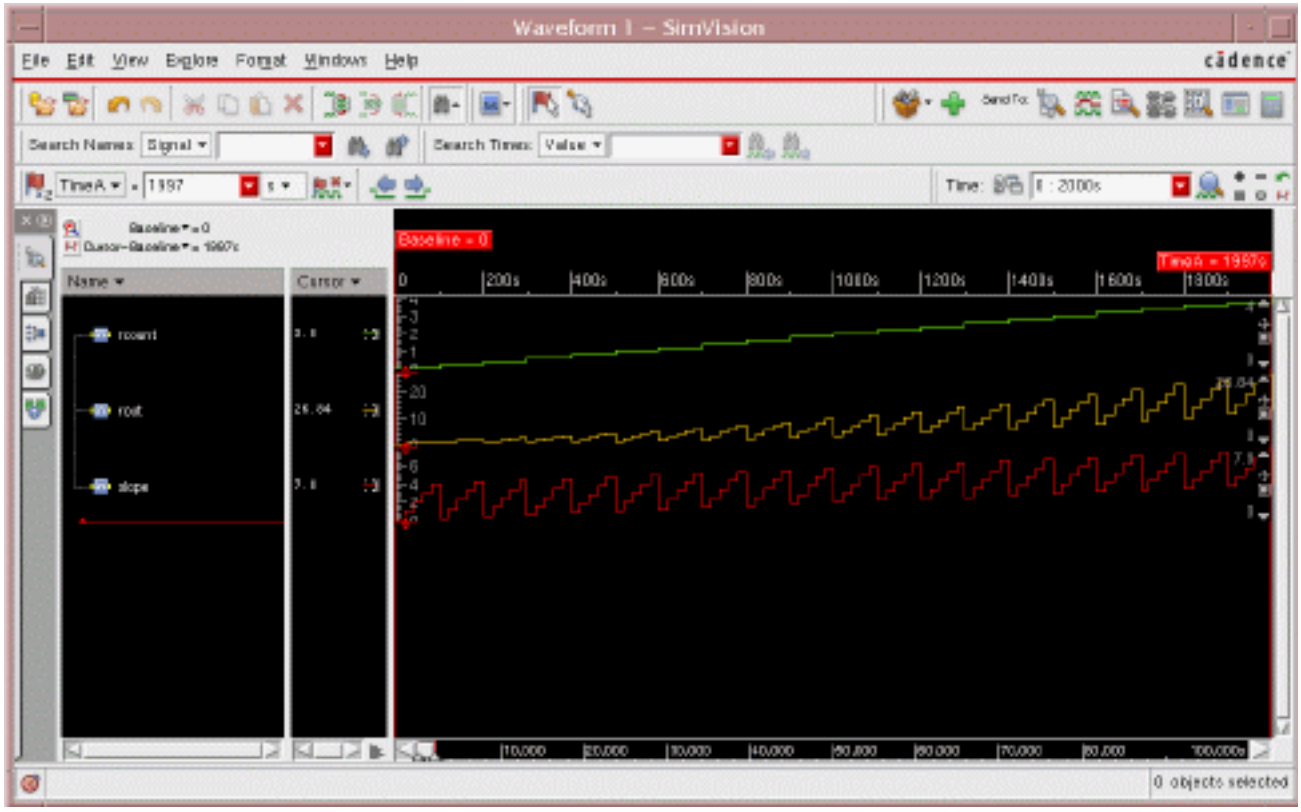
Tip

If it takes a long time to display the analog waveform, you might have a large number of data points occurring at the same simulation time. To reduce the number of data points used for drawing the waveform, enable the *Use data sampling* preference in the Analog Waveforms tab of the Preferences window. For more information, see "Analog Waveform" on page 315.

By default, each analog signal that you add to the window is assigned a unique color. You can change this default behavior in the Preferences window, as described in "Analog Waveform" on page 315.

Figure 21-1 on page 270 shows a Waveform window with three analog signals.

Figure 21-1 Displaying Analog Signals in the Waveform Window



A row of navigation buttons is added to the right side of the window within each analog signal area.

Tip

If the trace area is small, you might not see all of the navigation buttons. Drag the red slider to increase the height until you can access the buttons.

When you position the cursor over one of these buttons, it is highlighted and becomes active. The navigation buttons let you move around within an analog signal, as follows:


Click or drag to adjust the upper limit.


Click and drag to pan in all directions within the signal area.


Click to zoom out and fill the vertical area.


Click to zoom in on the Y-axis by increments.


Click to zoom out on the Y-axis by increments.

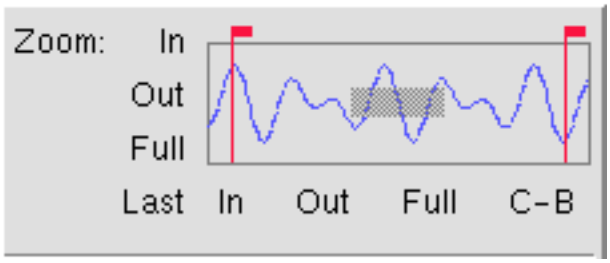

Click or drag to adjust the lower limit.

In addition to the Y-axis zoom commands, the X-axis commands are available for analog signals, as follows:

Toolbar Button	Menu Choice	Description
	*View – Zoom – In X*	Zoom in incrementally
	*View – Zoom – Out X*	Zoom out incrementally
	*View – Zoom – Previous*	Return to the previous zoom setting
	*View – Zoom – Full X*	Display the waveform data for all simulation times
	*View – Zoom – Full Y*	Display all data along the Y axis (for analog signals)
	*View – Zoom – Cursor-Baseline*	Display the waveform data between the primary cursor and the baseline cursor

These commands also appear on the pop-up menu over the waveform area, as shown in Figure 21-2 on page 272. The shaded area in the pop-up menu shows the area that is currently displayed in the waveform window.

Figure 21-2 Analog Zoom Command Pop-Up Menu



Zoom commands that affect the X-axis are displayed along the bottom of the menu; Y-axis zoom commands are displayed on the left. As you hold your cursor over a command, its corresponding icon is displayed in the menu.

Displaying Real Values with Full Precision

When you hover the mouse cursor over a real value in the signal list of the Waveform window, SimVision pops up a tooltip that shows the real value with full precision. However, by default, the value displayed in the signal list is limited to five decimal places.

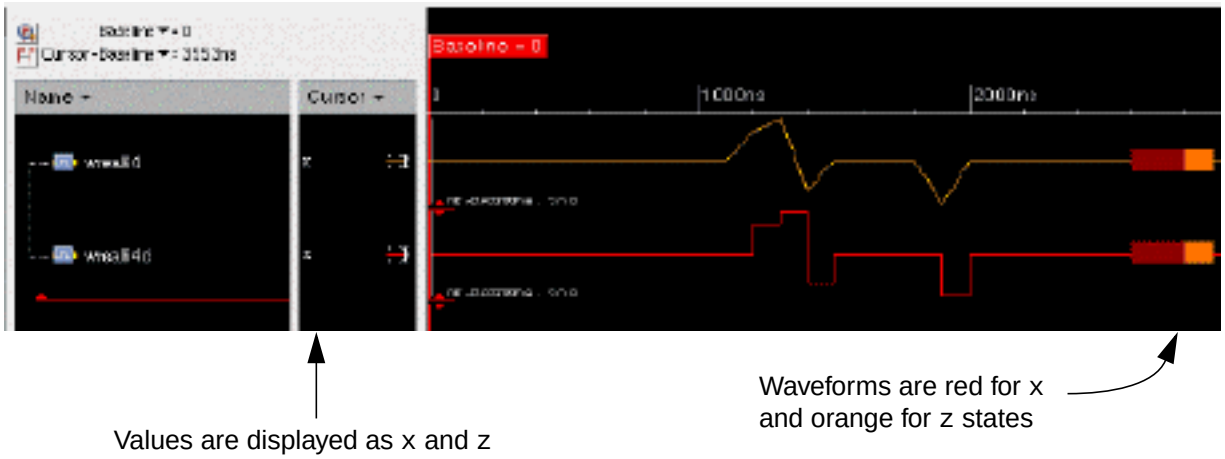
To display real values with full precision in the signal list:

➤ Choose *Edit – Preferences*, open *Signal Options*, and enable *Display Real values with full precision*.

Displaying wrealXState and wrealZState Values

When a `wreal` signal has an `X` or `Z` value, the signal values are displayed as `X` and `Z` in the values pane. The waveforms are displayed as rectangles. `X` values are colored red; `Z` values are colored orange, as shown in Figure 21-3 on page 273.

Figure 21-3 wrealXState and wrealZState Values in the Waveform Window



Creating an Overlay Group

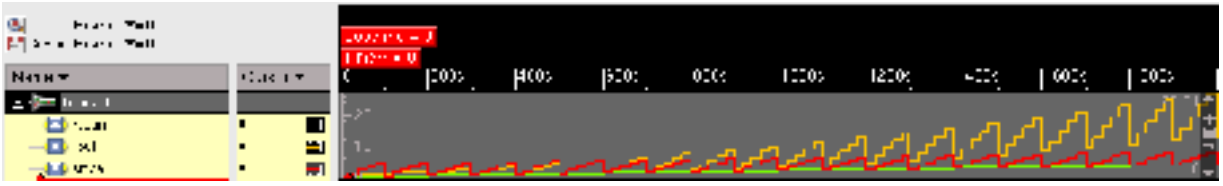
An overlay group is a group of signals that are displayed together within the same signal area. Overlay groups can help you visualize the relationships between signals during simulation.

To create an overlay group:

➤ Select the signals that you want to combine, then choose *Edit – Create – Analog Overlay* from the menu bar or right-click and choose *Create Analog Overlay*. The overlay group is named `Group` *`N`*, where *`N`* is a sequential number.

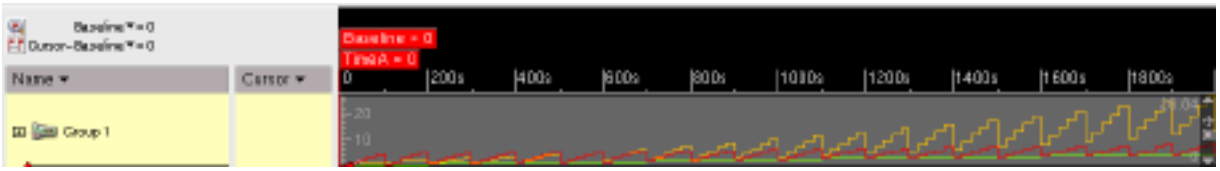
Figure 21-4 on page 273 shows an overlay group. The waveform shows the combined waveforms of the signals in the group.

Figure 21-4 Expanded Overlay Group



Initially, the group is expanded to show the individual names of the signals in the group. You can click the - button next to the group name to collapse the group. When the group is collapsed, you see the combined waveforms next to the group name, as shown in Figure 21-5 on page 274.

Figure 21-5 Collapsed Overlay Group



Tip

In the Groups tab of the Properties window, you can make any group an overlay group by enabling the *Analog Overlay* button next to the *Current Group* field. This adds the combined waveform to the Waveform window. When you disable the *Analog Overlay* button, SimVision splits the group into its individual waveforms.

Creating a Bus with Analog Signals

When you create a bus that contains analog signals, SimVision converts the analog signals to digital signals. You must specify the threshold at which the analog signal is converted to a 0 or 1.

To create a bus with analog signals:

1. Select the signals that you want to include in the bus, and choose *Edit – Create – Bus* from the menu bar of the Waveform window. SimVision opens the Create Bus With Analog Signal(s) form.
2. Enter the *Analog to digital thresholds*, in one or both fields. If you enter a value in only one field, signal values above that threshold are converted to 1, and values below the threshold are converted to 0. If you enter values in both fields, signal values that match the lower threshold are converted to 0, signal values that match the higher threshold are converted to 1, and any other values are converted to the unknown value, X.

   You do not need to specify the units for threshold values. SimVision extracts the correct threshold units either from the database or from the simulator, depending on the mode in which SimVision is running.

If you split a group that contains analog signals, the analog signals are not converted back to analog values; they remain digital signals.

Changing the Colors of Waveforms

When you display several analog waveforms in an overlay group, it can be difficult to determine which waveform belongs to which signal. SimVision lets you choose colors for each analog signal that you display.

To change the color of a waveform:

1. Select an analog signal in the signal list, then choose *Format – Color* from the menu bar. SimVision displays a menu of colors to choose from.
2. Select the color that you want to use.
3. If you do not see the color you want, click *More* to open the Select a New Color form and mix your own color.
4. Click *OK* to apply the color to the waveform.

Changing the Format of the Waveform Lines

By default, analog waveforms are drawn as solid lines.

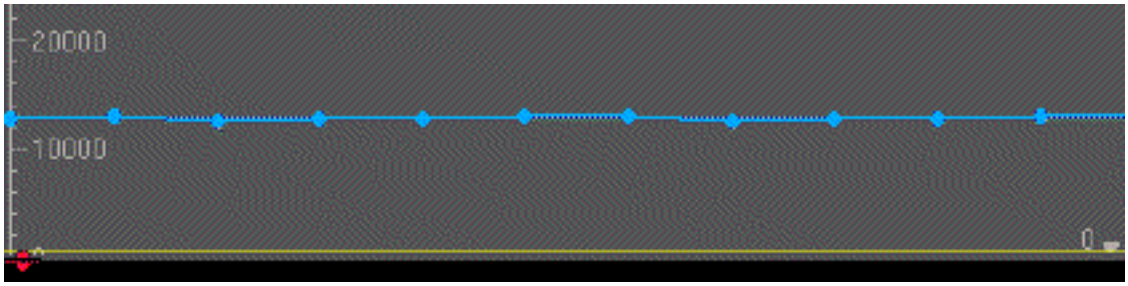
To change the waveform format:

➤ Select the signal whose format you want to change, then choose *Format – Symbol* from the menu bar. The menu shown in Figure 21-6 on page 275. On the right side of the menu, choose the symbol used to mark points along the waveform. Enable how you want to connect the points.

Figure 21-6 Format Symbol Menu Choices

Figure 21-7 on page 276 shows two analog signals. The first is displayed as points and lines. The second is displayed as a solid line.

Figure 21-7 Setting the Waveform Format



Tip

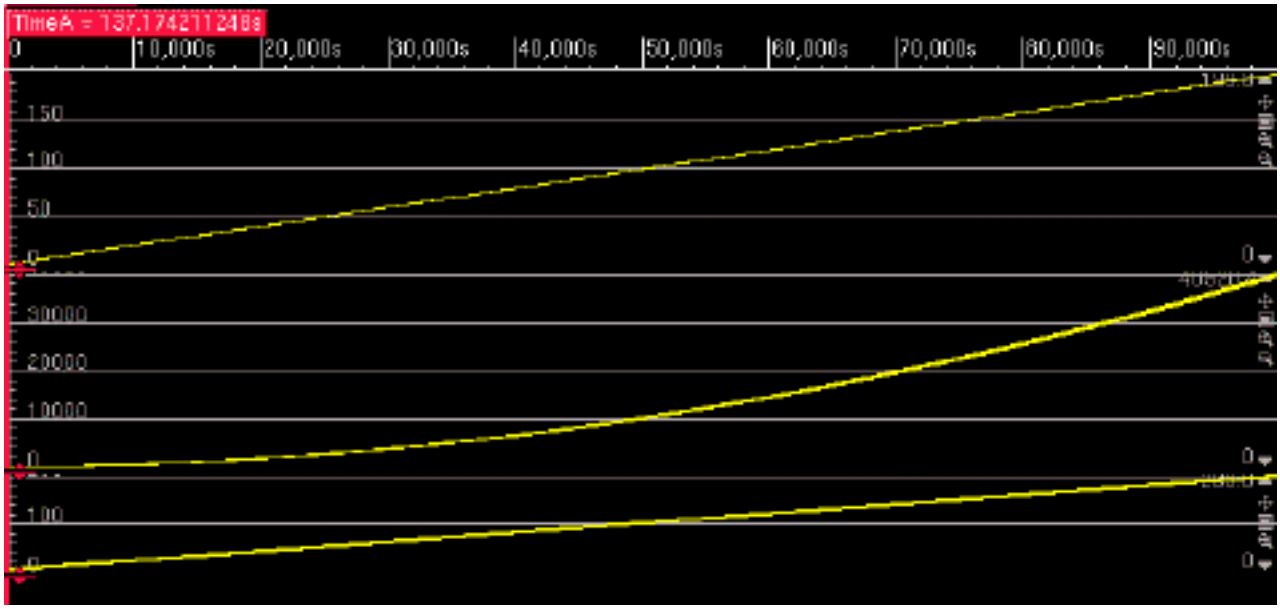
If you cannot see the symbols that you have chosen, zoom in until they are clearly visible.

Displaying a Y-Axis Grid

To turn on the y-axis grid:

1. Enable *View – Y Grid* in the menu bar. SimVision places a grid within the analog signal waveforms, as shown in Figure 21-8 on page 276. The grid does not appear in any digital signals that are in the Waveform window.

Figure 21-8 The Y-Axis Grid



Creating a Reference Line

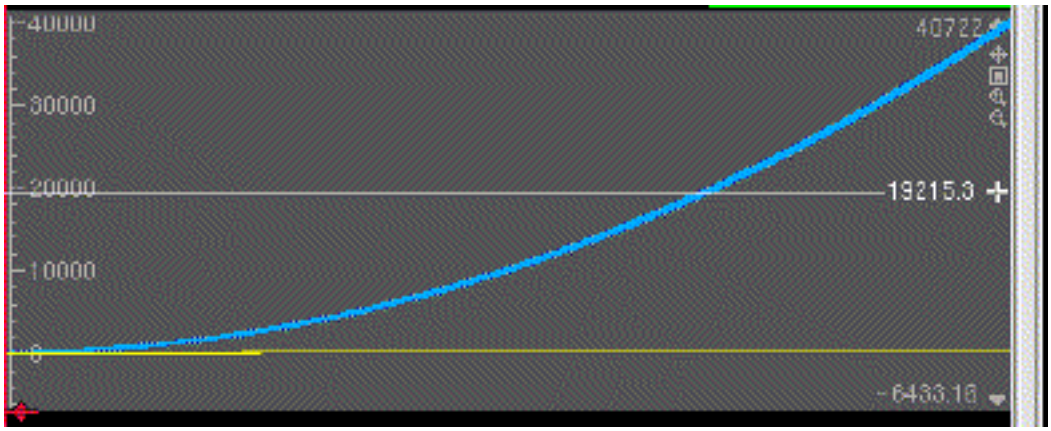
When analyzing analog data, you may want to place a reference marker along the Y-axis, so that you can see the deviation of a signal above and below that point.

To set a reference line:

1. Choose *Create Reference Line* from the pop-up menu over the waveform area. SimVision opens the Reference Line Create form.
2. Enter a number in the *Value* field, or move the slider up and down the scale until you reach the point where you want to place the reference line, then click *OK*.

   The reference line is a solid gray line, as shown in Figure 21-9 on page 277. Use the cross-hatch at the far right end of the line to adjust the location of the line along the Y-axis.

Figure 21-9 Displaying a Reference Line



Setting the Scale

The signals within an overlay can have different scales, such as current and voltage, but each analog overlay can display only one scale at a time.

To change the scale of an overlay:

1. Right-click the analog trace and choose *Activate* for the scale that you want to use.

You can also activate the scale associated with a particular signal by clicking on the style sample icon on the right side of the signal values area.

To set the minimum and maximum range of the scale you have chosen:

1. Right-click the analog trace and choose *Scale – Modify Range* to open the Range Modification form.
2. Adjust the slider or set the *Maximum* and *Minimum* fields with the values you want to use for the minimum and maximum range, or click *Automatic Fit* to adjust the range to the highest and lowest values in the waveform display, then click *OK*.

22

Viewing Transactions

Transaction-based verification lets you simulate and debug your design at a higher level of abstraction than when simulating at the signal level. For example, you can define a transaction to read a value, supply an address, and return a result. In this way, you can construct large-scale tests by defining transactions rather than by defining large numbers of test vectors.

You can create testbenches that generate transactions. If you simulate with SystemC, you can use a testbench written using the SystemC Verification (SCV) library and the Cadence Verification Extensions (CVE) to specify transactions to record. If you include assertions in your simulation, you can view sequential assertions as transactions.

You can also define transactions by embedding commands in your design models: SDI2 for C++ models, SDI-Verilog for Verilog models, and SDI-VHDL for VHDL models.

The simulator gathers information about your design transactions and stores the information in a simulation database. You can view the resulting transactions in the Waveform window, along with the lower-level signal waveforms.

If you have Transaction Explorer (TxE), you can interactively query transaction information stored in simulation databases. For information about TxE, see the *Transaction Explorer User Guide* and the *Transaction Explorer Reference Manual*.

Viewing Transaction Streams

Important

In IUS 5.6 and prior versions, SimVision displayed transactions that could not overlap on fibers. As of IUS 5.7, the Waveform window can display overlapping transactions. The term *fiber* has been replaced by *stream*, to reflect this change. The two terms are synonymous.

Transactions are displayed in the SimVision Waveform window on *streams*, shown as purple lines. One row of transactions corresponds to a transaction stream. Each transaction in the row is drawn as a rectangle with rounded corners, as shown in Figure 22-1 on

page 280. Assertion transactions in the Finished state are green, and Failed transactions are red.

Figure 22-1 Transactions on a Stream

An overlap occurs when any part of a transaction is contained between the start and end of another transaction. As shown in Figure 22-1 on page 280, whenever an overlap occurs, the transaction heading is dimmed. For more information about overlapping transactions, see "Viewing Overlapping Transactions" on page 284.

Finding Streams using the Design Search Sidebar Tab

The simplest way to find all transaction streams and error signals in your simulation database is by using the Design Search sidebar, because it can search all scopes in all open databases at the same time.

To find all streams in the Design Search tab of the sidebar:

1. In the *Consider* field, choose *Only Signals/Variables*.
2. Disable all but *Streams* (or *Fibers*), , and *Error Signals*, .
3. Click *Search Now*.

SimVision returns a list of all transaction streams and error signals in the database.

Attributes are classified as begin attributes, end attributes, and special attributes. A transaction attribute is similar to a variable—it has a type and a value. To see the attributes for the transactions in each stream, you can double-click on a stream in the signal list and expand it.

For more information about transaction attributes, see "Viewing Attributes" on page 283.

Finding Streams in the Design Browser Signal List

To search for streams and error signals in the Design Browser signal list:

1. Deselect all but *Show/Hide Fibers*, .

2. Select a scope in the sidebar.

The streams defined in that scope are displayed in the signal tree.

For more information about using the sidebars, see Chapter 5, “Accessing Design Objects.”

Viewing Transaction Information

The Waveform window displays transactions as rectangular areas that span periods of simulation time. These areas contain information about the values of transaction attributes during that time period.

Initially, the waveform displays only the name of the transaction. Use the vertical slider to display the transaction attributes. For detailed information about viewing attributes, see “Viewing Attributes” on page 283.

You can view transaction information within the waveform area in the following ways:

- Use the scroll bar at the bottom of the waveform area to scroll through the stream, from time 0 to the end of simulation time.
- Transactions can be nested in substreams of a stream. To expand and collapse the transaction hierarchy, click on the + and - buttons next to the stream names.
- Sometimes there are too many lines of attribute information to fit in the waveform area. To enlarge the vertical space allowed for the transaction, drag the vertical slider down.
- Sometimes the attribute values do not fit within the transaction. In this case, zoom in to enlarge the width of the transaction.
- To move the primary cursor to the next or previous edge of the transaction, click to select a stream in the signal list, then click *Next Edge*, , and *Previous Edge*, .
- Shift-click or double-click within a transaction waveform to select it. All instances of the transaction are highlighted.
- To determine the sequence time at which a transaction begins, select the stream and click *Next Edge*, , until you find the transaction you are interested in.

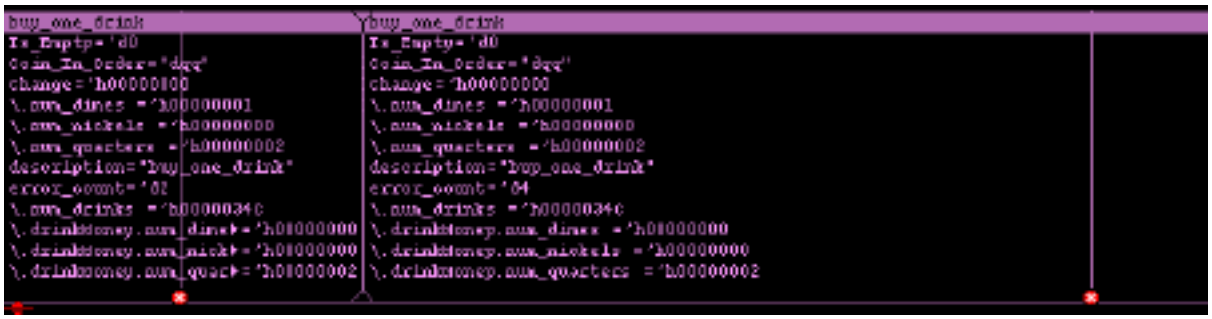
  The sequence time for the start of the transaction appears in the primary cursor flag. For example, time 13,300(4) ns indicates sequence 4 at time 13,300 ns. For more information about sequence time, see “Viewing Events in Sequence Time” on page 255.

Viewing Error Information in the Waveform Window

For transactions on a stream, error transactions are marked with an icon at the start time of the error.

When the transaction verification model detects errors that occur in the simulation, the errors show up in the waveform as sequence time markers, as shown in Figure 22-2 on page 282.

Figure 22-2 Error Markers in a Transaction



The simulator also increments a signal named `error_count` whenever a transaction error occurs. You can display the `error_count` signal in the Waveform window, as shown in Figure 22-3 on page 282.

Figure 22-3 Displaying the error_count Signal

The `error_count` signal keeps a running count of the error transactions recorded in the database up to the current point in the simulation. When an error is recorded in the past, the error count increments in the past at the time at which the error transaction was recorded, and all subsequent counts are also incremented.

In the Waveform window for a live simulation, recording an error in the past alters the existing waveform. If the database was split up into simulation time segments because of a save, restart, or reset, the error count reflects only errors that were recorded within segments of the simulation timeline for database files that are currently open. The error count marks only the beginning of an error transaction and not its end.

Viewing Attributes

If all of a transaction's attributes do not fit in its rectangle, a single vertical line is shown along the right edge of the block. You can roll over this line to activate a scroller that brings other attributes into view, similar to the way you can view portions of buses.

You can right-click on any attribute in the transaction block to display a pop-up menu.

In addition to the zoom, sequence time, and marker menu choices, the following operations are available:

- *Break Out Attribute: attribute_name*

  Places *attribute_name* into a new trace. The trace displays the single attribute's value in a sample-and-hold manner.

  In time spans where there are no transactions with that specific attribute, the trace is left blank. During a transaction that does contain the attribute, the value is displayed in the same way that any other digital value is displayed.

  If another transaction starts and it also contains the specific attribute, the trace transitions to the new value.

  Note: The `stacking_priority` value (see "Viewing Overlapping Transactions" on page 284) is not used in deciding what to draw in this trace.

  For more information about using this option, see "Viewing Transactions with a Given Attribute or Attribute Value" on page 286.

- *attribute_name Radix*

  Controls the radix that is used to display attribute values within the transaction blocks. A pull-right menu provides the following choices:

  - *As Recorded*
  - *Binary*
  - *Octal*
  - *Decimal*
  - *Hexadecimal*
  - *ASCII*

  Note: The radix changes for the transaction display only. If you break out an attribute, it maintains the original radix.

- *Expand Stream By attribute_name*
  Expand Stream
  Collapse Stream

  For more information about using this option, see "Viewing Transactions with a Given Attribute or Attribute Value" on page 286.

Viewing Overlapping Transactions

For overlapping transactions, the end of the first transaction is not visible because the second transaction is drawn over it. The overlap region is called a stack. The second transaction is on top of the stack, and the first transaction is on the bottom of the stack.

Changing the Stacking Order

Transactions in a stream are drawn from left to right, based on their start times. If a new transaction starts before a previous one ends, the previous one is obscured, because the new transaction is drawn over it—that is, the default stacking order is determined by the start times, so that later in time is higher in the stack.

You can change the stacking order in several ways.

Use the stacking_priority Attribute

When you write your transaction recording code, you can specify a reserved attribute called `stacking_priority` on each transaction.

- A positive `stacking_priority` value raises the transaction above other transactions that might otherwise obscure it. For example, a value of 2 raises this transaction above all others with a lower `stacking_priority`.
- Similarly, a negative `stacking_priority` value lowers the transaction.
- If no `stacking_priority` is recorded, the value is 0.

Note: The stacking order might not be obvious from the waveform display. To check the stacking order, use *Re-order* on the pop-up menu on a given transaction.

Override the Stacking Priority in TxE

In TxE, you can override the `stacking_priority` value on each individual transaction.

You can create TxE scripts that control stacking order in a variety of ways—for instance, based on ranges of attribute values. For more information, see the *Transaction Explorer Reference*.

Interactive Control of the Stacking Order

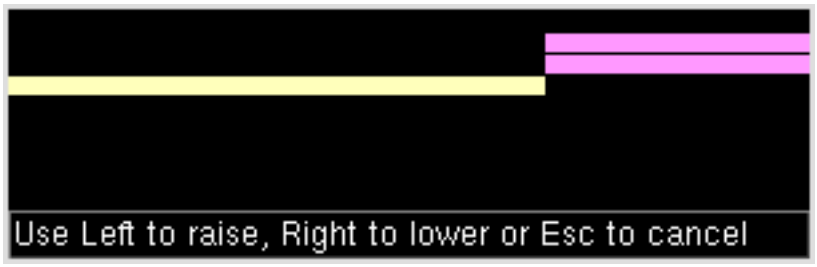
You can interactively control the stacking order in the Waveform window. Interactive stack adjustments override any recorded `stacking_priority` attribute values.

Note: Interactively modified `stacking_priority` values are not saved in the SST2 database.

To manipulate the transaction stack, do the following:

1. Right-click on the heading of any transaction and choose *Re-order*. This displays a cross-section of the stack, as shown in Figure 22-4 on page 285.

Figure 22-4 Transaction Stack Cross-Section



You can do the following:

- Left-click a stream indicator to raise it to the top of the stack.
- Right-click a stream indicator to lower it to the bottom of the stack.
- Click anywhere outside the dialog box to close the stacking order window and save your changes.
- Press `Esc` to close the dialog box without changing the stacking order.

2. Choose *Raise attribute_name* or Lower *attribute_name* to send the transaction to the top or bottom of the stack.

Remove Stacking

To simultaneously view all of the transactions unobstructed, right-click the stream and choose *Expand Stream*. This changes how transactions are drawn in the trace. The expanded view

does not represent the `stacking_priority` in any way. In the expanded view, the top portion of the transaction stream is drawn as usual, but the overlapping transactions are drawn below it in one or more additional rows, so that none of the transactions overlap. The exact position of transactions will change, depending on zooming or scrolling.

The *Collapse Stream* menu item undoes the expand operation.

Filtering Streams

You might have a stream with many transactions, only a few of which interest you. There are several ways for you to look at the transaction information you want:

- Creating a new stream of transactions that have a given attribute or attribute value
- Expanding a stream by depth
- Creating a new stream that matches one or more specified conditions

Viewing Transactions with a Given Attribute or Attribute Value

You can display all of the transactions on a stream that have a specific attribute and value, as follows:

➤ Right-click any attribute in the transaction block and choose *Expand Stream By attribute_name*. SimVision displays new traces for each value of the attribute.

Tip

To display all transactions of a given type, you can expand on the `trans_type` attribute.

If there are many different values for the selected attribute, a limited number of traces are added. A single vertical line is shown along the right edge of the signal list area. You can roll over the right edge of this area to reveal a scroller that can be used to bring the other traces into view.

To create more trace subsets:

➤ Right-click again to select another attribute to subset on, as described earlier.

The display adds new traces for each value of the new attribute.

Expanding Streams by Depth

Transactions are automatically assigned a `depth` attribute that specifies its level in the stream. A depth of 0 indicates the top level, 1 is the first level under the top level, and so on.

Note: Assertion transaction do not use the `depth` attribute.

When you left-click on the + button next to a stream name, a trace is added for each unique `depth` value on the stream. For example:

```
rw_stream[[depth==0]]
rw_stream[[depth==1]]
rw_stream[[depth==2]]
```

When you expand a trace that has already been expanded on depth, it is expanded on the attribute named `trans_type`. For example:

```
rw_stream[[depth==0 ]][[trans_type==bus_tx]]
rw_stream[[depth==0 ]][[trans_type==dphase]]
```

If there is no `trans_type` attribute in the stream, this expansion is skipped. Thereafter, the default expansions are on additional attributes based on the recent history of other expansions.

When you right-click a +/- button, a pop-up provides other expansion choices, as follows:

- The *attribute_name* of the most recently selected attribute, then the *attribute_name* of the next most recently selected attribute, and so on.
- *Transaction Attribute*

  Presents the Select Expand Attribute form with a pull-down menu that lets you select which attribute to expand on. Click *OK* to expand the trace on the attribute, and add the attribute name to the top of the list of most recently selected attributes.
- *All Attributes*

  Adds a new trace for each attribute on the stream. Unlike the previous expansion operations, these traces each have the attribute broken out as described for *Break Out attribute_name* in “Viewing Attributes” on page 283.

The transaction depth represents the number of parent relationship links above a transaction. However, if a parent is on another stream, or a transaction has multiple parents, or the parent/child relationship is circular, the transaction depth is unclear, and defined to be 0.

Because a child transaction is not required to fit within the time range of its parent, the relationship is not visually obvious. You can use the following techniques to view parent/child relationships:

- Use the TxE *View* menu to highlight all parent/child transactions.
- Use the `depth` attribute.

  See "Viewing Transactions with a Given Attribute or Attribute Value" on page 286 for the technique to use to view transactions with a given attribute.

Filtering a Stream Based on a Condition

SimVision provides functions that you can call from the Expression Calculator to filter streams and generate a new stream that contains only the transactions that you want to view. The stream filter function filters a stream such that only those transactions where a specified condition is true are included in a new stream.

Using the Stream Filter

The stream filtering operation has the following syntax:

`stream_name``[[`*`condition`*`]]`

The *`condition`* argument is any Boolean expression that you create with the Expression Calculator. The stream filter operation filters the stream so that only those transactions where the condition is true are included.

You can use the name of an attribute directly in a comparison, or with the `exists` function inside the condition. For example, `stream[[change > 0]]` creates a derived stream that includes only those transactions of `stream` where the `change` attribute has a value greater than zero, and `stream[[exists(change)]]` includes those transactions that have a `change` attribute. A comparison involving an attribute that does not exist in the transaction is false for that transaction.

Conditions can be combined with the `&&`, `||`, and `!` logical operators.

To start the Expression Calculator, you can do one of the following:

- ➤ Right-click the stream displayed in the Waveform window and choose *Create Condition*.
- ➤ Select the stream in the Waveform window and click *Expression Calculator*,  .

Either of these methods inserts the stream name into the Expression Calculator expression area. You can then enter the condition on which to filter the stream. For example, if you want to display a derived stream that contains the transactions of the `Fiber_In_Tvm_drink_test` stream where the `Coin_In_Order` attribute is equal to the value `qdq`, enter the following into the expression area:

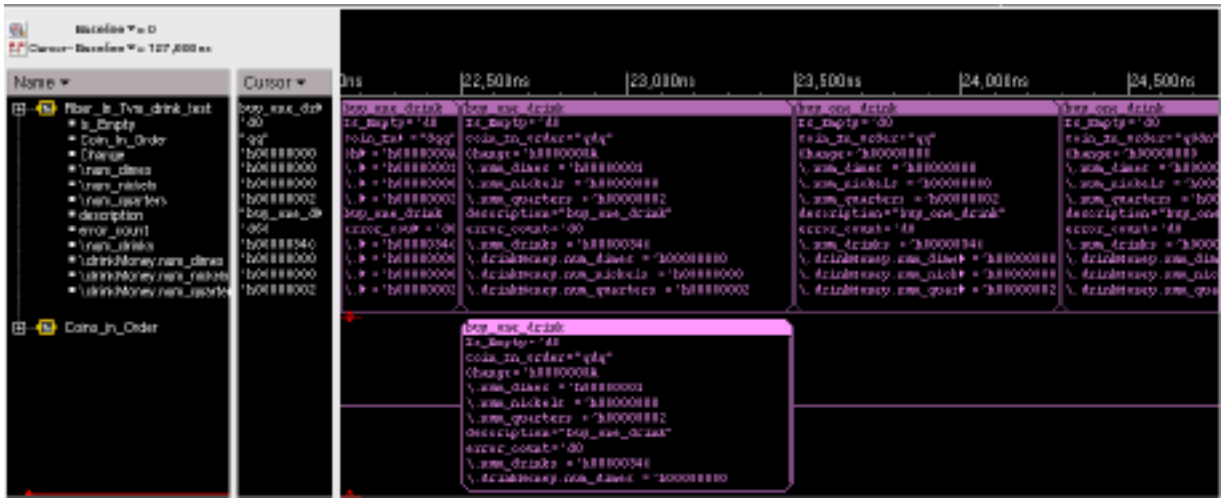
```
TBDatabase::test_drink.drink_test.Fiber_In_Tvm_drink_test[[Coin_In_Order ==
"qdq"]]
```

When you click *Waveform*, , in the Expression Calculator, the new stream is added to the Waveform window.

Note: If you have not entered a current scope, you need to use the full hierarchical name of the stream.

Figure 22-5 on page 289 shows two streams. The first contains all transactions. The second filters all but the transactions with the `Coin_In_Order` attribute equal to `qdq`.

Figure 22-5 Filtered Transaction Stream



If you want to refer to the expression later, you can enter a name in the *Name* field of the Expression Calculator and click *Check*, , to save the expression to the specified name. The expression will then show up on the Expressions tab of the Properties window.

Using the genevents Function

The *`condition`* argument can also call the `genevents` function. The `genevents` function specifies one or more simulation sequence times. A transaction that begins at any of the specified times is included in the generated stream.

The genevents function has the following syntax:

```
genevents(Time(Sequence)Units, ...)
```

Tip

One way to determine the *Time(Sequence)* of a transaction is to select the transaction (shift-click the left mouse button), then click the TxE *Next* or *Previous Transaction* button. The *Time(Sequence)* is displayed in the cursor flag.

Suppose you have a stream that has several transactions, but you want to look at only those transactions that occur at specific times. You can write an expression to display only those transactions. For example:

```
TBDatabase::test_drink.drink_test.Fiber_In_Tvm_drink_test[[genevents(8500(7)ns,
13300(4)ns)]]
```

The streamoverlay Function

The streamoverlay function creates a stream from any number of input streams. The new stream contains all transactions from the input streams.

This function has the following syntax:

```
streamoverlay( stream1[N], stream2[N], ... )
```

For example, the following expression uses the streamoverlay function to analyze two transaction streams. The second stream is a child of the first. Therefore, portions of these streams overlap:

```
streamoverlay(TBDatabase::test_drink.drink_test.Fiber_In_Tvm_drink_test[1],
TBDatabase::test_drink.drink_test.Fiber_In_Tvm_drink_test[2] )
```

Overlapping areas of the overlaid transactions are displayed as described in "Viewing Transaction Streams" on page 279.

The format, match, and rematch Functions

The format, match, and rematch functions can perform pattern matching operations on transaction attribute values. You can use these functions alone or as arguments to the stream filtering function to filter streams by matching string values.

With the format function, you define a format string and specify one or more signals. The function substitutes the signal values into the format string.

The format function has the following syntax:

```
format( format_string, signal[,signal2, ...] )
```

The *format_string* can contain any of the format specifiers listed in Table 22-1 on page 291.

Table 22-1 Format Specifiers

Format Specifier	Format
%c	Character
%s	String
%b	Binary
%d	Decimal
%o	Octal
%x or %h	Unsigned hexadecimal
%f	Floating-point number
%e	Real number in mantissa-exponent form
%g	%e or %f, whichever is shorter
%t	Decimal time scaled from the timescale of the object's module to the simulation's timescale
%v	Strength value, wires only

For example, you can write a `format` expression to display an error count in decimal format, as follows:

```
format ("Error Count is %d",
TBDatabase::trans_global.error_count )
```

This expression creates a waveform such as the one shown in Figure 22-6 on page 291.

Figure 22-6 Waveform Created by the format Function

You can use the `format` function to format multiple attributes. For example:

```
format ("Value of num_drinks is %d , Coin_In_Order = %s",
TBDatabase::test_drink.drink_test.Fiber_In_Tvm_drink_test.buy_drinks.num_drinks,
```

```
TBDatabase::test_drink.drink_test.Fiber_In_Tvm_drink_test.buy_one_drink.Coin_In_
Order)
```

The `match` function determines whether an input signal matches a glob-style pattern. The `rematch` function determines whether an input signal matches a regular expression. Regular expressions are more expressive than glob-style patterns, but they are more complicated to write.

Table 22-2 on page 292 shows the special characters you can use to specify glob-style patterns and regular expressions.

Table 22-2 Special Characters for Glob-Style Patterns and Regular Expressions

Search Criteria	Glob Rule	Regular Expression Rule
Match any single character	?	.
Match character *c*	\c	\c
Match 0 or more characters	*	.*
Match 0 or more occurrences of *regexp*		*regexp**
Match 1 or more occurrences of *regexp*		*regexp*+
Match 0 or 1 occurrence of *regexp*		*regexp*?
Match a set of characters	[*abc*]	[*abc*]
Match a range of characters		[*a-z*]
Match any characters except those in a set		[^*abc*]
Match any characters except those in a range		[^*a-z*]
Match any character or *regexp* in a set	{a,b,c}	*regexp* | *regexp*
Match the beginning of a string		^
Match the end of a string		$

The following expression uses `rematch` to return `0` when the error count is `1`, and `1` when the error count is `0`:

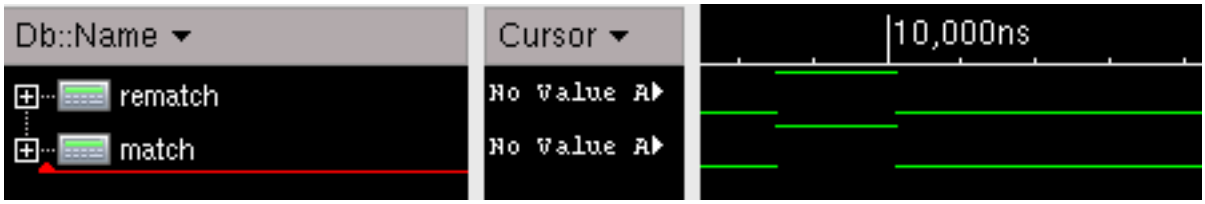
```
rematch("^[0]$", format("%d",
TBDatabase::test_drink.drink_test.Fiber_In_Tvm_drink_test.Change))
```

You can use the `match` function to produce the same results, as follows:

```
match("0", format("%d",
TBDatabase::test_drink.drink_test.Fiber_In_Tvm_drink_test.Change))
```

These expressions return waveforms such as those shown in Figure 22-7 on page 293.

Figure 22-7 Waveforms Created by the match and rematch Functions

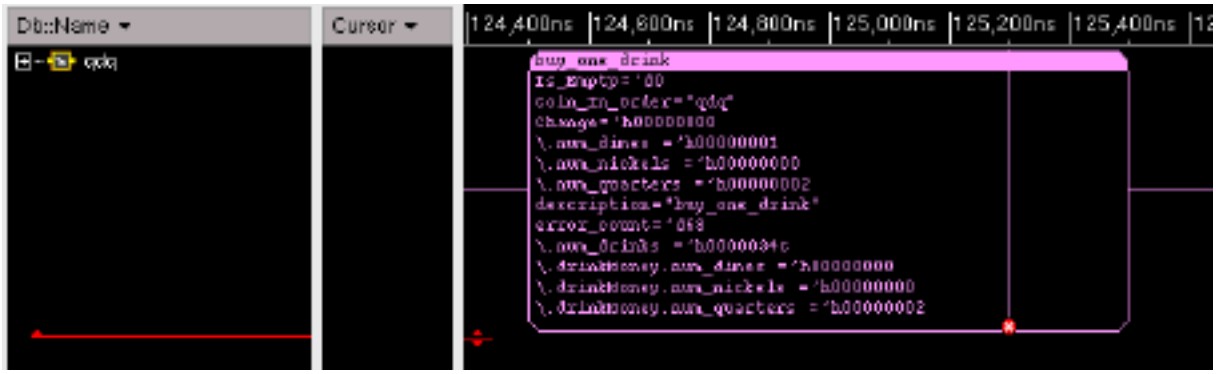


You can use the `match` and `rematch` functions with the stream filtering operation to filter streams by matching string values. For example, the following expression creates a new stream of the transactions whose `Coin_In_Order` attribute contains the value `qdq`.

```
TBDatabase::test_drink.drink_test.Fiber_In_Tvm_drink_test[[match("qdq",
Coin_In_Order)]]
```

Figure 22-8 on page 293 shows the stream that is created by this expression.

Figure 22-8 Transaction Created by the Stream Filter and match Functions



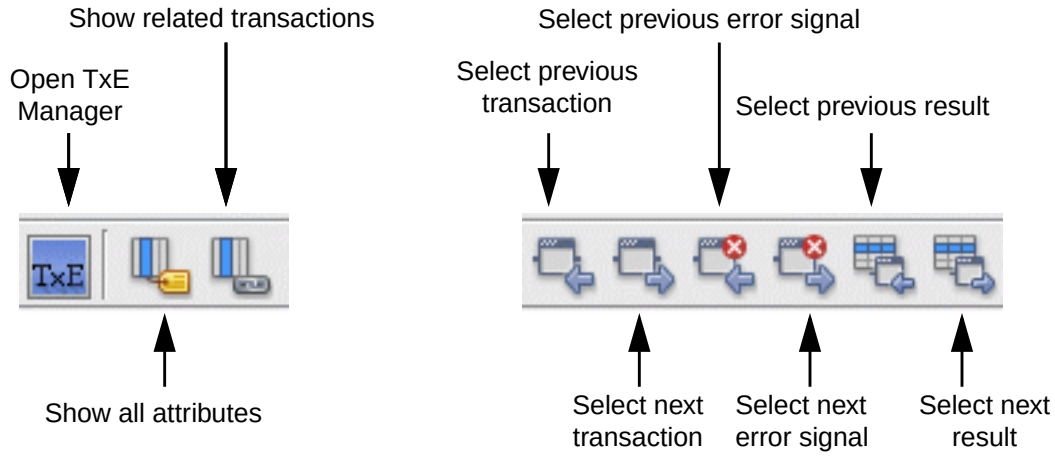
Using the Transaction Explorer Toolbar

You can use the Transaction Explorer (TxE) to interactively explore information in your simulation database.

If Transaction Explore is installed on your system, the Waveform window contains the following additional button:

Click to show or hide the Transaction Explorer toolbar, shown in Figure 22-9 on page 294.

Figure 22-9 Transaction Explorer Toolbar



For more information about the Transaction Explorer, see the *Transaction Explorer User Guide* and *Transaction Explorer Reference*.

23

Creating Custom Views of Simulation Data

The Register window lets you create custom views of simulation data. These views can contain simulation objects and their values, plus graphical elements, such as rectangles, lines, and arrows. You can use the Register window to monitor data from a running simulation or to view data stored in a simulation database.

The Register window is made up of pages, where each page contains one view of the data. The Register window displays one page at a time, but you can easily switch from page to page.

You can create multiple Register windows. Although each window is made up of the same collection of register pages, you can display a different page in each window. By using multiple Register windows, you can monitor different views simultaneously.

Opening a Register Window

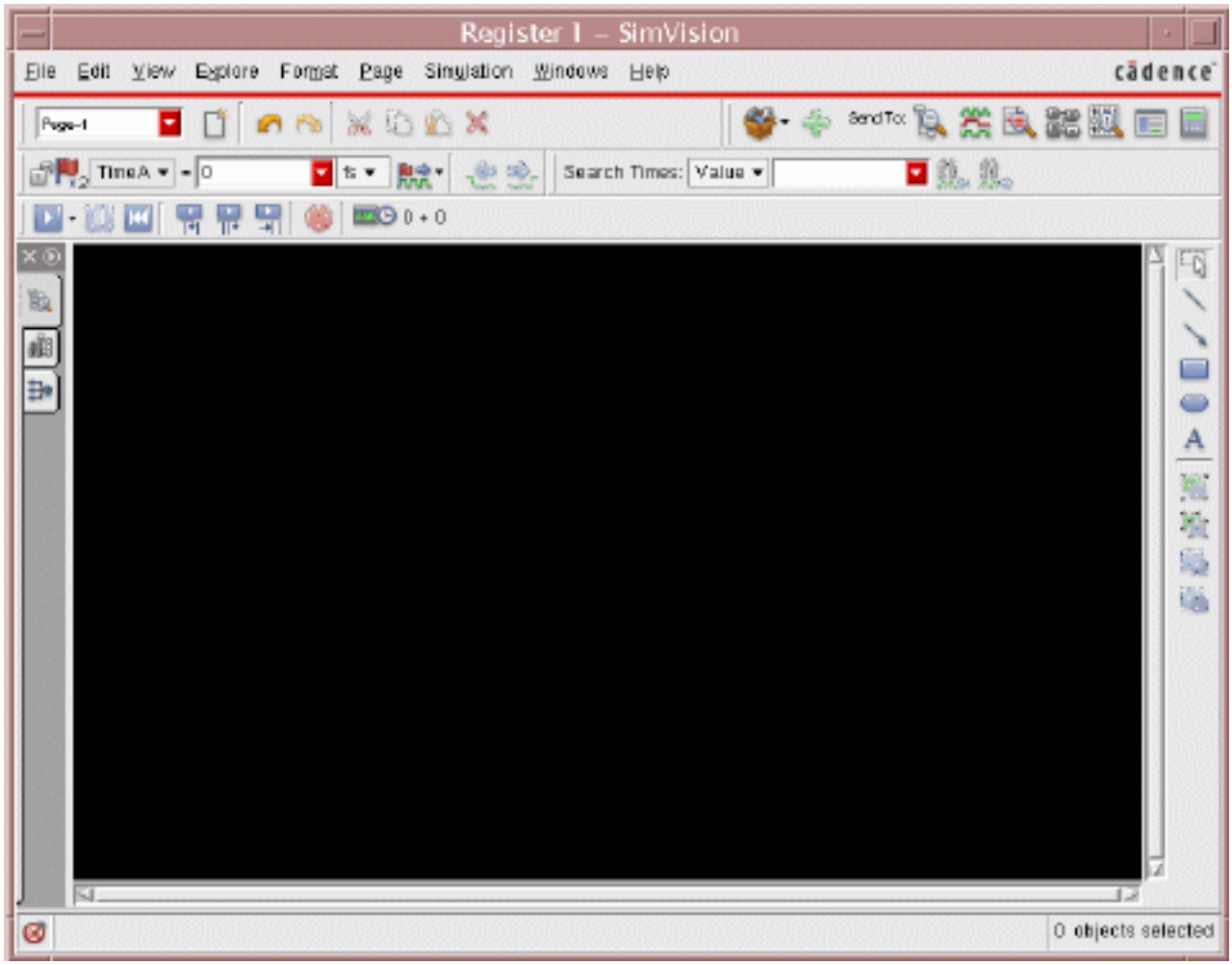
To open the Register window:

➤ Select some signals and click *Register Window*,  , or choose *Windows – New – Register* from the menu bar of any SimVision window.

If you do not preselect any signals, a blank Register window opens, as shown in Figure 23-1 on page 296. You can then use the Design Browser or Design Search sidebar to pick the signals you want to add to the Register window.

Initially, the Register window contains the default set of toolbars. You can control which toolbars are displayed by enabling or disabling them in the *View* menu. Disabling a toolbar gives you more room for data in the Register window.

Figure 23-1 Register Window



Managing Register Pages

When you open a Register window for the first time, it has one register page, named `Page-1`. As you add pages, their names are added to the drop-down list of pages.

To create additional pages:

➤ Click *New Page*, , or choose *Page – New* from the menu bar. The Register window opens the New Page form. By default, new pages are named `Page-`*N*, where *N* is a sequential number. Use this form if you want to give the page a different name.

To create a duplicate of an existing page:

➤ Choose *Page – Clone* from the menu bar.

To rename a register page:

1. Choose *Page – Rename* from the menu bar. This opens the Rename Page form.
2. Enter the new name for the page and click *OK*. The name is updated in the current Register window, and in any other Register windows that you have opened.

To switch from one page to another:

➤ Select a page from the drop-down list of pages.

Using the Target Icon in Register Windows

Every Register window has a target icon,  , located in the lower left corner of the window. You can enable this icon in only one Register window at a time. When you enable the target icon, you make the currently visible page the target of any operation that is aimed at Register windows. For example, when you send the object to the current page of the target Register window, that page also exists in every other Register window. Therefore, the object is also added to the same page in the other Register windows.

Adding Objects to a Register Page

The register page can display signals and variables in your design, as well as expressions that you have defined with the Expression Calculator.

There are several ways to add objects to a register page:

➤ Expand the sidebar to select the objects you want to add to the window. See Chapter 5, “Accessing Design Objects” for information on using the sidebar.

➤ Drag them from another SimVision window and drop them into the Register window.

➤ Select the objects in another window and click *Add*,  , in the Register window.

➤ Click *Copy*,  , to copy selected objects in one window, then click *Paste*,  , in the Register window.

➤ Add one signal, select it in the Register window, then choose *Explore – Add Contributing Signals* or *Explore – Add Module Inputs* from the menu.

Note: If you copy and a paste a signal in the same Register page, the signal is pasted on top of the original signal. After you paste the signal, you must move it to another location in the page.

Display Format of Signals and Variables

Signals and variables are displayed in the following format:

type name = value

The *type* is an icon that indicates whether the signal is an input, output, inout, internal signal, or fiber.

The *name* is the object's simple name. Editing the name in the Register window does not change the object's name in the simulation database.

The *value* is the value of the signal or variable at the current simulation time. If you have copied the object from the Design Browser sidebar, the *value* is displayed in the radix specified in the database for that object. If you have copied the object from any other window, the *value* is displayed in the radix displayed by the Source Browser. And if the signal value changes during the current clock cycle, the transition is displayed with the notation *old-value -> new-value*.

Changing the Radix of a Signal

To change the radix of a signal:

➤ Select the object, then enable a radix in the *Format – Radix* menu.

You may want to display a value shifted by a certain amount of time. Time-shifting can occur relative to the current cursor position, or in absolute terms.

Shifting a Signal in Time

To time-shift a signal value:

1. Select the signal, then choose *Format – Shift* from the menu. This opens the Time Shift form.
2. Enter the amount of time by which you want to shift the value.
3. Enable *Relative* to shift the value relative to the current location; enable *Absolute* to shift the value relative to the beginning of simulation. Then click *OK*.

Annotating Register Pages

The drawing toolbar lets you add lines, arrows, rectangles, ellipses, and text to your register pages. You use the mouse to indicate the beginning and ending points of lines and arrows, the diagonally opposing points of rectangles and ellipses, and the beginning of a line of text.

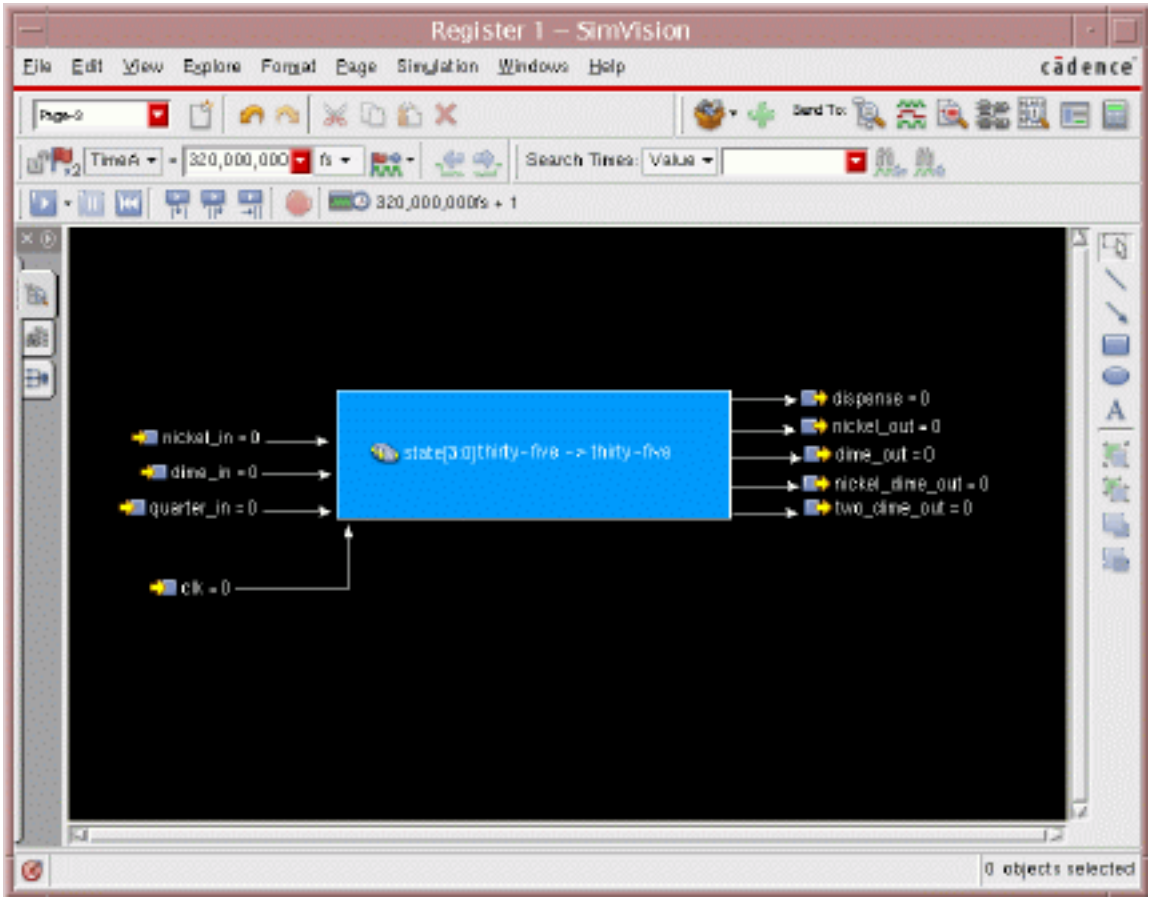
For example, to draw an arrow:

1. Click the arrow button.
2. Click and drag the mouse in the Register window. The arrow begins at the point where you press the mouse button and ends at the point where you release the mouse button.
3. Click and drag the mouse again to draw another arrow at a new location. You can continue to draw arrows in this way until you select another operation in the drawing toolbar, shown in Figure 23-1 on page 296.

Figure 23-2 Annotation Toolbar

Figure 23-3 on page 300 shows a register page that contains several signals. Arrows and rectangles show the relationship between the signals.

Figure 23-3 Annotated Register Window



Changing the Appearance of Objects

The *Format* menu lets you specify the appearance of objects in the window, including the signals that you add to the window, as well as the graphical objects that you create.

To change the color of an object:

1. Select the object, then choose *Format – Color*.
2. Select the color you want to apply to the object from the color menu.
3. If you do not see the color you want, click *More*. This opens the Select a New Color form, where you can mix your own color or enter the hex value for the color.

To change the size of the font for signal names, values, and text that you have created:

➤ Select the text object, then choose a font size from the *Format – Font* menu.

Grouping and Ungrouping Objects

Sometimes it is easier to manipulate several objects—such as signals, rectangles, and arrows—if you treat them as a single entity.

To group objects in the Register window:

➤ Select the objects, then click *Group*,  , in the drawing toolbar.

Important

A group that you create in this way is simply a set of graphical objects that you can treat as a single entity. It has no relationship to the groups that can appear in the Waveform window.

To split a group into its individual objects:

➤ Select the group and click *Ungroup*,  .

When you add a signal to the Register window, its `type`, `name`, and `value` are treated as a group. You can split a `value` from its `type` and `name`, but you cannot split the `type` and `name`.

Navigating through Simulation Time

The values of signals in the Register window are determined by the location of the primary cursor. As you move the cursor, the values are updated in the window.

The primary cursor in the Register window is linked to the cursors in the other SimVision windows. That is, if you change the location of the cursor in the Register window, it changes in the other SimVision windows. If you change the cursor in another SimVision window, it changes in the Register window.

When *Watch Live Data* is enabled,  , values in the Register window are not updated until simulation pauses or stops.

Printing and Saving Register Pages

To print the target register page:

1. Choose *File – Print* from the Register window menu. This opens the Print form.
2. In the *Printer* area, choose *Command* to print the register page, then enter the appropriate print command.

   Choose *Print to file* to save the register page as a postscript file, then enter a filename or click the browse button to choose a file. You do not need to enter a file extension; the file is given the `.ps` extension by default.
3. In the *Comments* area, enter a title and any other information that you want to appear at the top of the page.
4. In the *Print paper* area, choose your paper size, orientation, and color options. Each option has a drop-down list of choices.

   If you click *Fit to page*, the register page is enlarged or reduced to fit within the margins of the selected paper size.

24

Measuring Signal Values

The Measurement window lets you create tables of waveform measurements. You can perform the following measurements on the waveform data between the baseline and the primary cursor:

Value At Cursor	Displays the value of the signal at the primary cursor.
Value At Baseline	Displays the value of the signal at the baseline.
Baseline - Cursor	Displays the difference between those two values.
Slope Between Cursors	Displays the slope of the signal.
Minimum Value	Displays the minimum value of the signal.
Maximum Value	Displays the maximum value of the signal.
Average Value	Displays the average value of the signal.
Peak-to-Peak	Displays the difference between the peak values.
RMS Value	Displays the root mean square value.
Rise/Fall Time	Displays the rise or fall time, using value and threshold parameters that you specify.

Note: Many of these measurements are relevant only for analog data.

Opening a Measurement Window

To open a Measurement window:

1. In the Waveform window, set the baseline and the primary cursor to the beginning and ending simulation times that you want to measure, then select the signals that you want to measure.
2. Choose *Windows – New – Measurement*. By default, the Measurement window contains a single column labeled *Variable Name*.

3. In the Measurement window, click *Add*,  to add the selected signals to the window.
4. From the *Add Column* drop-down menu, select the measurements that you want to display. When no insertion point is set, columns are added to the far right side of the window.

   To set the insertion point, click on a column heading with the middle mouse button. This turns the column heading blue to indicate that the selected column the insertion point. Columns that you add are placed to the right of the insertion point. Click the middle mouse button again to remove the insertion point from the column.

   As you add columns, the Measurement window performs the requested measurements and displays the results in the table.

   Table cells can contain the following values:

n units	Shows the result of the measurement and the units of measure.
x	The result is the unknown value.
N/A	The measurement cannot be performed.
ERROR	An invalid result was obtained.

 *Tip*

Use the Preferences window to choose the columns that you want to appear by default in new Measurement windows, as described in “Measurement Window” on page 320.

Setting the Location of the Cursors

You can take measurements at different simulation times, by setting the baseline and primary cursor.

To set the baseline and primary cursor:

1. Select the primary cursor from the list of cursors. When you change the primary cursor, it also changes in the Waveform windows.
2. Set the simulation time for the primary cursor and baseline.

   SimVision keeps a history of the times you enter, so you can quickly return to a time by choosing it from the drop-down menu. You can also enter a partial time and press Tab. If the history contains one matching time, SimVision seeds the field with that string. If the

history contains more than one matching time, it displays a list of those times, and you can choose one.

All measurements are recalculated with the new locations, and the Measurement window is updated with the new results.

Moving Rows and Columns

You can move rows and columns to rearrange the contents of the Measurement window.

To move a column:

1. Click the middle mouse button within the column heading that you want to move.
2. Drag the column heading to the new location.

Important

You can move only one column at a time.

To move one or more rows:

1. Select the rows that you want to move, as follows:
   - Click on a single row to select it.
   - `Shift`-click to select contiguous rows.
   - `Control`-click to select non-contiguous rows.
2. Hold down the middle mouse button and drag the row to the desired location. When you release the mouse button, the rows are inserted at that new location.

When you move a column, its title is displayed in blue, indicating that it is the insertion point. Any columns that you add are placed to the right of the insertion point. You can remove the insertion point by clicking on the column with the middle mouse button.

Sorting the Contents of the Window

To sort the window:

1. Place the cursor anywhere within the column that you want to sort.
2. Right-click and choose *Arrange – Sort* to sort the measurements in ascending order, or choose *Arrange – Reverse Order* to sort the measurements in descending order.

An arrow is added to the column heading. The direction of the arrow indicates whether the values are in ascending or descending order.

Setting Rise/Fall Time Parameters

When an analog signal represents binary data, it has a low value and a high value, and you define the range of values that represent those low and high values. *Rise/Fall Time at Cursor* measures how long it takes to go from low to high, or high to low.

The *Rise/Fall Time at Cursor* column uses the following parameters when calculating this measurement:

- Low value represents the bottom of the range of values that the signal can have. The default low value is 0.0.
- High value represents the top of the range of values that the signal can have. The default high value is 3.3.
- Low threshold represents the lowest percentage of the low or high value that a signal can have and still be considered low or high. The default low threshold is 10.0%.
- High threshold represents the highest percentage of the low or high value that a signal can have and still be considered low or high. The default high threshold is 90.0%.

To set these parameters:

1. Right-click the *Rise/Fall Time* column heading and choose *Edit Parameters*. This opens the Rise/Fall Time Parameters form.
2. Enter the new parameter values and click *OK*.

   Whenever you enter a new parameter, it is added to that parameter's drop-down list. You can easily go back to a previous setting by choosing it from the list.

Saving Measurements

To save the results of your measurements:

1. Choose *File – Save Window Contents* from the menu bar. The Export Measurement Window form opens.
2. Enter a filename and click *Save*. This creates a file in ASCII format.

25

Setting Preferences

You can set the following preferences to customize SimVision windows. Preference settings take effect in all open windows when you apply them, except where noted.

- General Options
- Signal Options
- VHDL Signal Options
- Verilog Signal Options
- Verilog AMS Signal Options
- Keyboard Shortcuts
- Toolnet Cross-Probing
- Simulation Settings
- Waveform Window
- Waveform Display
- Analog Waveform
- Waveform Keyboard Shortcuts
- Design Browser
- Scope View
- Signal List
- Source Browser
- Source Browser Colors
- Source Browser Files
- Measurement Window

- Schematic Tracer
- Schematic Tracer Colors
- Memory Viewer
- Memory Viewer Colors
- Simulation Cycle Debug
- Transaction Explorer

General Options

To customize the general appearance and behavior of SimVision:

1. Choose *Edit – Preferences* from the menu bar and select *General Options* in the left column of the Preferences form.
2. Select the toolbar style that you prefer. You can choose to display only the icons, only the icon labels, or both icons and labels. By default, SimVision displays only the icons.
3. Enable *Limit the number of lines kept in the console window*, if you want to set the size of the Console window buffer. The default size is 5000 lines.
4. Enable the *Restore Application State* button if you want SimVision to automatically restore any settings that were saved during a previous session.

   Note: The automatic restore feature will not be supported in the next release of SimVision. You should save and restore your simulation environment with a command script, as described in "Saving and Restoring Your Debugging Environment" on page 53.
5. Choose the behavior you want SimVision to have when it exits:

   - Enable *Save Application State* if you want SimVision to save any settings that you define during a session. When used in combination with *Restore Application State*, you can restore all of the windows, groups, and cursors in subsequent sessions.

     Note: The automatic save feature will not be supported in the next release of SimVision. You should save and restore your simulation environment with a command script, as described in "Saving and Restoring Your Debugging Environment" on page 53.
   - By default, SimVision deletes all unused cursors and groups when it exits. When you start up SimVision again (without the `-norestore` option), it does not restore the

unused objects. Disable *Delete Unused Groups* and *Delete Unused Cursors* if you do not want these things deleted.

- ❑ By default, SimVision prompts you on exit. You must click *Yes* to exit the tool. Disable *Prompt* in the Preferences form to exit without displaying the Exit form.

6. Choose startup defaults for the following options:

   - ❑ Enable *Show Full Signal Names* if you want to display the full hierarchical names of signals, not including the database name.
   - ❑ Enable *Show Database Names* if you want to include the database name in signal names.

Note: You might need to scroll the Preferences window to see the following options.

7. Enable *Watch live data enabled with creating cursor*, if you want the SimVision windows to update as simulation progresses. This is the default when you are connected to a simulation.
8. Enable *Create Cursor for each New Window*, if you want each window to have its own primary cursor. By default, windows share cursors, so that as you move a cursor in one window, all windows update to show the same cursor view. When you enable this option, each window maintains its own view.
9. When *Create Cursor for each New Window* is enabled, you can specify the time units for those cursors.
10. By default, SimVision displays a warning message when you try to save a command script that that may not restore properly. This can happend if you have used an input file to set up the simulation environment. If you do not want to see this warning message, disable the *Warn me when saving scripts that may be problematic due to input files* option.

Signal Options

To customize the signal list:

1. Choose *Edit – Preferences* from the menu bar, and select *Signal Options* from the list on the left.
2. When you enable *Display signal transition values*, SimVision displays two values when the value of an object changes during the current clock cycle. For example, if a signal transitions from 0 to 1 in the current clock cycle, SimVision displays the transition as `‘b0 -> ‘b1`.

3. Enable *Display signal value underscores* to display an underscore between every eight characters of an octal, hexadecimal, or binary value. Disable this option to display the value with no underscores.
4. Enable *Display Real values with full precision* if you want to display real values with full precision in the signal list. Ordinarily, real values are limited to five decimal places.
5. Use *Display signal type icons* to enable or disable the signal type icons. You can disable icons if you need to conserve space in your Waveform windows.
6. When you enable *Highlight signal names that have been traced with Trace Sidebar*, signals that you have traced are highlighted in all windows in which they are displayed. In the Source Browser, they are displayed in green. In the other windows, they are underlined. Choose the *View – Clear Trace Highlights* in any window to clear the highlights in all windows.
7. Enable *Color signal names according to type* to use different colors for each signal type, then select the colors that you want to assign to those signal types.
8. Click *Default colors* at any time to return to the default color scheme.

VHDL Signal Options

To change the colors of waveforms:

1. Choose *Edit – Preferences* from the menu bar, expand *Signal Options* in the list on the left, and select *VHDL*. The Preferences form displays the VHDL color map.
2. To change a color setting, click on the color that you want to change, and choose the new color from the pop-up menu.

   If you want to mix your own color, click *More* in the pop-up menu. This opens the Select a New Color form. Slide the pointers to the left and right to adjust the amount of red, green, and blue that you want to use. The *Selection* fields show the hexadecimal number for the color and a color sample.
3. Click *OK* when you are satisfied with the color.

Verilog Signal Options

To change the colors of waveforms:

1. Choose *Edit – Preferences* from the menu bar, expand *Signal Options* in the list on the left, and select *Verilog*.

2. Enable *Show Signal Strength in Values* if you want the Waveform window to display the strength along with the signal values.

   Note: This setting does not affect windows that are already open, only new Waveform windows that you create.

3. Enable *Show colors by value* to color waveforms according to their values, and to choose the colors for each value.

4. Enable *Show colors by strength* to color waveforms according to their strength, and to choose the colors for each strength.

5. To change a color setting, click on the color that you want to change, and choose the new color from the pop-up menu.

   If you want to mix your own color, click *More* in the pop-up menu. This opens the Select a New Color form. Slide the pointers to the left and right to adjust the amount of red, green, and blue that you want to use. The *Selection* fields show the hexadecimal number for the color and a color sample.

6. Click *OK* when you are satisfied with the color.

Verilog AMS Signal Options

To customize Verilog AMS waveforms:

1. Choose *Edit – Preferences* from the menu bar, expand *Signal Options* in the list on the left, and select *Verilog AMS*.

2. Choose how you want to display AMS branches, as follows:

   - Enable *Show Potential* to display potential quantities.
   - Enable *Show Flow (if available)* to display flow quantities.

3. Choose how you want to format potential and flow values, as follows:

   - Enable *Show Scale Factor and Unit* to use scalar multipliers and nature units, such as `-30.646mV`.
   - Enable *Show Floating Point* to use floating-point format, such as `-0.0306461`.

Keyboard Shortcuts

You can execute every SimVision command by using the menus and buttons, or by pressing a keyboard shortcut. For example, in the Waveform window, you can select a set of signals

and create a group by clicking *Group*,  , or by choosing *Edit – Create – Group*. However, you can also use the keyboard shortcut, `Control-G`, to create the group.

SimVision comes with two sets of keyboard shortcut definitions: a default set and a set that mimics Signalscan key bindings. However, you can define your own set of shortcuts and bind your own key sequences to any SimVision command.

 *Tip*

To see a list of your current keyboard shortcuts, choose *Help – Keyboard Shortcuts*. Whenever you change the shortcut definitions, the new definitions appear in the Help display.

To create your own keyboard shortcuts:

1. Choose *Edit – Preferences* from the menu bar, expand *Signal Options* in the list on the left side of the Preferences form, then select *Keyboard Shortcuts*.
2. Select the name of the menu in the *Menus* area of the window. When you choose a menu, its menu entries are listed in the *Commands* area of the window.
3. In the *Commands* area, select the menu entry for which you want to define a shortcut.
4. In the *Keyboard shortcut* field, enter the key sequence that you want to bind to the command.
5. At any time, you can click *Reset to default bindings* to remove the shortcuts that you have defined, or click *Reset to Signalscan bindings* to define shortcuts that mimic Signalscan.

Toolnet Cross-Probing

If you are using the Synplicity® Synplify tool, you can use Toolnet to cross-select objects in SimVision and Synplify windows.

Note: Toolnet cross probing is not supported in Verilog-XL.

To enable Toolnet cross probing:

1. Choose *Edit – Preferences* from the menu bar, expand *Signal Options* in the list on the left side of the Preferences form, then select *Toolnet Cross Probing*.
2. Enable *Enable cross probing of Synplicity Tools* to turn on cross probing.
3. For Verilog designs, enter the path to the design snapshot that you have loaded into SimVision.

Simulation Settings

To specify simulator settings:

1. Choose *Edit – Preferences* from any SimVision window, then choose *Simulation settings* from the list on the left side of the window.
2. Enable *Prompt before Reinvoke* if you want SimVision to prompt you for command-line options when you reinvoke the simulation. If this button is disabled, SimVision reinvokes the simulator with the original command-line options that you used.
3. Enable *Respond to simvision commands sent from a simulator* if you want to accept commands sent to SimVision from the simulator. When this command is disabled, SimVision ignores commands sent by the simulator. You might want to disable a command for security reasons if you are accessing a simulation that is running on another system.
4. By default, SimVision filters objects from methodology libraries, such as OVM. Disable the *Enable Standard Methodology Filtering* option if you want to see this objects.
5. By default, SimVision displays the class hierarchy from the top of the class hierarchy. Enable the *Display Class Hierarchy starting from derived classes* option if you do not want to see the top level of the hierarchy.
6. From the *SystemC Debugger* drop-down menu, choose the debugging mode that you want to use, either *DDD integration* or *native SimVision gdb debugger*.
7. From the *Quick Thread Debugging* menu, choose whether you always want to use Quick Threads with the PLI Debugger, or whether you want to use Quick Threads only when SystemC is present.
8. By default, SimVision continues running in post-processing mode when you type `exit` at the simulator prompt in the Console window. Enable *Exit SimVision when exiting simulator* if you do not want to continue running SimVision when you exit the simulator.

Waveform Window

To customize the display of waveforms:

1. Choose *Edit – Preferences* from the menu bar and select *Waveform Window* from the list on the left side of the Preferences window.
2. Enable *Waveform banding* if you want the background of the Waveform window to alternate between bands of gray and black. Waveform banding can help you visually

align signal names and values with their corresponding waveforms. Disable this option if you want the background to be all black.

3. Enable *Search Toolbar on by default* if you want to display the toolbar that lets you search for signal names, groups, markers, expressions, and values.
4. Enable *Show Times on Markers* if you want the current simulation time to appear in markers. You can choose not to display simulation times if you want to keep the size of markers as small as possible.
5. Enable *Show Times on Cursors* if you want the current simulation time to appear in cursors. You can choose not to display simulation times if you want to keep the size of cursors as small as possible.
6. Disable *Snap Cursor to Signal Transition* if you do not want the cursor to jump to the nearest signal transition as you move it through simulation time.
7. Enable *Zoom Out Full on Initial Add* if you want the initial view of the simulation database to include the entire simulation timeline. By default, SimVision displays a portion of the timeline. The amount of time displayed depends on the total amount of simulation time.
8. Enable *Enable Undo When Adding Signals* if you want to be able to click *Undo* to remove any signals that you have just added. This button is disabled by default, because an undo operation after you have added many signals can take a long time.
9. From the *Signal Radix* drop-down menu, choose the radix that you want to use when adding signals to the Waveform window. By default, the Waveform window displays values in the radix in which they were recorded to the simulation database.
10. Select one of the following actions to take when you double-click on a waveform:

    - `No action`
    - *Go to Cause*
    - *Display in Trace Signals Sidebar*
    - *Trace X in Trace Signals Sidebar*
    - *Tcl Command in the SimVision Console*
    - *Tcl Command in the Simulator Console*

    If you choose *Tcl Command in the SimVision Console*, you must specify a SimVision command to execute. If you choose *Tcl Command in the Simulator Console*, you must specify a simulator command to execute.

Waveform Display

To set display options in the Waveform window:

1. Choose *Edit – Preferences* from the menu bar, expand *Waveform Window* in the list on the left side of the Preferences form, and choose *Display*.
2. Set the *Sequence Time Width* to the number of pixels you want to use between ticks when you expand sequence time. The default is 30 pixels.
3. Set the *Waveform height* for the number of pixels you want to use for the height of a digital waveform. The default is 12 pixels.
4. Set the *Waveform space* to the number of pixels you want to use between waveforms. The default is 6 pixels.
5. Set the *Transaction height* to the initial height of a transaction. The default is 3 times the height of a digital trace. The height of transactions is adjustable after they are displayed.
6. Set *Dynamic Arrays height* to the number of elements you inititally want to see when displaying dynamic arrays. By default, no elements are displayed. The height is adjustable after they are displayed.

Analog Waveform

To set the height of analog waveforms:

1. Choose *Edit – Preferences* from the menu bar, expand *Waveform Window* in the list on the left side of the Preferences form, then choose *Analog*.
2. Set the *Analog height* to the initial height of an analog waveform. The default is 3 times the height of a digital trace. The height of analog waveforms is adjustable after the waveform is displayed.
3. From the *Analog Format* menu, choose the format that you want to use when displaying analog signal values—either *Linear interpolation* or *Sample/Hold*.
4. Set the *Analog to digital threshold low* and *Analog to digital threshold high* fields to specify how analog signals are converted to digital signals. The low threshold specifies the value at which a signal is converted to 0; the high threshold specifies the value at which a signal is converted to 1.
5. Enable *Use data sampling* to reduce the number of data points when drawing analog waveforms. This improves performance but can remove interesting features of the waveform, such as spikes, when waveforms are zoomed out. This option is disabled by

default, and analog waveforms are drawn at each data point. This provides the most accuracy at all zoom levels, but can be slow when drawing waveforms with many data points.

6. Disable *Use automatic colors for Analog signals* if you do not want SimVision to assign a unique color to each analog signal that you add to the Waveform window.

Waveform Keyboard Shortcuts

SimVision defines a set of keyboard shortcuts for the buttons and menu choices that manage the waveform display, such as zooming in and out, and scrolling the display. You can define your own keyboard shortcuts for these commands.

Tip

To see a list of your current keyboard shortcuts, choose *Help – Keyboard Shortcuts*. Whenever you change these shortcut definitions, the new definitions appear in the Help display.

To define keyboard shortcuts for waveform commands:

1. Choose *Edit – Preferences* from the menu bar, expand *Waveform Window* in the list on the left side of the Preferences form, then choose *Keyboard Shortcuts*.
2. Place the cursor anywhere in the text field for the definition you want to change, and press the `Del` or `Back Space` key. This removes the old definition.
3. With the cursor still in the text field, press the desired key combination. The names of the keys that you press appear in the text field.
4. At any time, you can revert back to the default key bindings by clicking *Reset to default bindings*, or replace the key bindings with the Signalscan bindings by clicking *Reset to Signalscan bindings*.

Design Browser

To set general Design Browser preferences:

1. Choose *Edit – Preferences* from any SimVision window, then choose *Design Browser* from the list on the left side of the window.
2. Disable *Incremental Signal Filtering* if you do not want the Design Browser window to update the signal list incrementally. When this option is enabled, the signal list updates

each time you enter a character of the search string in the signal list *Filter* field. When this option is disabled, the signal list is updated only after you press `Return` in the *Filter* field.

3. Enable *Show Module/Unit Names* if you want to display the module or design unit name next to each scope. By default, this option is off, and only the scope name is displayed.

   Note: *Show Module/Unit Names* does not apply to the ***e*** unit tree. The ***e*** unit tree displays the short names for units at all times.

4. Disalbe *Show OVM Verification Hierarchy* if you do not want to see OVM objects in the Design Browser. This option is enabled by default.

5. Choose when you want to display signal values: *Never*, *Always*, and *When connected to simulator*.

Scope View

To set scope view preferences:

1. Choose *Edit – Preferences* from any SimVision window, expand *Design Browser* from the list on the left side of the window, then choose *Scope View*.

2. Choose the types of Verilog, VHDL, and SystemC, and ***e*** objects you want to display in the scope view.

3. Enable *Icons* to display icons that indicate object type, or disable this option to display only the object name.

4. Enable *Tree View* to display the scopes of the design hierarchically. Disable this option to display only the objects at the current level of hierarchy.

5. Choose whether you want to sort the objects in the scope view by *Name* or *Declaration type*.

Note: You might need to scroll the Preferences window to see the following option.

6. Enable *Track between target source browser and target watch list* if you want the Source Browser to display the source code for the scope that you select in the Design Browser sidebar.

Signal List

To set signal list preferences:

1. Choose *Edit – Preferences* from any SimVision window, expand *Design Browser* from the list on the left side of the window, then choose *Signal List*.
2. Set the signal filter defaults for your design language by choosing the types of objects you want to display in the signal list. You can override these defaults by enabling and disabling the signal list filter buttons in the Design Browser window.
3. Select the radix that you want to use when displaying signal values.
4. Enable *Show Strength in Values* if you want to display the signal strength with the signal values.
5. Enable *Filter using regular expressions* if you want to include wildcard characters in the filter string.

Source Browser

To set Source Browser options:

1. Choose *Edit – Preferences* from any SimVision window, and choose *Source Browser* from the list on the left side of the window.
2. Enter the command to invoke the editor in the *Editor Command* field, using substitution strings to specify a file (%F) and a line number (%L), if your editor accepts those arguments. For example, to set `vi` as your editor, enter the following command:

   ```
   xterm -e vi %F
   ```

   Click *Default Editor* to reset the editor to the default.
3. Enter values for the following preferences:

   - In the *Source Browser Tab Size* field, enter the size of a tab stop.
   - In the *Maximum History Count* field, enter the number of scopes you want to save in the history list. See "Navigating the Design Hierarchy" on page 65 for information on using the history list.

   These settings take effect when you click *OK* or *Apply.*
4. In the *On Signal Double Click* field, choose the behavior you want when you double-click on a signal in the Source Browser. The drop-down menu provides several

options, such as displaying the signal definition, or sending the signal to a specific window.

If you choose *Run a Tcl Command in the SimVision Console*, you must specify a SimVision command to execute. If you choose *Run a Tcl Command in the Simulator Console*, you must specify a simulator command to execute.

5. In the *Drag and Drop* field, choose the behavior you want when you drop a signal into the Source Browser. The drop-down menu provides several options, such as displaying the signal definition, or sending the signal to a specific window.

   If you choose *Run a Tcl Command in the SimVision Console*, you must specify a SimVision command to execute. If you choose *Run a Tcl Command in the Simulator Console*, you must specify a simulator command to execute.

6. Disable the *Show Popup Tooltips on Signals in the Source Code* if you do not want the Source Browser to display the signal name, value, and probe status when you hover the cursor over a signal name. This information appears in a tooltip in the source area of the window.

7. Disable *Highlight Syntax in the Source Browser* if you do not want to display keywords, comments, and output strings in different colors. You can specify these colors in the Source Browser Colors Preferences tab.

Note: You may need to scroll or resize the window to see the following options.

8. Disable *Show Signal Strength in Values* if you do not want to see the strength of a signal value.

9. From the *Value Radix* drop-down list, select a radix for values displayed when you place the cursor over a signal. By default, the Source Browser displays values in the radix in which they were recorded.

Source Browser Colors

You can change the colors used by the Source Browser to display text elements such as comments, keywords, and strings in the source code.

To change the colors used by the Source Browser:

1. Choose *Edit – Preferences* from the menu bar, then select *Colors* from the list on the left side of the Preferences form, below *Source Browser*.

2. From the *Item* drop-down menu, select the text element whose color you want to change.

3. Click *Color* to open the *Color* menu. Choose a color from the menu, click *Default* to reset the text element to its default value, or click *More* to create a custom color with the Select a New Color form.

   When you select the new color, SimVision updates the sample in the Preferences form to show how the Source Browser will look when you apply your changes.

4. At any time, if you decide that you do not want to use the preferences you have set, click *Revert to Default Colors*.

Source Browser Files

You can specify the file types that the Source Browser recognizes when highlighting syntax.

To specify Source Browser file types:

1. Choose *Edit – Preferences* from the menu bar, then select *Files* from the list on the left side of the Preferences form, below *Source Browser*.
2. Click a row in the *Extensions* column, then add or remove a file extension.

Measurement Window

The Measurement window displays a table of measurements for selected signals. By default, the Measurement window includes no measurements, only the signal or variable name.

To choose the types of measurements displayed by default in a Measurement window:

1. Choose *Edit – Preferences* from the menu bar, then select *Measurement Window* from the list on the left side of the Preferences form.

   A list of *Available measurement* types is shown on the left, and a list of *Default measurements* is shown on the right.

2. Select the type of measurement you want to include in your Measurement window from the list of *Available measurements*, then click the right arrow to add your selections to the list of *Default measurements*. Select the type of measurement you want to remove from the list of *Default measurements*, then click the cross-hatch button.

   Buttons to the right of the *Default measurements* list let you rearrange the list, as follows:

    Move the selected measurement up in the list.


Move the selected measurement down in the list.


Remove the selected measurement from the list of *Default measurements*.

In the Measurement window, *Default measurements* are displayed from left to right, beginning with the first measurement in the list. Therefore, moving a measurement up in the list has the effect of moving it to the left in the Measurement window. Conversely, moving the measurement down in the list, has the effect of moving it to the right in the window.

Schematic Tracer

To set Schematic Tracer options:

1. Choose *Edit – Preferences* from any SimVision window, and choose *Schematic Tracer* from the list on the left side of the window.
2. Choose when you want the Schematic Tracer to show signal values, as follows:
   - *When connected to simulator*
   - *Always*
   - *Never*
3. Enable *Show RTL Logic when available* to display logic elements in all new Schematic Tracer windows, by default.
4. Enable *Show Cell Shapes, if available*, to display ASIC library cells using the shapes that you have specified in a cell map file. If you enable this option, you must also specify a filename in the *Cell Shapes Mapfile* field.
5. Enable *Fill instances added to the schematic* if you want the Schematic Tracer to automatically add any subscopes, blocks, and internal signals for the instances that you add to the schematic.
6. Enable *Fill parent module after an Ascend operation* if you want the Schematic Tracer to automatically add the subscopes, blocks, and internal signals of any parent modules encountered during an ascend operation.
7. By default, the Schematic Tracer highlights any logic it has added as the result of a fill operation. Disable *Highlight new logic after a fill operation* if you do not want this logic highlighted.

8. Enable *Show selected scope in the status bar area* if you want the Schematic Tracer to display the full name of the selected scope in the status area of the window.
9. Enable *Show internal pins when using cell mapping* if you want the Schematic Tracer to display these pins.
10. Disable *Show thumbnail image of schematic in panner* if you do not want to see the thumbnail image in the panner. When you disable this preference, the panner is blank, except for the control box. You can still resize the box to zoom in and out, and move the box to scroll the display.
11. Disable *Center and Zoom after adding objects* if you do not want the Schematic Tracer to zoom in on the objects that you add.
12. In the *Cell Shapes Mapfile* field, specify the name of an ASIC library cell map file. This file defines the shapes that the Schematic Tracer uses when displaying library cells.
13. From the *Double Click* menu, choose the action you want to take when you double-click on a schematic element. If you choose *Tcl Command in the SimVision Console*, you must specify a SimVision command to execute. If you choose *Tcl Command in the Simulator Console*, you must specify a simulator command to execute.

Schematic Tracer Colors

To choose the colors used in the Schematic Tracer window:

1. Choose *Edit – Preferences* from any SimVision window. If necessary, expand the *Schematic Tracer* entry on the left side of the window, then choose *Colors*.
2. From the *Item* drop-down list, choose the type of schematic element whose color you want to change.
3. Click *Color* to open the *Color* menu. Choose a color from the menu, click *Default* to reset the text element to its default value, or click *More* to create a custom color with the Select a New Color form.

   When you select the new color, SimVision updates the sample in the Preferences form to show how the SchematicTracer will look when you apply your changes.
4. At any time, if you decide that you do not want to use the preferences you have set, click *Revert to Default Colors*.

Memory Viewer

To set the Memory Viewer preferences:

1. Choose *Edit – Preferences* from any SimVision window, and choose *Memory Viewer* on the left side of the window.
2. From the *Radix* drop-down menus, choose the radix that you want the Memory Viewer to use when displaying values and addresses. Choose *As Recorded* to display the values and addresses in the radix in which they were recorded in the simulation database.
3. Enable *Size to fit* to let the Memory Viewer adjust the width and number of columns used to display memories, or disable that button and specify a fixed number of columns.
4. From the *Font Size* menu, choose to display the memory values in *Small*, *Medium*, or *Large* font.
5. In the *Double Click* field, specify the action that you want the Memory Viewer to take when you double-click on a memory element. The *Edit Deposit* action is the default, which lets you specify a value to deposit at the memory location.

   If you choose *Tcl Command in the SimVision Console*, you must specify a SimVision command to execute. If you choose *Tcl Command in the Simulator Console*, you must specify a simulator command to execute.

Memory Viewer Colors

To choose the colors used in Memory Viewer windows:

1. Choose *Edit – Preferences* from any SimVision window. If necessary, expand the *Memory Viewer* entry on the left side of the window, then choose *Colors*.
2. From the *Item* drop-down menu, select the type of value whose color you want to change:

   - *Value has changed*—Choose the foreground and background colors to indicate that the value changed during the current sample time, but its ending value is the same as its starting value.
   - *Value is changing*—Choose the foreground and background colors to indicate that the value is changing during the current sampled time.
   - *Value is different*—Choose the foreground and background colors to indicate that the value changed during the current sample time, and its ending value is different from its starting value.

3. Click *Color* to open the *Color* menu. Choose a color from the menu, click *Default* to reset the text element to its default value, or click *More* to create a custom color with the Select a New Color form.

   When you select the new color, SimVision updates the sample in the Preferences form to show how the Memory Viewer will look when you apply your changes.

4. At any time, if you decide that you do not want to use the preferences you have set, click *Revert to Default Colors*.

Simulation Cycle Debug

Note: The Simulation Cycle Debugger is not available in Verilog-XL.

To set Simulation Cycle Debugger preferences:

1. Choose *Edit – Preferences* from any SimVision window, and choose *Simulation Cycle Debug Options* from the list on the left side of the window.
2. Specify the number of process or behavioral blocks you want to display in the Simulation Cycle Debugger. The maximum number is 100, because displaying more items can result in poor performance.

Transaction Explorer

The Transaction Explorer preferences tabs let you customize the behavior of the TxE graphical user interface.

For more information, see Chapter 4, “Graphical User Interface,” in the *Transaction Explorer Reference Manual*.

26

Customizing Toolbars

SimVision lets you choose the toolbars and buttons that you want to display in any SimVision window. You can also create new toolbars and buttons to perform functions that are not already available through the SimVision toolbars.

Adding and Removing Toolbars and Buttons

Each window type has its own set of toolbars and buttons. However, the basic procedure for adding and removing toolbars and buttons is the same, regardless of the window type.

 Video

Adding and Removing Toolbars and Buttons

To add or remove a toolbar or button:

1. Choose *View – Toolbars – Customize* from any SimVision window. This opens the Customize Toolbars form.

   Enable or disable the check mark for a toolbar to add or remove it in the window.

 *Tip*

You can also add and remove toolbars by right-clicking over a blank space or separator in the toolbar area, and then enabling and disabling toolbars in the pop-up menu.

2. Highlight a toolbar in the *Toolbars* list, and the Customize Toolbars form displays the commands available in that toolbar, including the buttons that represent those commands.

   Enable or disable the check mark for a command to add or remove its button in the toolbar for that window. At any time, you can return the toolbar's button settings to their default values by clicking the *Restore Default* button.

3. At any time, you can restore all toolbars to their default settings by clicking *Restore all Toolbars*. If a toolbar that you have defined is in one of the default toolbar locations, your toolbar will be moved.

When you modify the toolbars and buttons for a window type, SimVision applies those changes to the current window and to any new windows of that type. Any other windows of that type that are already opened are not affected by your changes.

Creating a Custom Toolbar

The amount of customization that you can perform on the SimVision toolbars is limited. That is, you can enable and disable buttons on the toolbar, but you cannot add or delete buttons and you cannot rearrange their order in the toolbar. However, you do have complete control over the contents and layout of the toolbars that you create.

 Video

Creating Toolbars and Buttons

To create a toolbar:

1. Click *New Toolbar*, and SimVision adds a *User Toolbar* to the list of toolbars. At this point, the toolbar is empty; no commands have been defined.

   You can create additional toolbars by clicking *New Toolbar*. Additional toolbars are assigned the names *User Toolbar_1*, *User Toolbar_2*, and so on.

2. Click *New Button* to define a button for the toolbar. This opens the Create a New Button form.

3. Click *Icon*. This displays a menu of icons.

   Choose an icon in one of the following ways:

   - Choose an icon from the menu.
   - Click *No Icon* to specify that you do not want to use an icon for the button. In this case, the label is used for the button.
   - Click *More* to use your own GIF file for an icon. If you use your own GIF file, SimVision sizes the icon for you.

4. In the *Label* field, enter the string you want to display below the icon when the *Toolbar Style* is set to *Text only* or *Both* in the General Options tab of the Preferences form.

5. In the *Help Text* field, enter the string you want to display in the status area of the window and in a tooltip when the mouse cursor hovers over the button.
6. Enable *Button is sensitive to selection* if this button is enabled when objects have been selected in a SimVision window. Otherwise, the button is disabled.
7. Enable *Button is a toggle button* if you want this button to toggle. That is, the button appears to push in when the user clicks it once, then appears to pop out when the user clicks it again. Otherwise, this button appears to push in and pop out with a single mouse click.

   Note: You can set this option only at the time the button is created. After that, you cannot change this option setting.
8. In the *Script* drop-down list, choose the console that you want to send the button commands to, as follows:

SimVision	Sends the commands to the *SimVision* console, which accepts SimVision commands, described in the *SimVision Command Language Reference*.
Simulator	Sends the commands to the *simulator* console, which accepts simulator commands, described in the *Help* for your simulator.
gdb	Sends commands to the *gdb* console, which accepts `gdb` debugger commands. See the *NC-SC Simulator User Guide* for information on GDB support.

Tip

You can send Tcl/Tk commands to the *SimVision* console and to the *simulator* console for the Incisive simulator. You cannot send Tcl/Tk commands to the *gdb* console or to the *simulator* console for the Verilog-XL simulator.

9. In the text area for the *Script* field, enter the commands that you want to send to the specified console. For example, if you want to send a `describe` command to the simulator when you click the button, enter the following command in the text area:

   ```
   describe test_drink.top.dispense test_drink.top.vending.current_state
   ```

You can use the substitution strings listed in Table 26-1 on page 328 to pass arguments to your commands.

Table 26-1 Substitution Strings for Command Scripts

`%b`	Button name
`%c`	All selected scopes
`%i`	All selected signals
`%n`	Window name
`%s`	All selected items
`%t`	Window type
`%{prompt “string”}`	Opens a dialog box and prompts the user for a string. The string is passed as an argument to the command.

For example, the following Tcl command uses `%n` to pass the window name to the `puts` command:

```
puts "%n"
```

You could also prompt the user for the name of a signal, then pass that name to the `describe` command, as follows:

```
describe %{prompt “Signal”}
```

10. Click *OK* to add the button to the Commands area of the Customize Toolbars form.

11. Click *Close*. The button is added to the toolbar.

Tip

If the toolbar does not appear in the window, open the Customize Toolbars form again, and make sure the toolbar is enabled in the list of toolbars.

Adding a Custom Toolbar to Other Windows

The toolbars that you create can be added to any SimVision window.

To add your toolbars to other SimVision windows:

1. In the window to which you want to add the toolbar, choose *View – Toolbars – Customize*. This opens the Customize Toolbars form for that window type.
2. Click *New Toolbar*, and SimVision adds the user toolbars that you have created to the Toolbars list.
3. Select the user toolbar that you want to add to the window. The buttons that you created for that toolbar are listed in the Commands area of the form.
4. Enable the check mark for the buttons that you want to display in the toolbar, then click *Close*. The toolbar and buttons now appear in the window.

Writing a Window-Independent Script

SimVision commands are window-specific. That is, a command to invoke a SimVision menu entry must be invoked from a particular window, such as a Design Browser or Waveform window. For example, suppose you want to create a button to save the user environment and then reinvoke the simulator. If you wrote the script in the following way, it would be tied to a particular window:

```
browser invoke -using “Design Browser 1” “File>SaveCommandScript”
browser invoke -using “Design Browser 1” “Simulation>ReinvokeSimulator”
```

As written, this script will work only if a Design Browser window named “Design Browser 1” exists.

However, there is no way to guarantee that a particular window exists when the user runs the script. You need to write the script so that it can run in any window.

Here are two ways to do this:

- You can use the Tcl `set` command to return the type of the current window, as follows:

  ```
  set type [window type %n]
  ```

  The %n argument passes the name of the current window to the `window type` command. This command returns the type of the window, such as `browser` or `waveform`.

The window type is also the name of the command that you use to invoke the *Save Command Script* and *Reinvoke Simulator* commands. You can use the value of the `type` variable as the command name. For example:

```
$type invoke -using %n "File>SaveCommandScript"
$type invoke -using %n "Simulation>ReinvokeSimulator"
```

- You can use the `window` command, as follows:

  ```
  window invoke %n -type menu "File>SaveCommandScript"
  window invoke %n -type menu "Simulation>ReinvokeSimulator"
  ```

  The `window` command can invoke a command for any window type, specified by the %n argument.

Modifying Custom Toolbars

You can modify the toolbars that you create, as follows:

- Remove the check mark for a button to remove it from the toolbar. This does not delete the button definition. The button is still available to other windows that contain this toolbar.
- Click *Delete Button* to delete a button, if it does not appear in another toolbar. If the button is enabled and it appears in other toolbars, *Delete Button* disables the button. If you try to delete a disabled button that appears in other toolbars, SimVision displays an error message.
- Select a button and click *Edit Button* to modify the button definition in the Edit Button form. When you edit a button, it is changed in every window in which it appears.
- Select a button and click *Move Up* or *Move Down* to rearrange the order in which the buttons appear in the toolbar.

Index

Numerics

0-delay error
    locating with Simulation Cycle Debugger 153
-64BIT
    simvisdbutil command option 118
    simvision command option 44

A

abstracted RTL, schematic elements 186
-access +w, elaborator option
    forcing and depositing values 139
active drivers, tracing in Trace Signals sidebar 201
Add a Bookmark button
    in the Design Browser 107
    in the Schematic Tracer 180
    in the Source Browser 77
Add button 33
    Create Force form 140
    Create Monitor form 144
    Create Probe form
        NC simulator 126
        Verilog-XL 126
    in containing window
        from Design Browser sidebar 89
        from Design Search sidebar 95
    Properties window, Groups tab 233
    Register window 261, 297
    Waveform window 213
Add Variables button 166
-ADDINDICES, simvisdbutil command option 118
addition operator (+), in Schematic Tracer 187
aggregate signal
    expanding and collapsing
        in Design Browser window 102
        in Waveform window 214
alias 150
always statement, in Schematic Tracer 184
analog
    changing color of waveforms 275
    navigation buttons 271
    overlay group 273
    signal
        depositing a value 141
        forcing the value of 140
        in Waveform window 269 to ??
        measuring values 303
AND gate, in Schematic Tracer
    abstracted RTL element 186
    gate primitive 188
annotation toolbar, in Register window 299
API applications, debugging 159 to 168
API source file name
    in the Source Browser 162
-APPEND_KEY, simvision command option 44
-APPEND_LOG
    simvisdbutil command option 118
    simvision command option 44
Arrange menu
    Reverse Order 305
    Sort 305
ASIC library cells
    viewing in Schematic window 194
Assertion Browser 20
assertion, show or hide button
    in the sidebar 90
    in the signal list 101
attributes, of transaction streams 280
automatic save and restore 54
Average Value measurement 303

B

Baseline - Cursor measurement 303
baseline cursor 243
behavior exection phase 153
bit-select, forcing the value of 140
Bitwise shift left operator (<<), in Schematic Tracer 187
Bitwise shift right operator (>>), in Schematic Tracer 187
block
    schematic element 183

scope view icon 84
bookmarks
in the Design Browser 106
in the Schematic Tracer 180
in the Source Browser 77
Bookmarks sidebar 29
opening in the Design Browser 107
opening in the Schematic Tracer 180
opening in the Source Browser 77
Bookmarks tab, Properties window 21
breakpoint 129
in the debugging environment 53
deleting, in Properties window 138
enabling and disabling
in Properties window 138
with Source Browser 72
function
setting in API code 162
line
setting in API code 162
setting with NC simulator 133
setting with the Source Browser 71
setting with Verilog-XL 133
memory object 265
options 137
setting in API code 162
setting in Properties window 138
signal
setting with NC simulator 134
setting with Verilog-XL 134
time
setting with NC simulator 129
setting with Verilog-XL 131
BUF3S, in Schematic Tracer 188
BUF3SL, in Schematic Tracer 188
BUFGATE, in Schematic Tracer 188
bus 235
creating 235
with analog signals 274
button
adding and removing 325
user-defined 27, 326

C

call stack
viewing in the Source Browser 74
viewing SystemC/C++/C 165
Call Stack sidebar 29
for API debugging 75
CDS_SHM_COMPATIBILITY environment variable 110
cell map file
creating 189
format of 189
handling errors in 195
translating .lib file 193
Cells menu
Generate Cell Template 192
Show Cell mapfile Warnings 195
Translate .lib File 193
Class Hierarchy sidebar 29
class, scope view icon 83
Click to Add button 205
Close Database button 123
Close Panner button 175
Collapse Modules button 172
Collapse Sidebar button 29
Color menu
Register window 300
Source Browser preference 320
colormap, specifying on the simvision command 46
command script
contents of 59
for simulator connection 56
for post-processing 57
.simvisionrc 61
specifying on the simvision command line 45
comments, in the Waveform window 221
Comparison Results sidebar 29
comparison, creating in the Waveform window 242
component instance, scope view icon 84
component instantiation statement, in Schematic Tracer 184
-COMPRESS
simvisdbutil command option 118
simvision command option 45
concurrent
assertion scope view icon 84
procedure call
in Schematic Tracer 184
scope view icon 84
signal assignment
in Schematic Tracer 184
scope view icon 84
condition breakpoint 135
condition expression 237
-CONNECT, simvision command

option 45
connectivity information, in simulation database 49
Console window 20, 25, 42
sourcing a command script 59
continuous assignment, in Schematic Tracer 184
contributing signals, adding to Waveform window 214
Create a New Button form 326
Create Force form 140
Create Monitor form 144
Create new group button 232
CSV database
batch conversion 117
format 115
-CSV, simvisdbutil command option 118
current execution point, in Source Browser 74
cursor 243 to 250
in the debugging environment 53
delta time 252
setting the location of in Measurement window 304
Cursor Control toolbar 27
cursor mode, in Waveform window 224
Cursors tab, Properties window 20
Customer Service, Help menu 35
-CVTMVL9, simvisdbutil command option 118

D

database
batch translation utility 117
closing 123
converting to SST2 format 113
creating
with NC simulator 111
with Verilog-XL 112
in the debugging environment 54
in Design Browser sidebar 81
displaying information about 121
exporting 115
formats 109
open command in SimVision command script 61
opening 48, 113
require command in SimVision command script 61
statement trace information in 76
Databases tab, Properties window 20
debugging environment, saving and restoring 53
‘define macros, in the Source Browser 69
-DELAY, simvisdbutil command option delay, simvisdbutil command option] 118
Delete button
Properties window
Aliases tab 150
Breakpoints tab 138
Forces tab 141
Probes tab 128
Schematic Tracer 172
Waveform window
deleting comments 222
deleting groups 234
deleting time ranges 255
delta cycle 153
phases 153
running the simulation 157
delta time, in the Waveform window 252
deposit
in the debugging environment 53
in Memory Viewer 265
Design Browser button 19, 97
Design Browser sidebar 28, 81 to 90
finding transaction streams in 280
Design Browser tab, Preferences form
scope view 317
signal list 318
Design Browser window 19, 97 to 108
closing a database in 123
design file (.dsn) 109
design hierarchy
accessing
when connecting to a simulation 47
when invoking the simulator with SimVision 40
when opening a simulation database 49
navigating
in the Design Browser sidebar 81
in the Source Browser 65
Design Search sidebar 29, 92 to 95
adding signals to the Waveform window 213
finding transaction streams 280
Disconnect and Terminate from C/C++ Debugger button 160

Disconnect button 42
Display Parent Module button 171
Display Values button 69
-DISPLAY, simvision command option 45
division operator (/), in Schematic Tracer 187
drag and drop operation 33
driving logic
  tracing in the Schematic Tracer 176
  tracing in the Trace Signals sidebar 200
    active drivers 201
.dsn file 109

E

e language objects, scope view icon 84
Edit menu 31
  Ascend 171
  Collapse Modules 172
  Create
    Analog Overlay 273
    Bus 235
      analog signals 274
    Comment 221
    Comparison 242
    Condition 240
    Marker 249
    Mnemonic Map 219
  Delete Modules 172
  Explode 87
  Fill Modules 171
  Go To Address 264
  Go To Line 68
  Modify
    in Memory Viewer 265
  Preferences
    Design Browser 108, 316
      Scope View 317
      Signal List 318
    General Options 55, 308
    Measurement window 320
    Memory Viewer 323
    Schematic Tracer 321
      Colors 322, 323
    Signal Options 309
      Keyboard Shortcuts 312
      Toolnet Cross Probing 312
      Verilog 310
      VHDL 310
    Simulation Cycle Debug 324
    Simulation Settings 313
    Source Browser 318
      Colors 319
      Files 320
    Waveform window 313
      Keyboard Shortcuts 315, 316
  Search
    Memory Viewer 264
    Schematic Tracer 176
  Select All 27
  Text Search
    Console window 26
    Source Browser 67
Edit Source button 78
end of time phase 154
environment variable
  CDS_SHM_COMPATIBILITY 110
  LDV_SIMVISION_CONNECTIONS 51
  LDV_SIMVISION_PASSWORD_FILE 52
  SIMVISION_HOME 55
  SIMVISION_WORKDIR 48, 59, 66, 113, 114
  SIMVISIONOPTS 47
Epic database 20
  batch conversion 117
  opening 114
Equal to operator (=), in Schematic Tracer 187
error information, transaction
  viewing in Waveform window 282
event
  locating ordering error 153
  in sequence time 255
  source code 157
execution phase 153
Expand Sidebar button 29
Explore menu
  Add Contributing Signals 297
  Add Module Inputs 297
  Go To
    Cause 77
expression
  schematic element 185
Expression Calculator
  editing an expression in 241
Expression Calculator button 19
Expression Calculator window 19
-EXPRESSION, simvisdbutil command option 118
Expressions tab

Properties window 241
Expressions tab, Properties window 21

F

File menu
Close Database/Simulation
to close a database 123
Close Window 24
Create Database
NC simulator 111
to save event information 255
Verilog-XL 112
Export 115
Export Window 306
Load Memory 267
New 24
Assertion Browser 20
Clone 23
Measurement 20
Open Database 48, 113
to convert a database to SST2 format 114
Open Simulation 47
Open Source File 66, 162
to set a breakpoint 161
Print Window
Memory Viewer 268
Register window 302
Schematic Tracer 181
Simulation Cycle Debugger 158
Waveform window 229
Rename Window 25
Save Command Script
for post-processing 58
for simulator connection 56
Save Memory 267
Source Command Script 59
for simulator connection 57
-FILE, simvisdbutil command option 118
Fill Modules button 171
flip-flop, in Schematic Tracer 186
Follow into scope button, in Trace Signals sidebar 199
force
in the debugging environment 53
deleting from Properties window 141
releasing 141
setting 139
from Memory Viewer 265
from Properties window 140
format function 290
Format menu
Color
for analog waveforms 275
in Register window 300
Radix, in Schematic Tracer 174
Radix/Mnemonic, in Waveform window 218, 220
Shift 228
Symbol 275
Trace 240
function breakpoint 162
function, scope view icon 84

G

gate primitive, schematic element 188
gdb command, in user-defined button 327
gdb debugger, native-mode 160 to 167
General Options tab, Preferences form 308
generate statement
in Schematic Tracer 184
scope view icon 84
genevents function 288
glob rule 292
Go button, Source Browser 67
Greater than operator (>), in Schematic Tracer 186
Greater than or equal to operator (>=), in Schematic Tracer 187
grid, Waveform window 228
group 231 to 234
analog overlay 273
in the debugging environment 53
in Register window 301
Groups tab, Properties window 21

H

handle, GDB command 168
-HELP
simvisdbutil command option 118
simvision command option 45
Help button 34
Help menu
About SimVision 35
Cadence Help Library 35

Customer Service 35
Keyboard Shortcuts 33, 35, 312
KPNS 35
Related Products 35
SimVision Command Language Reference 35
SimVision User Guide 35
SimVision Windows 35
Tutorials 35
What's New 35
Hide Panner button 175
Hide Search button 67
Hide Sidebar button 30, 211
Hide Values button 69
high threshold, rise/fall time parameter 306
high value, rise/fall time parameter 306
hotkey 33
zoom functions, Waveform window 226
HSPICE list database 20
batch conversion 117
opening 113
HSPICE transient output database 20
batch conversion 117
opening 113

I

ignore, DBX command 168
inout port
filtering in Design Browser window 101
show or hide button
in the sidebar 90
signal list 101
input port
filtering in Design Browser window 101
show or hide button
in signal list 101
in the sidebar 90
-INPUT, simvision command option 45
insertion bar, Waveform window 222
insertion point, Measurement window 304
interface
in Schematic Tracer 184
scope view icon 83
internal signal
filtering in Design Browser window 101
show or hide button
in signal list 101
in the sidebar 90
Interrupt C Debugger window 165
Interrupt C/C++ Debugger button 160
INV3S, in Schematic Tracer 188
INV3SL, in Schematic Tracer 188
INVGATE, in Schematic Tracer 188
irun
compiling and linking an API application 159
elaborator options to use with SimVision 38
invoking with SimVision 38
options to use with SimVision 37

K

Keyboard Shortcuts tab, Preferences form 311, 316
-KEYFILE, simvision command option 45

L

latch, in Schematic Tracer 186
LDV_SIMVISION_CONNECTIONS environment variable 51
LDV_SIMVISION_PASSWORD_FILE environment variable 52
Less than operator (<), in Schematic Tracer 186
Less than or equal to operator (<=), in Schematic Tracer 187
Liberty file, translating to cell map format 193
line breakpoint 132
line number
searching for in Source Browser 67
-linedebug compiler option
for process breakpoints 136
Link button 248, 251
Load Local Variables button 166
Load Parameters button 166
loading logic
tracing in the Schematic Tracer 176
tracing in Trace Signals sidebar 198
Lock button 30
-LOGFILE
simvisdbutil command option 118
simvision command option 45
logic cone, tracing in Schematic Tracer 177
Logical AND, in Schematic Tracer 186

Logical NOT, in Schematic Tracer 186
Logical OR, in Schematic Tracer 186
low threshold, rise/fall time parameter 306
low value, rise/fall time parameter 306

M

marker 249
  changing
    in the Properties window 250
    in the Waveform window 250
  creating 249
  finding 251
  linking a Waveform window to 251
  new command in SimVision command script 61
  searching for 227
Markers tab
  Properies window 20
match function 290
Maximum Value measurement 303
Measurement window 20, 303 to 306
  customizing 320
Measurement Window tab, Preferences form 320
Memory Viewer 19, 259 to 268
  customizing 323
    color 323
Memory Viewer button 19, 260
Memory Viewer Colors tab, Preferences form 323
Memory Viewer tab, Preferences form 323
memory, forcing the value of 139
Minimum Value measurement 303
-MISSING, simvisdbutil command option 118
mnemonic map 218
  applying to a signal 220
  creating 219
Mnemonic Maps tab, Properties window 21
modport
  in Schematic Tracer 184
  scope view icon 83
module
  SystemC
    scope view icon 84
  Verilog
    scope view icon 83
module instance
  in Schematic Tracer 184
modulo operator (%), in Schematic Tracer 187
multiplication operator (*), in Schematic Tracer 187
mux, in Schematic Tracer 186

N

named block, scope view icon 84
NANDGATE, in Schematic Tracer 188
navigation buttons
  for analog waveforms 271
ncelab
  options to use with SimVision 38
ncsim
  invoking in post-processing mode 50
  invoking with SimVision 38
ncvhdl
  options to use with SimVision 37
ncvlog
  options to use with SimVision 37
New Page form 297
Next Condition button 240, 247
Next Edge button
  Schematic Tracer 173
  for transactions 281
  Waveform window 241, 247
Next Scope button 65, 162
-NOCOPYRIGHT
  simvisdbutil command option 119
  simvision command option 46
-NOLOG, simvisdbutil command option 119
-NOPLUGINS, simvision command option 46
NOR gate, in Schematic Tracer
  abstracted RTL 186
  gate primitive 188
-NORC, simvision command option 46
-NORESTORE, simvision command option 46
Not equal to operator (!=), in Schematic Tracer 187
NOT gate, in Schematic Tracer 186
Nutmeg database 20
  batch conversion 117
  opening 113
NXOR gate, in Schematic Tracer
  abstracted RTL 186

gate primitive 188

O

Open Database button 48, 113
to convert non-SST2 database 114
Open Database form 113
Open File button 160, 161, 162
Open Simulation button 47
Open Source button 66
Options button, Design Browser sidebar
scope view 91
signal list 91
OR gate, in Schematic Tracer
abstracted RTL 186
gate primitive 188
output port
filtering in Design Browser window 101
show or hide button
in the sidebar 90
signal list 101
-OUTPUT, simvisdbutil command
option 119
overlay group 273
-OVERWRITE, simvisdbutil command
option 119

P

package instance, scope view icon 84
Page menu
Clone 297
Rename 297
page, Register window
See register page
panner, Schematic Tracer 175
part-select, forcing the value of 140
password, for simulation processes 51
Peak-to-Peak measurement 303
-PERIOD, simvisdbutil command
option 119
plug-in application 34
disabling 46
plugin::application command
in .simvisionrc file 61
Point To Point Tracing form
for a pin and a cell 179
for a pin and a scope 178
for two pins 178
pop-up menu 32
post-processing mode 22, 37
Design Browser scope view 85
invoking SimVision in 50
saving and restoring a command script for 57
switching to 43
Power sidebar 29
Preferences form
Design Browser
Scope View 317
Signal List 318
General Options 308
Keyboard Shortcuts 311
Measurement window 320
Memory Viewer 323
Colors 323
Schematic Tracer 321
Colors 322
Signal Options
Toolnet Cross Probing 312
Verilog AMS 311
Simulation Cycle Debug 324
Simulation Settings 313
Source Browser 318
Colors 319
Files 320
Transaction Explorer 324
Waveform window
Keyboard Shortcuts 316
preferences set command, in SimVision command script 60
Previous Condition button 240, 247
Previous Edge button
Schematic Tracer 173
for transactions 281
Waveform window 241, 247
Previous Scope button 65, 162
primary cursor 244
choosing in Waveform window 246
setting in the time toolbar 30
tracking signal changes with 247
Print form
Memory Viewer 268
Register window 302
Schematic Tracer 181
Simulation Cycle Debugger 158
Waveform window 229
-PRIVATE, simvision command option 46
probe
creating in the Waveform window 213

in the debugging environment 53
enabling and disabling 127
in Properties window 127
memories 265
setting 125
with NC simulator 125
with Source Browser 73
with Verilog-XL 126
statement trace information 76
Probe button 128
probed signals, show or hide button
in the Design Browser sidebar 90
in the Design Browser signal list 102
procedure statement, scope view icon 84
process
breakpoint 136
execution phase 153
statement in Schematic Tracer 184
program block, scope view icon 83
Properties button 19, 21
Properties window 19
Aliases tab 150
Bookmarks tab
opening in the Design Browser 107
opening in the Schematic Browser 181
opening in the Source Browser 78
Breakpoints tab 138
Databases tab
closing a database in 123
Expressions tab 241

Q

Qsim database 20
batch conversion 117
opening 114
-QUIET, simvisdbutil command option 119

R

-RADIX
simvisdbutil command option 119
radix
in Create Monitor form 144
in Design Browser 101
in Register window 298
in Schematic Tracer 173
in Signal List Preferences form 318
in Source Browser Preferences form 319
in Waveform window 218
-RANGE, simvisdbutil command option 119
reference line 277
register, forcing the value of 140
Register button 19
register page 295
adding objects to 297
annotating 299
managing 296
printing 302
Register window 19, 295 to 302
displaying at startup 46
-REGISTER, simvision command option 46
regular expression 292
rematch function 290
restore.tcl command script 56
restore.tcl.svcf command script 56
Rise/Fall Time measurement 303
Rise/Fall Time parameters 306
RMS Value measurement 303
Run button 146
run-time driver, VHDL 143

S

scale, of analog waveforms 277
Schematic Tracer 19, 169 to 181
customizing 321
customizing color 322
displaying at startup 46
mapping ASIC library cells 189 to 196
tracing paths
from point to point 177
with theTrace Signals sidebar 206
Schematic Tracer button 19, 169
Schematic Tracer Colors tab, Preferences form 322
Schematic Tracer tab, Preferences form 321
-SCHEMATIC, simvision command option 46
scope
searching for 92
selecting 89
Scope Up button 65
scope view

Design Browser Sidebar Options
SystemC 91
filtering 87
icons 83
options 90
preferences 108
Scope View Design Browser tab, Preferences form 317
scrollable region
in Design Browser window 105
in Waveform window 216
Search Down button
Console window 26
Source Browser 67, 88
Waveform window 227
Search Up button
Console window 26
Source Browser 67, 88
Waveform window 227
security
of simulation processes 51
Select a New Color form 275
Register window 300
select operation 32
Send To toolbar 23, 27
Send Trace to Containing Window button 204
-SEQUENCE
simvisdbutil command option 119
simvision command option 46
sequence time
including in simulation database 46
viewing events in 255
Set Driver form 143
Set Function Breakpoint button 163
Set Function Breakpoint form 163
Set Watchpoint button 160, 164
-SHM, simvisdbutil command option 119
shortcut, keyboard 33
Show only X values along path button, in Trace Signals sidebar 202
Show Panner button 175
Show Search button
Design Browser sidebar 88
Source Browser 67
Show/Hide Fibers button 280
sidebar 28
Design Browser 81 to 90
Design Search 92 to 95
Source Navigation 74
Trace Signals 197 to 207
Sidebar Options form
Scope View
SystemC 91
SIGBUS 167
SIGEMT 167
SIGFPE 167
SIGILL 167
SIGINT 167
signal
changing the name of
in Waveform window 217
creating a scrollable region
in Design Browser window 105
in Waveform window 216
displaying
in Design Browser sidebar 89
in Design Browser window 99
in Design Search sidebar 95
in Register window 298
in Source Browser 68
in Waveform window contributing 214
expanding and collapsing
in Design Browser window 102
in Waveform window 215
rearranging
in Waveform window 222
searching for
in Design Search sidebar 92
sending to containing window
in Design Browser sidebar 90
in Design Search sidebar 95
splitting
in Design Browser window 104
in Waveform window 215
tracing the path of 197 to 207
signal breakpoint 133
signal evaluation phase 153
signal list
in Design Browser window
filtering 101
preferences 108
sorting 100
options 91
in Waveform window 210
searching 226
Signal List Design Browser tab, Preferences form 318
Signal List Search toolbar 27
signal name format
in Design Browser window 100

in Waveform window 216
signal transition, tracing in Source Browser 76
signal value
applying mnemonic map to 220
changing and monitoring during simulation 139
depositing 141
in Design Browser window
choosing the radix of 101
justifying 101
monitoring 97
in Verilog-XL 143
-SIGNAL, simvisdbutil command option 119
SIGPROF 167
SIGSEGV 167
simulation 145
connecting to 47
disconnecting 42
preparing your design for 37
reinvoking 147
resetting 147
running 145
in Source Browser 74
saving and restoring checkpoint 148
terminating and disconnecting 43
terminating and post-processing 43
Simulation Cycle Debug tab, Preferences form 324
Simulation Cycle Debugger 20, 153
customizing 324
invoking 155
Print form 158
saving and printing contents of 157
simulation database 109 to 123
See also database
Simulation menu
Advance 146
Delta Cycle 157
Process 157
Simulation Phase 157
Timepoint 157
Create Command Alias 150
Create Debug Variable 151
Create Driver 143
Create Force 140
Create Monitor 144
Create Probe 73
NC simulator 125
Verilog-XL 126
Next 146
Next in Scope 146
Reinvoke Simulator 147
in Source Browser 79
Restart from Checkpoint 149
Return 146
Run 146
Save Checkpoint 148
Set Breakpoint
Condition 135
Line
NC simulator 133
Verilog-XL 133
Process 136
Signal
NC simulator 134
Verilog-XL 134
Subprogram 137
Time
NC simulator 129
Verilog-XL 131
Show
Aliases 150
Breakpoints 138
in Memory Viewer 265
Forces 140
Probes 127
Variables 151
Step 146
Stop 146
SystemC/C++/C Debug 161
Terminate & Disconnect 43
Terminate & Post-Process 43
Simulation Settings tab, Preferences form 313
simulation time toolbar 31
simulation toolbar 145
simulator
command in user-defined button 327
command script 60
debugging environment 53
Simulator tab, Properties window 21
-SIMVISARG, ncsim option 38
simvisdbutil, batch database translation utility 117
SimVision
command in user-defined button 327
command script 60
debugging environment 53
simvision command 44
specifying default options 47

simvision.svcf command script 57
.simvision/passwd file 51
SIMVISION_HOME environment variable 55
SIMVISION_WORKDIR environment variable 48, 59, 66, 113, 114
SIMVISIONOPTS environment variable 47
.simvisionrc file 46, 61
Slope Between Cursors measurement 303
-SNAPSHOT, simvision command option 46
Source Browser 19, 63 to 79
  customizing 318
  displaying at startup 46
  viewing event source code 157
Source Browser button 19, 63
Source Browser tab, Preferences form 318
  Colors 319
  Files 320
Source Navigation sidebar
  viewing API call stack 165
-SOURCE, simvision command option 46
-SST, simvision command option 46
SST2 database 20, 109
  creating
    with NC simulator 111
    with Verilog-XL 112
  exporting 115
  importing 113
  opening 113
-SST2, simvisdbutil command option 120
Standard toolbar 27
statement trace
  file (.stc) 109
  saving information 76
.stc file 109
stream function 288
stream, filtering 286 to 293
streamoverlay function 290
subprogram breakpoint 136
Subprogram Breakpoint form 137
subtraction operator (-), in Schematic Tracer 187
synchronization, of simulation data and windows 31
SystemC
  scope view icon
    module 84
    thread process 84
  Scope View Options form 91
SystemC/C++/C Variables sidebar 29, 166
SystemVerilog
  scope view icon
    class 83
    interface 83
    modport 83
    program block 83

T

Target button 23
target icon, in Register window 297
target window 23
task, scope view icon 84
Tcl/Tk
  command in user-defined button 327
  expression in condition breakpoint 135
Terminate & Disconnect button 43
Terminate & Post-Process button 43
thread process
  scope view icon 84
threshold, rise/fall time parameter 306
time
  breakpoint 129
  navigating in Register window 301
  range
    in Properties window 254
    saving 253
  shift
    in Register window 298
    in Trace Signals sidebar 204
    in Waveform window 228
  units 31
Time Ranges tab, Properties window 21
Time Search toolbar 28
Time toolbar 30
-TIMEUNITS
  simvisdbutil command option 120
-TITLE
  simvision command option 47
toolbar 27
  adding and removing 325
  annotation, Register window 299
  choosing to display 28
  Cursor Control 27
  moving 28
  Send To 23
  Signal List Search 27
  Time 30
  Time Search 28
  user-defined 27

adding to other SimVision windows 329
creating 326
modifying 330
writing a window-independent script 329
Zoom 28
zoom
Schematic Tracer 174
Toolnet Cross-Probing tab, Preferences form 312
Trace Active option, in Trace Signals sidebar 201
Trace button, in Trace Signals sidebar 199
Trace Driving Logic button 200
Trace Loading Logic button 198
Trace menu
Driving Logic 176
Follow-X Forwards 176
Loading Logic 176
Logic Cone Backwards 177
Logic Cone Fowards 177
Point to Point
between a pin and a cell 179
between a pin and a scope 178
between two pins 178
Trace Signal Through Hiererarchy 177
X Backwards 176
trace path, changing the color of 206
Trace Signals sidebar 29, 197
performing functions on objects in 206
sending signals to the containing window 204
shifting time in 204
tracing X values 202
using with the Schematic Tracer 206
Trace-X Backwards button 203
Trace-X Forwards button 203
transaction 279
error information 282
overlapping 284
stream
filtering
in Design Browser window 101
in Waveform window 286
finding in simulation database 279
viewing in the Waveform window 281
Transaction Explorer tab, Prefernces form 324
Transaction Explorer toolbar 293
transaction stream
show or hide button
in the sidebar 90
signal list 101
transition file (.trn) 109
Translate .lib to cellmap format form 193
.trn file 109

U

Unlock button 30
user-defined button 27
adding to other SimVision windows 329
creating 326
user-defined toolbar 27
creating 326
modifying 330

V

Value At Baseline measurement 303
Value At Cursor measurement 303
variable
debug 150
displaying
in Design Browser sidebar 89
in Design Browser window 99
in Design Search sidebar 95
in Register window 298
in Source Browser 68
forcing the value of 140
rearranging
in Waveform window 222
searching for
in Design Search sidebar 92
sending to containing window
in Design Browser sidebar 90
in Design Search sidebar 95
VHDL
depositing a value 141
forcing the value of 140
VCD database 20
batch conversion 117
exporting to 115
opening 113
-VCD, simvisdbutil command option 120
Verilog AMS Signal Options tab, Preferences form 311
verilog command
invoking in post-processing mode 50

invoking with SimVision 38
Verilog-XL
creating a database 112
invoking in post-processing mode 50
invoking with SimVision 38
setting line breakpoint 133
setting signal breakpoint 134
setting time breakpoint 131
-VERSION
simvisdbutil command option 120
simvision command option 47
View menu
Center on Cursor 248
Define Grid 229
Display Values 69
Lock/Unlock Scope 87
Manage Bookmarks
Source Browser 77
Number of Columns 264
Panner 175
Regenerate 172
RTL 186
Show Cell Shapes 194, 196
Show Full Signal Names 173
Show Values 108, 173
Sidebar 30, 197, 211
Size to Fit 263
Sort
By Declaration 100
By Name 100
Toolbars 28
Customize 325, 329
Zoom
Cursor-Baseline 225, 271
Fit 174
Full X 225, 271
Full Y 225, 271
In 174
In X 225, 271
Out 174
Out X 225, 271
Previous 225, 271
virtual signal 237

W

Watch Live Data button 31
Schematic Tracer 173
Source Browser 69
Waveform window 245
Watch window 106
watchpoint, C/C++ variable 164
waveform
add command, in .simvisionrc file 61
new command, in .simvisionrc file 61
Waveform button 19, 213
Waveform window 19
displaying at startup 47
displaying waveforms 209 to 229
keyboard shortcuts 316
managing time in 243 to 257
organizing signals 231 to 242
viewing analog data 269 to 278
viewing transactions 279 to 294
-WAVES, simvision command option 47
waves.shm file 110
WBUFGATE, in Schematic Tracer 188
where, C++ debugging command 168
window
iconfiying and activating 24
managing multiple 23
opening and closing 23
renaming 25
tiling 25
Windows menu
Cascade 25
Iconify All 25
New 24
Assertion Browser 20
Measurement 20, 303
Register 295
Schematic Tracer 97, 169
Tile Horizontally 25
Tile Vertically 25
Tools 24
Console 20
Cursors 245, 246, 247
Databases 122
Expressions 237, 241
Groups 232
Markers 250, 251
property type 21
SimCompare Manager 20
Simulation Cycle Debug 20, 155
Simulator 148
Databases 121
Time Ranges 254
Windows 24
wire
forcing the value of unexpanded 140
in Schematic Tracer 186

resolution phase 153

X

X value
tracing in Schematic Tracer 176
tracing in Trace Signals sidebar 202, 203
X-axis
grid 228
marker in analog waveforms 269
XNOR gate, in Schematic Tracer
abstracted RTL 186
gate primitive 188
XOR gate, in Schematic Tracer
abstracted RTL 186
gate primitive 188
-XSUB, simvisdbutil command option 120

Y

Y-axis
grid 276
marker 269

Z

zoom mode, in Waveform window 224
Zoom toolbar 28