Cadence NC-Verilog Simulator Tutorial with SimVision

Product Version 5.4
August 2004

Contents

Preface

This tutorial shows you how you can use NCLaunch and the SimVision analysis environment to simulate and debug a simple design.

This tutorial should take aproximately 1 hour to complete.

Audience

This tutorial is intended for designers who want to learn how to use the SimVision analysis environment for simulating digital designs.

Prerequisites

You must be familiar with the Verilog hardware description language.

Tool Requirements

Completion of this tutorial requires that you have access to the following tools:

- NC-Verilog simulator, version IUS 5.4 or later
- The SimVision analysis environment, version IUS 5.4 or later
- NCLaunch, version IUS 5.4 or later

This tutorial assumes that you are using either the UNIX or Windows platform.

For More Information

About the NC-Verilog simulator:

- See the *NC-Verilog Help*.

About NCLaunch:

- See the *NCLaunch User Guide*.

About the SimVision analysis environment:

- See the *SimVision User Guide*.

Customer Support

A qualified Applications Engineer at the Cadence Customer Response Center (CRC) is ready to answer all of your technical questions on the use of the product. You can contact the CRC by telephone at 1-877-CDS-4911 or via e-mail at `support@cadence.com`.

Typographic Conventions

The following typographic conventions are used in this manual:

Convention	Meaning	Example
Italic	Titles of books	*NC-Verilog Simulator Help*
	Names of buttons and menu choices in the graphical user interface	*File – Open Database*.
`Literal`	Program output or text that you type at the command line	`ncvhdl *.vhd`
`literal italic`	Variables, such as command-line arguments	`ncelab` *`top`*
Blue underlined	Hypertext links, which take you to a relevant section of this book, another document, or an exercise	See the *SimVision User Guide* for more information.
🎞 Video	Links to interactive exercises that you can run to duplicate the steps described in the tutorial	🎞 Video Now try this interactive exercise, About Interactive Exercises.

1 Introduction

This tutorial introduces you to the Cadence NC-Verilog simulator and SimVision.

The example used in the tutorial is a design for a drink dispensing machine written in the Verilog hardware description language. Using this example, you will learn how to:

- Compile Verilog source files, elaborate the design, and run the simulation using NCLaunch, a graphical user interface that helps you manage large design projects. NCLaunch helps you configure and launch the compiler, elaborator, and simulator. You can also run other tools from NCLaunch, such as the SDF Compiler, HDL Analysis and Lint, Code Coverage Analyzer, NCBrowse, and Comparescan.
- Debug a problem in the design using the SimVision analysis environment.

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, VHDL, SystemC, or mixed-language.

You can run SimVision in either of the following modes:

- Simulation mode

  In simulation mode, you view “live” simulation data. That is, you analyze the data while the simulation is running. You can control the simulation by setting breakpoints and stepping through the design.

  SimVision provides several tools to help you track the progress of the simulation:

  - Console Window
  - Source Browser
  - Design Browser
  - Cycle Viewer
  - Schematic Tracer
  - Waveform Window

- ❑ Register Window

  All of these windows are linked so that when you select an object in one window, it is selected in the other windows as well.

- Post-processing environment (PPE) mode

  In PPE mode, you analyze simulation data after simulation has completed. You have access to all of the SimVision tools, except for the simulator. As in simulation mode, all of these windows are linked so that, when you select an object in one window, it is selected in the other windows as well.

This tutorial introduces you to some of the major features of the following SimVision tools:

Console window

No icon — The Console window lets you enter Tcl simulator commands or SimVision commands.

Design Browser

The Design Browser lets you access the signals and variables in the design database.

Register window

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.

Source Browser

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

Waveform window

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

In addition to the SimVision features, this tutorial also shows you how to prepare your design for simulation with NCLaunch, a graphical user interface that helps you manage large design projects and configure your Cadence simulation tools.

The Drink Machine Example

The drink machine design is made up of three modules:

- `drink_machine` counts the amount of change that the user has entered, dispenses a drink, and returns any change that is due.
- `coin_counter` loads the machine with coins and determines when the machine is out of change.
- `can_counter` loads the machine with drinks and determines when the machine is empty.

The example also includes a testbench, which initializes the machine and buys drinks by depositing different combinations of nickels, dimes, and quarters.

The behavior for accepting coins and dispensing drinks is modeled as a state machine. The amount of money that the user has deposited so far defines the current state. The type of coin that the user deposits determines the machine's next state.

For example, when no money has been deposited, the machine is in the `idle` state. When the user adds a nickel, the machine transitions to the next state, `five`. When the current state is `five` and the user adds a quarter, the machine transitions to the next state, `thirty`. When the user has added exactly 50 cents, the machine dispenses a drink and transitions back to the `idle` state. When the user adds more than 50 cents, the machine dispenses a drink, returns the correct change, and transitions back to the `idle` state.

Table 1-1 Drink Machine State Table

Current State	Transition Value	Next State

`idle 4’d0`	`nickel_in`
	`dime_in`
	`quarter_in`
		`five 4’d1`
		`ten 4’d2`
		`twenty_five 4’d5`

`five 4’d1`	`nickel_in`
	`dime_in`
	`quarter_in`
		`ten 4’d2`
		`fifteen 4’d3`
		`thirty 4’d6`

`ten 4’d2`	`nickel_in`
	`dime_in`
	`quarter_in`
		`fifteen 4’d3`
		`twenty 4’d4`
		`thirty_five 4’d7`

`fifteen 4’d3`	`nickel_in`
	`dime_in`
	`quarter_in`
		`twenty 4’d4`
		`twenty_five 4’d5`
		`forty 4’d8`

`twenty 4’d4`	`nickel_in`
	`dime_in`
	`quarter_in`
		`twenty_five 4’d5`
		`thirty 4’d6`
		`forty_five 4’d9`

`twenty_five 4’d5`	`nickel_in`
	`dime_in`
	`quarter_in`
		`thirty 4’d6`
		`thirty_five 4’d7`
		`fifty 4’d10`

`thirty 4’d6`	`nickel_in`
	`dime_in`
	`quarter_in`
		`thirty_five 4’d7`
		`forty 4’d8`
		`nickel_out 4’d11`

`thirty_five 4’d7`	`nickel_in`
	`dime_in`
	`quarter_in`
		`forty 4’d8`
		`forty_five 4’d9`
		`dime_out 4’d12`

`forty 4’d8`	`nickel_in`
	`dime_in`
	`quarter_in`
		`forty_five 4’d9`
		`fifty 4’d10`
		`nickel_dime_out 4’d14`

`forty_five 4’d9`	`nickel_in`
	`dime_in`
	`quarter_in`
		`fifty 4’d10`
		`nickel_out 4’d11`
		`two_dime_out 4’d14`

`fifty 4’d10`		`idle`
`nickel_out 4’d11`		`idle`
`dime_out 4’d12`		`idle`
`nickel_dime_out 4’d13`		`idle`
`two_dime_out 4’d14`		`idle`

About Interactive Exercises

After each procedure in this book, you have the opportunity to perform the procedure yourself. These interactive exercises simulate the tool. You see the same windows, make the same choices from menus, and enter the same information in forms, as you would if your were running the tool. However, if you need help performing a particular step, you can click a *Help* button, and the interactive exercise displays instructions for that step.

Interactive exercises are marked by the *Video* icon. To learn more about them, click on the title of the exercise below.

Video

Now try this interactive exercise, About Interactive Exercises.

For More Information

SimVision provides other tools not used in this tutorial.

Tool/Feature	**Description**
Schematic Tracer	The Schematic Tracer displays a Verilog or VHDL design as a schematic diagram and lets you trace a signal through the design. See Chapter 12, “Viewing a Design Schematic,” in the *SimVision User Guide*.
Simulation Cycle Debugger	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 for Verilog-XL or AMS Designer. See Chapter 11, “Debugging at the Delta Cycle Level,” in the *SimVision User Guide*.

2 Getting Started

Before you can simulate your design, you must compile and elaborate it. Compiling the design produces an internal representation for each HDL design unit in the source files. Elaborating the design constructs a design hierarchy based on the instantiation and configuration information in the design, establishes signal connectivity, and computes initial values for all objects in the design.

You can compile, elaborate, and simulate your design by running the following tools:

`ncvlog`	Compiles the Verilog source files.
`ncelab`	Elaborates the design and generates a simulation snapshot.
`ncsim`	Simulates the snapshot.

You can run NC-Verilog in single-step invocation mode with the `ncverilog` command.

You can also use NCLaunch, a graphical user interface, to manage your design projects. NCLaunch helps you configure and launch the simulation tools, including Comparescan, SDF Compiler, HDL analysis, Code Coverage Analyzer, and NCBrowse. You can run the tools in multi-step mode or in single-step mode.

This tutorial shows you how to use NCLaunch in multiple step mode.

Copying the Tutorial Source Files

All of the source files for this design are included in your Cadence installation hierarchy.

On UNIX systems, the source files are in the following directory:

```
install_dir/doc/ncvlogtut/examples/files
```

Create a directory, and then copy the tutorial files to this directory. For example:

```
mkdir tutorial
cd tutorial
```

```
cp -r install_dir/doc/ncvlogtut/examples/files .
```

On Windows systems, if you have installed the software in the default location, the source files are in:

```
C:\Program Files\Cadence Design Systems\IUS\doc\ncvlogtut\examples\files
```

Copy and paste this folder into a working directory. This tutorial uses the following location:

```
C:\tutorial
```

Starting NCLaunch on UNIX Systems

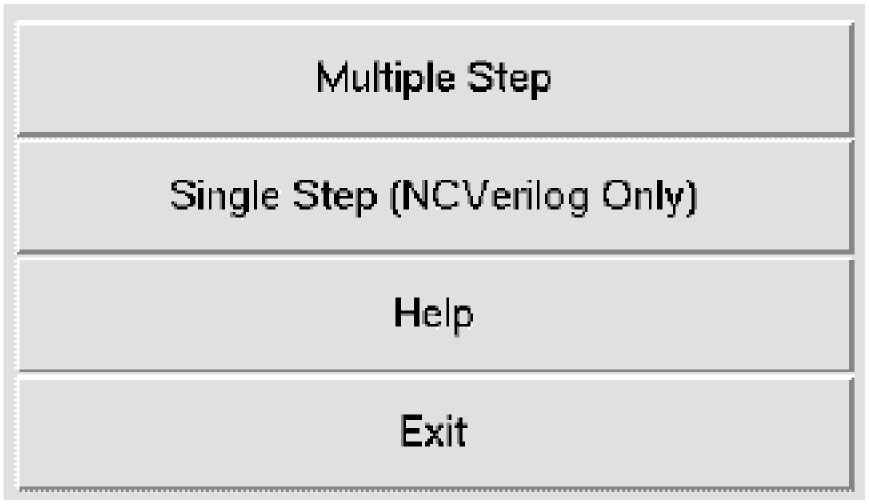
1. Start NCLaunch from the directory into which you have copied the source files for the tutorial.

   ```
   nclaunch -new &
   ```

   The `-new` option specifies that you want to work on a new design.

   At startup, NCLaunch displays a list of modes in which you can run the tool, as shown in Figure 2-1 on page 2.

Figure 2-1 NCLaunch Run Modes

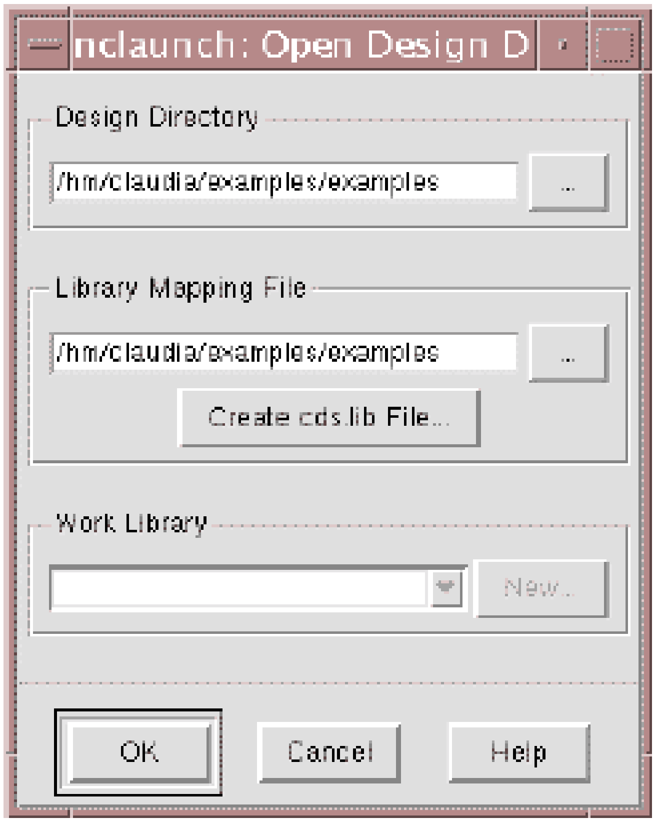


   Multiple Step mode uses the `ncvlog` and `ncelab` commands to compile and elaborate your design; Single Step mode uses the `ncverilog` command.

2. Click *Multiple Step*.

Because this is a new design, you must define a cds.lib file and work libarary. NCLaunch opens the Open Design Directory form, shown in Figure 2-2 on page 3, so that you can do this.

Figure 2-2 Open Design Directory Form


nclaunch: Open Design D

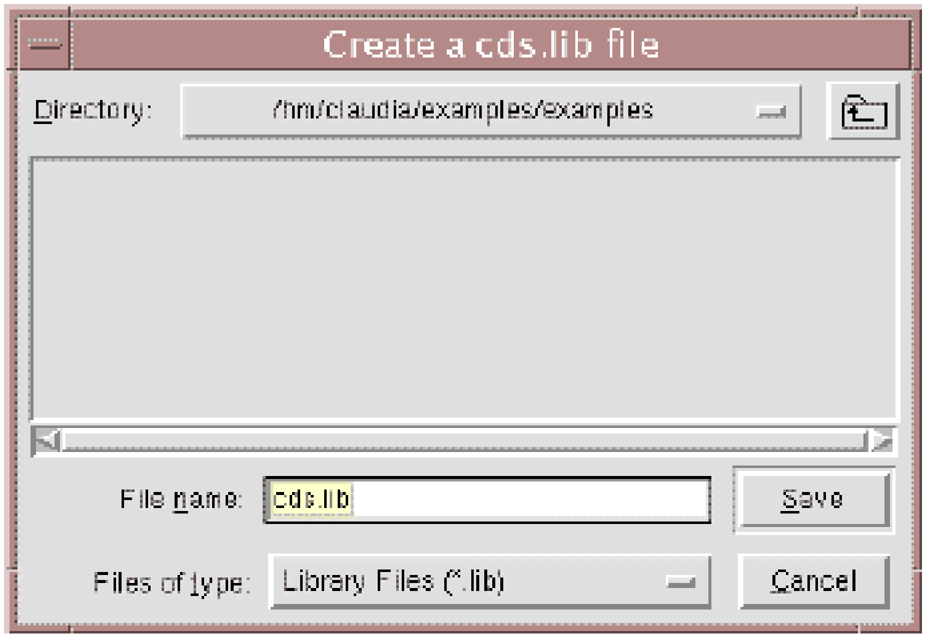
Design Directory: /hm/claudia/examples/examples

Library Mapping File: /hm/claudia/examples/examples

Work Library:

3. In the *Library Mapping File* field, click the *Create cds.lib File* button. This opens the Create a cds.lib file form, as shown in Figure 2-3 on page 4.

Figure 2-3 Create a cds.lib File Form



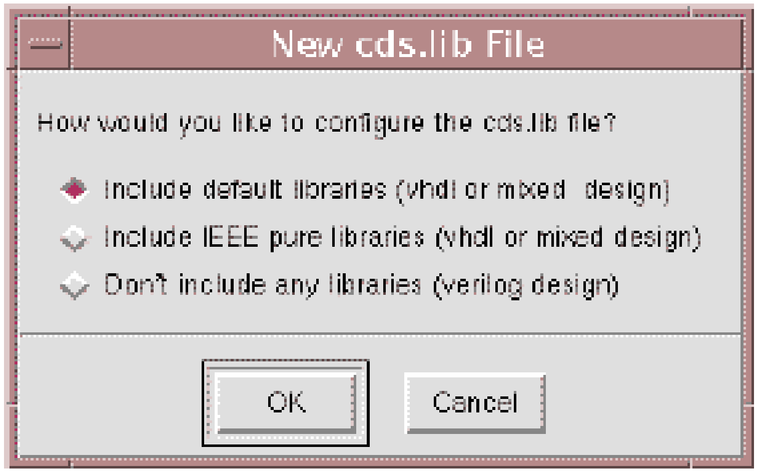
Directory: /hm/claudia/examples/examples

File name: cds.lib

Files of type: Library Files (*.lib)

4. Click *Save* to create a library mapping file with the default name, `cds.lib`. NClaunch opens the New cds.lib File form, as shown in Figure 2-4 on page 5. This form lets you pick the libraries that you want to use in the design. For Verilog files, you can choose *Don't include any libraries*. For VHDL and mixed-language designs, choose either the default libraries or the IEEE pure libraries.

Figure 2-4 New cds.lib File Form

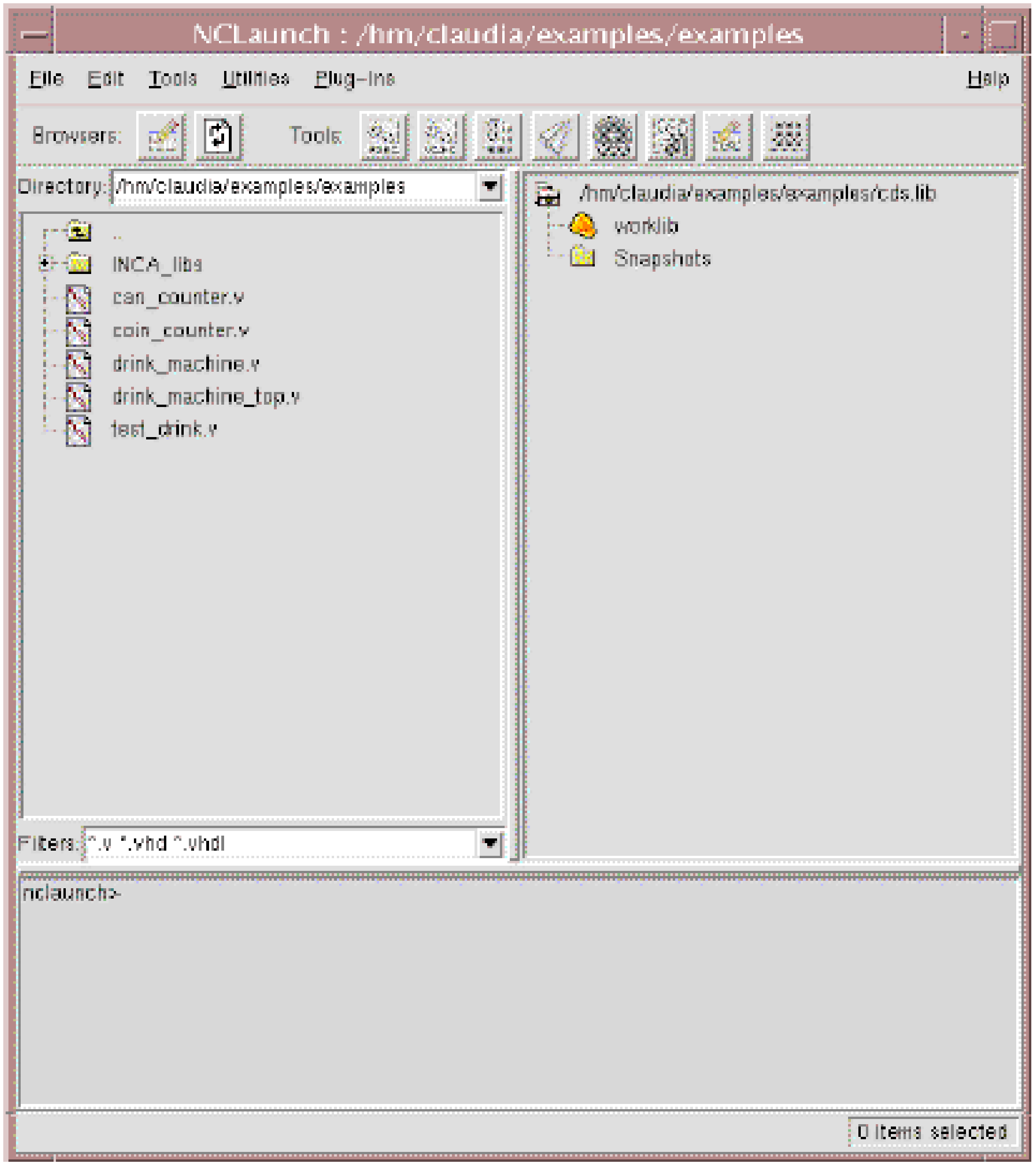

New cds.lib File

How would you like to configure the cds.lib file?

[✓ Include default libraries (vhdl or mixed design)]
Include IEEE pure libraries (vhdl or mixed design)
Don't include any libraries (verilog design)

5. Click *OK* to close the New cds.lib File form. NCLaunch displays the main window, as shown in Figure 2-5 on page 6.

Figure 2-5 NCLaunch Main Window



The left side of the window shows all of the files in the current directory. The right side will show the design libraries, after you have compiled the source files and elaborated the design. The top of the window contains menus and buttons for starting the tools.

Video

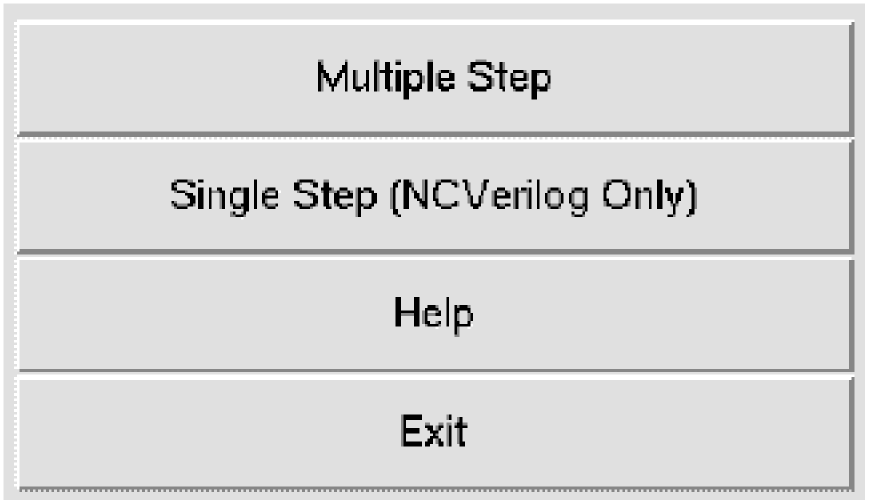
Now try this interactive exercise, Getting Started on UNIX Systems.

Starting NCLaunch on Windows Systems

1. Invoke NCLaunch from the *Start* menu, by choosing *Programs – Cadence Design Systems – Design & Verification – NCLaunch*.

   At startup, NCLaunch displays a list of modes in which you can run the tool, as shown in Figure 2-1 on page 2.

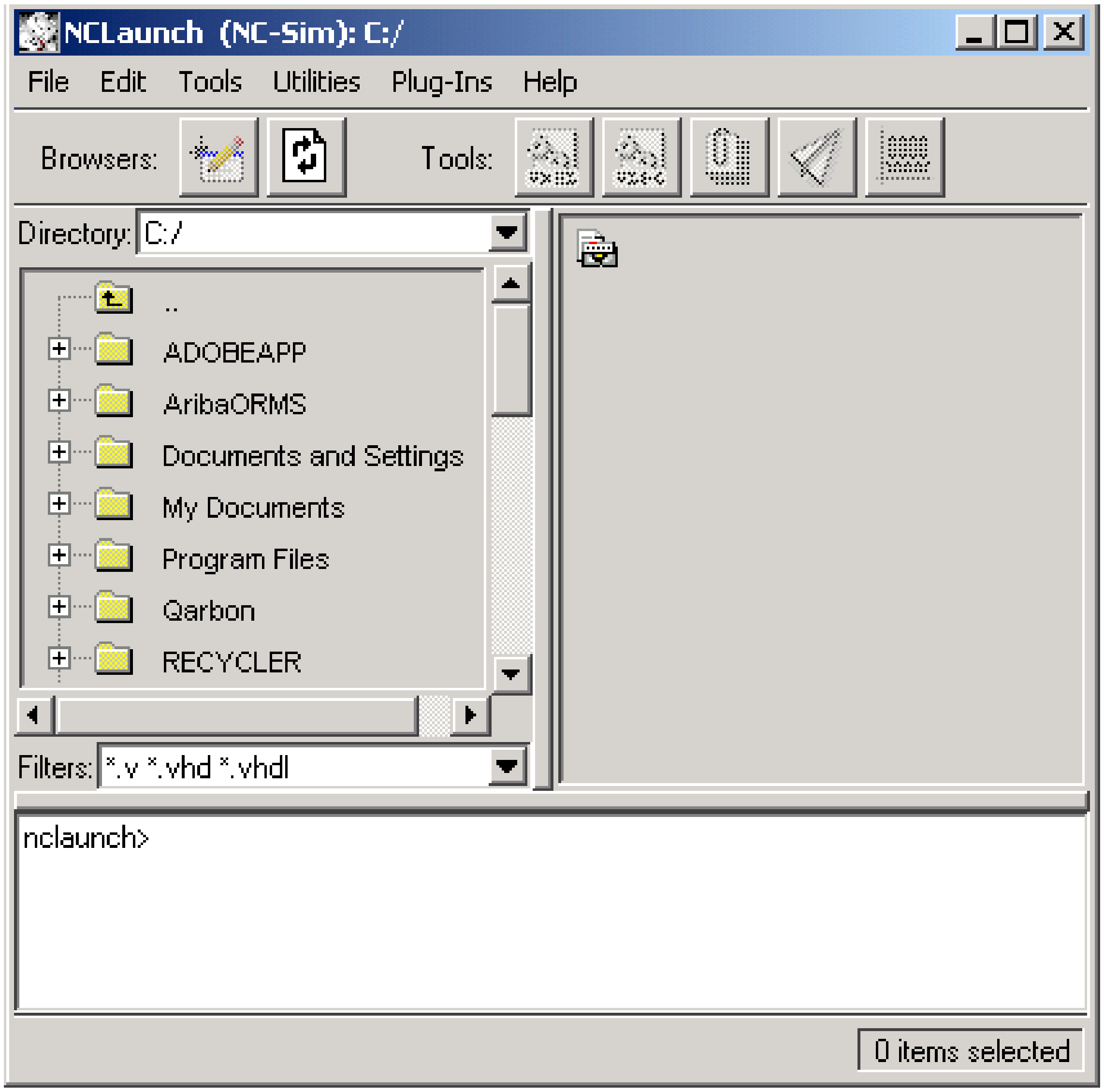
Figure 2-6 NCLaunch Run Modes



   Multiple Step mode uses the `ncvlog` and `ncelab` commands to compile and elaborate your design; Single Step mode uses the `ncverilog` command.

2. Click *Multiple Step*. NCLaunch displays the main window, as shown in Figure 2-7 on page 8.

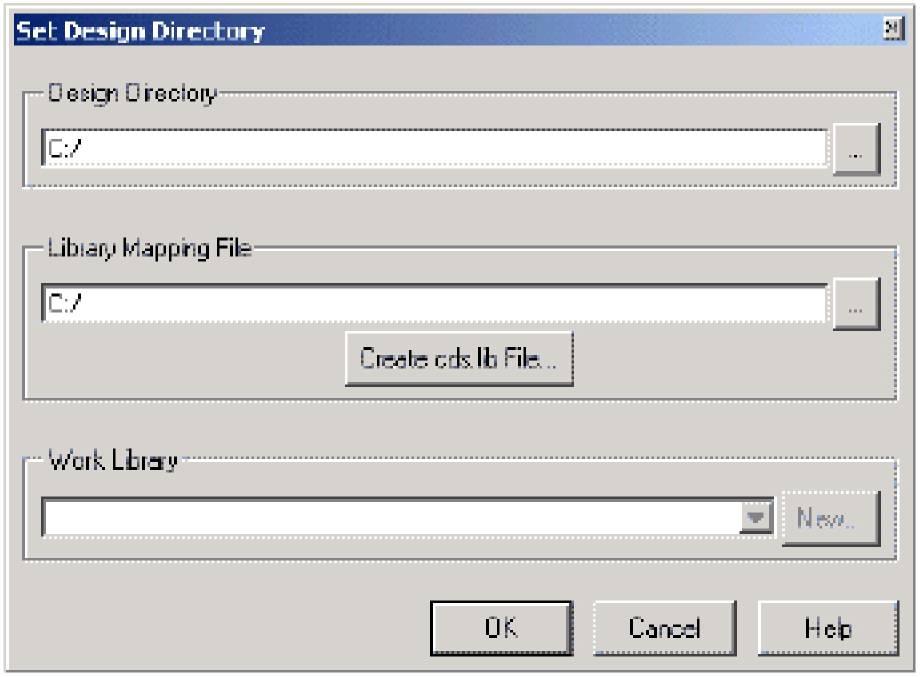
Figure 2-7 NCLaunch Main Window



```
nclaunch>
```

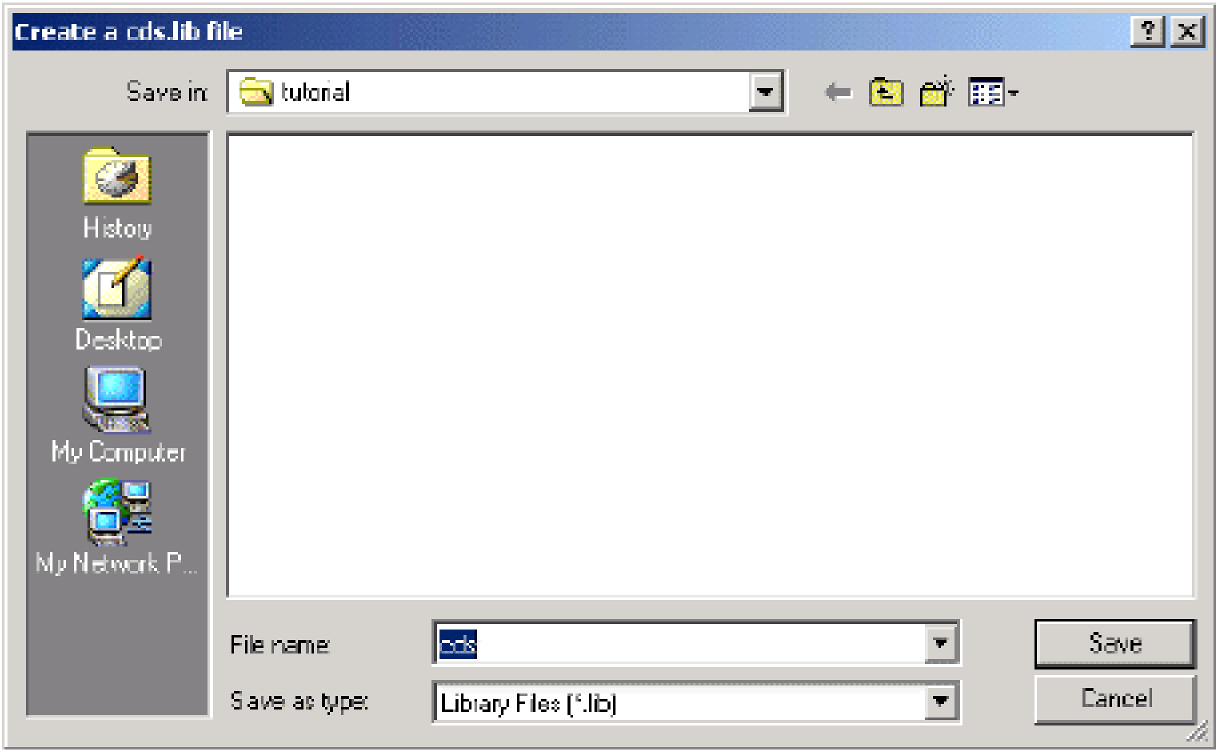
3. Select *File – Set Design Directory*. This opens the Set Design Directory form, as shown in Figure 2-8 on page 9.

Figure 2-8 Set Design Directory Form



4. Specify the name of the directory in the *Design Directory* field.
5. In the *Library Mapping File* field, click the *Create cds.lib File* button. This opens the Create a cds.lib file form, as shown in Figure 2-9 on page 10.

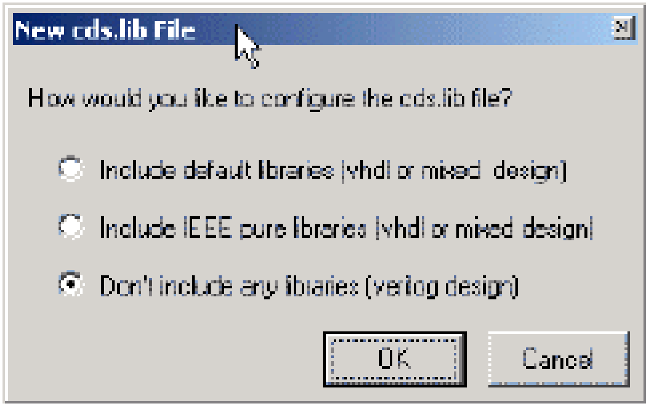
Figure 2-9 Create a cds.lib File Form



File name: cds
Save as type: Library Files (*.lib)

6. Click the *Save* button.
7. In the New cds.lib File form, shown in Figure 2-10 on page 10, enable the *Don't include any libraries (verilog design)* button and click *OK*. This closes the form.

Figure 2-10 New cds.lib File Form



How would you like to configure the cds.lib file?

- Include default libraries (vhdl or mixed design)
- Include IEEE pure libraries (vhdl or mixed design)
- [✓ Don't include any libraries (verilog design)]

8. In the Set Design Directory form, click *OK*. In the NCLaunch window, a folder called `INCA_libs` has been created. This folder contains a folder called `worklib`, which will be used as the work library.

Video

Now try this interactive exercise, Getting Started on Windows Systems.

Compiling and Elaborating the Design

Before you can simulate your design, you must compile the source files using the Verilog compiler, and you must elaborate the design into a snapshot using the elaborator. A snapshot is the representation of your design that the simulator uses. It is stored in the work library along with the other intermediate objects generated by the compiler and elaborator.

The NCLaunch main window gives you access to the tools you need to compile and elaborate the design, as well as access to several utilities. You access the tools and utilities by using the *Tools* or *Utilities* menu, or by clicking the appropriate button on the toolbar. Not all tools and utilities listed in the menus are available on the toolbar, but you can customize the toolbar to add any tools or utilities that you like.

Table 2-1 on page 11 shows the buttons that NCLaunch includes in the toolbar.

Table 2-1 NCLaunch Toolbar Buttons

Button	Tool
VHDL	VHDL Compiler
VLOG	Verilog Compiler
📎	Elaborator
✈	Simulator

Button	Tool
	HDL Analysis and Lint
	NCBrowse (UNIX systems only)
	HAL Definition Editor
	Waveform window

Compiling the Design

To compile the design:

1. Select the Verilog files that make up the design:

   ```
   can_counter.v
   coin_counter.v
   drink_machine.v
   drink_machine_top.v
   test_drink.v.
   ```

   To select multiple files, hold down the `Control` key and click on each filename.

2. Click the *Verilog Compiler* button.

   The I/O area at the bottom of the window displays the `ncvlog` command that runs as a result of your selections, and it displays the messages that NC-Verilog generates as it compiles the design files.

By default, NC-Verilog creates a directory called `INCA_libs` and a subdirectory called `worklib`. All modules in the design are compiled into the `worklib` directory. Notice that the `INCA_libs` directory now appears in the file browser (left side) of the NCLaunch window, and that the design library `worklib` has been added to the library browser (right side) of the window.

Video

Now try this interactive exercise, Compiling the Design Source Files.

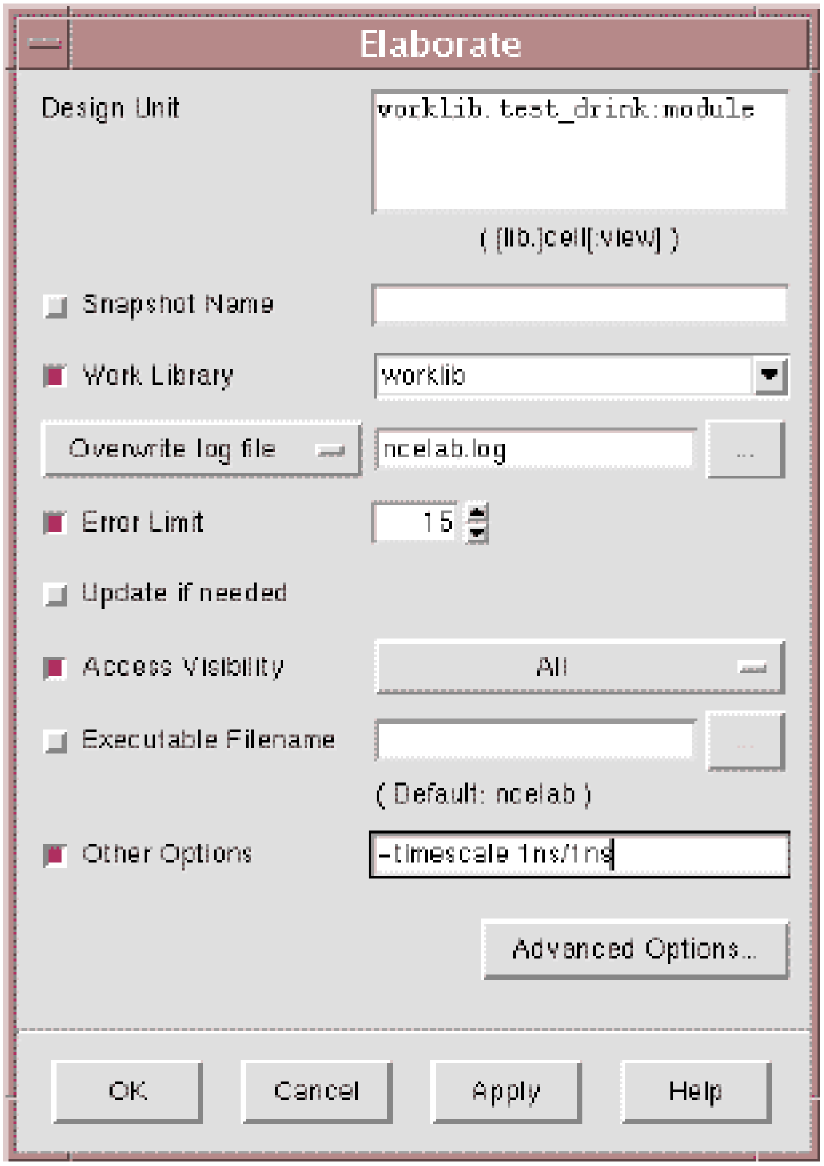
Elaborating the Design

To elaborate a design, you typically expand the work library (`worklib`), select the top-level design unit, and click on the *Elaborator* button. However, for this tutorial, you must set some elaborator options.

Perform the following steps to set the options and elaborate the design:

1. Expand the work library (`worklib`) by clicking on the plus sign next to the hardhat icon.
2. Expand the top-level design unit. In this example, the top-level unit is the Verilog testbench, `test_drink`.
3. Select `module`.
4. Choose *Tools – Elaborator* to open the Elaborate form, shown in Figure 2-11 on page 14.

Figure 2-11 Elaborate Form



Design Unit	worklib.test_drink:module
☐ Snapshot Name	
■ Work Library	worklib
Overwrite log file	ncelab.log
■ Error Limit	15
☐ Update if needed	
■ Access Visibility	All
☐ Executable Filename	
■ Other Options	-timescale 1ns/1ns

Notice that the *Access Visibility* button is selected and that the value is set to *All*. This option provides full access (read, write, and connectivity access) to simulation objects so that you can probe objects and scopes to a simulation database and debug the design.

Note: Access to simulation objects is on by default when you are using NCLaunch. When you are using the command-line interface, access is off by default and you must start the elaborator with the `-access` option. For example:

```
ncelab -access +rwc worklib.test_drink:module
```

Note: If you are running the Verilog Desktop, debug options, such as *Access Visibility*, are always on. This option is not available in the Verilog Desktop.

5. Only one module in the drink machine contains the `‘timescale` compiler directive. To prevent the elaborator from issuing errors because the other modules do not set a timescale set, enable the *Other Options* button and enter the following option in the text field:

   ```
   -timescale 1ns/1ns
   ```

6. Click *OK* to elaborate the design.

   The I/O area at the bottom of the window displays the `ncelab` command that runs as a result of your selections, and it displays the messages that the elaborator generates.

Tip

If you receive elaborator error messages, you may have made a mistake when running these steps. For example:

❑ Did you select the correct design unit name?

❑ Did you remember to include the `-timescale` option?

Perform these steps again if you get errors during elaboration.

Video

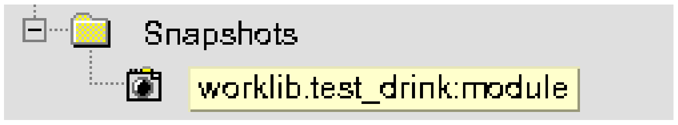
Now try this interactive exercise, Elaborating the Design.

Starting the Simulator

To start the simulator:

1. Expand the *Snapshots* folder to display the snapshots that are available in your library.
2. Select the snapshot you want to simulate, as shown in Figure 2-12 on page 16.

Figure 2-12 Selecting a Snapshot to Simulate

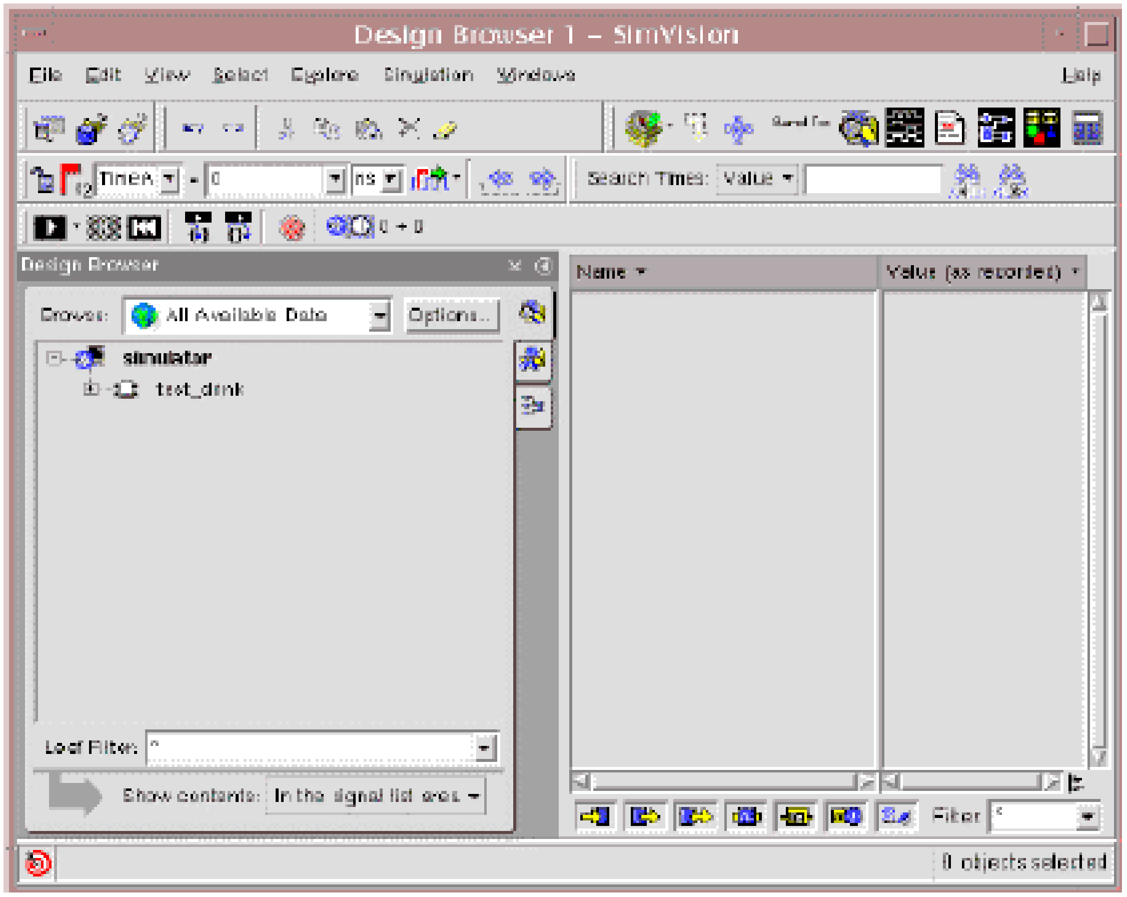


3. Click the *Simulator* button.

The Design Browser and the Console window appear. You can access your design hierarchy in the Design Browser and enter SimVision and simulator commands in the Console window.

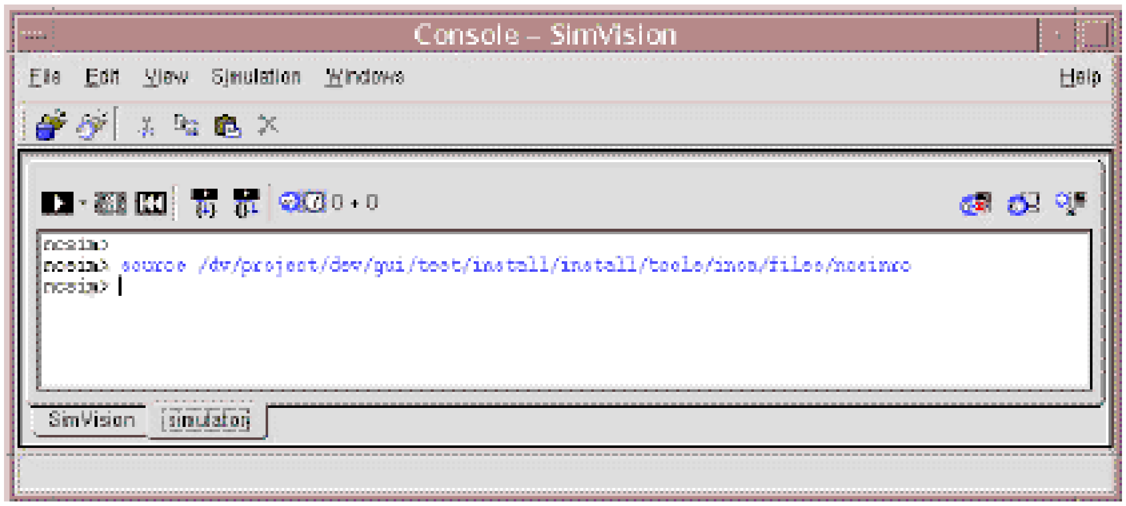
Figure 2-13 on page 16 shows the Design Browser at startup. In the Design Browser sidebar on the left side of the window, SimVision places the simulation at the top of the hierarchy and assigns it the name `simulator`. The top-level of the design hierarchy is placed below the simulation. In this example, it is named `test_drink`.

Figure 2-13 Design Browser



At startup, the Console window has two tabs, as shown in Figure 2-14 on page 17. The *SimVision* tab lets you enter SimVision commands and 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 2-14 Console Window



```
ncsim>
ncsim> source /dv/project/dev/gui/test/install/install/tools/inca/files/ncsimrc
ncsim>
```

Video

Now try this interactive exercise, Starting the Simulator.

Exiting NCLaunch

After invoking the simulator, you can exit NCLaunch.

To exit NCLaunch:

➤ Bring the NCLaunch main window to the foreground and choose *File – Exit* from the menu bar.

For More Information

This chapter describes how to run NC-Verilog in multi-step invocation mode using NCLaunch. There are other ways to prepare your design for simulation.

Tool/Feature	**Description**
NCLaunch	You can compile the source files, elaborate the design, and start the simulator in a single step using the NCLaunch `ncverilog` support as described in the *NCLaunch User Guide*.
NC-Verilog, Multiple Step mode	You can compile the source files, elaborate the design, and start the simulator using the `ncvlog`, `ncelab`, and `ncsim` commands, as described in Multi-Step Invocation (Library-Based Mode), in the *NC-Verilog Simulator Help*.
NC-Verilog, Single Step mode	You can compile and elaborate the design in a single step using the `ncverilog` command, as described in Single-Step Invocation with ncverilog in the *NC-Verilog Simulator Help*.

3 Simulating the Design

SimVision lets you choose the simulation data that you want to save for particular objects or scopes. This can help keep the size of simulation data files as small as possible. At a later time, you can load a simulation data file back into the Waveform window and reexamine the simulation results.

Selecting the Simulation Data to Save

You can save simulation data by probing the design during simulation and saving the values of the probed objects to a database.

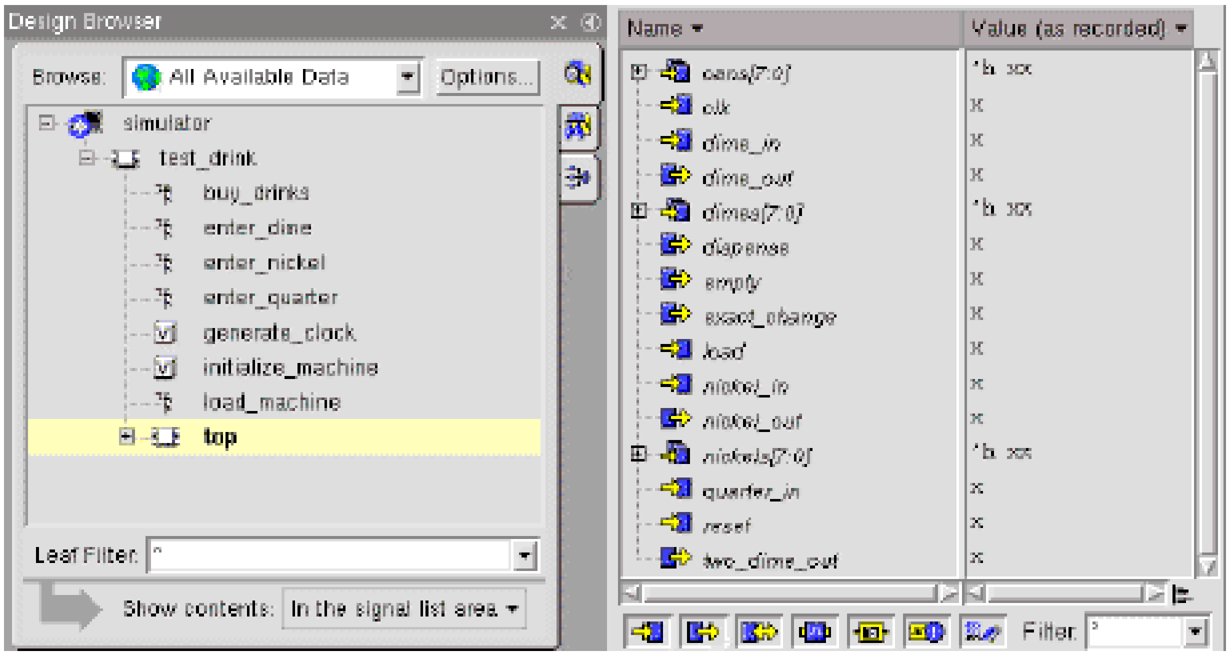
There are two types of probe commands:

- Probe a specific object or objects. The values of the specified objects are saved in the database.
- Probe a scope or scopes. You can choose the type of information you want to save, such as the inputs to that scope, and you can choose whether to probe some or all subscopes.

To probe all objects in all scopes, beginning at the `top` module:

1. In the Design Browser, click on the + icon next to `test_drink` to expand the hierarchy.
2. Select the `top` scope. The signal list on the right side of the window displays the signals for the `top` scope, as shown in Figure 3-1 on page 2. The signal list indicates the type of each signal—input, output, inout, internal signal, or transaction. You can use the filter buttons to choose which signals you want to see in the list.

Figure 3-1 Choosing the Top Scope



Browse: All Available Data

- simulator
  - test_drink
    - buy_drinks
    - enter_dime
    - enter_nickel
    - enter_quarter
    - generate_clock
    - initialize_machine
    - load_machine
    - **top**

Leaf Filter: *

Show contents: In the signal list area

Name	Value (as recorded)
cans[7:0]	'h xx
clk	x
dime_in	x
dime_out	x
dimes[7:0]	'h xx
dispense	x
empty	x
exact_change	x
load	x
nickel_in	x
nickel_out	x
nickels[7:0]	'h xx
quarter_in	x
reset	x
two_dime_out	x

Filter: *

Tip

If the signals are not displayed in the right side of the window, the *Show contents* field may be set to *In the selector below*. If so, choose *In the signal list area* from the drop-down menu, and the signals should appear as shown in Figure 3-1 on page 2.

3. Choose *Simulation – Create Probe* from the menu bar.

SimVision opens the Set Probe form. This form lets you probe one or more levels of subscope, choose the type of signals you want to probe, and write the probed information to any database.

4. For this probe:

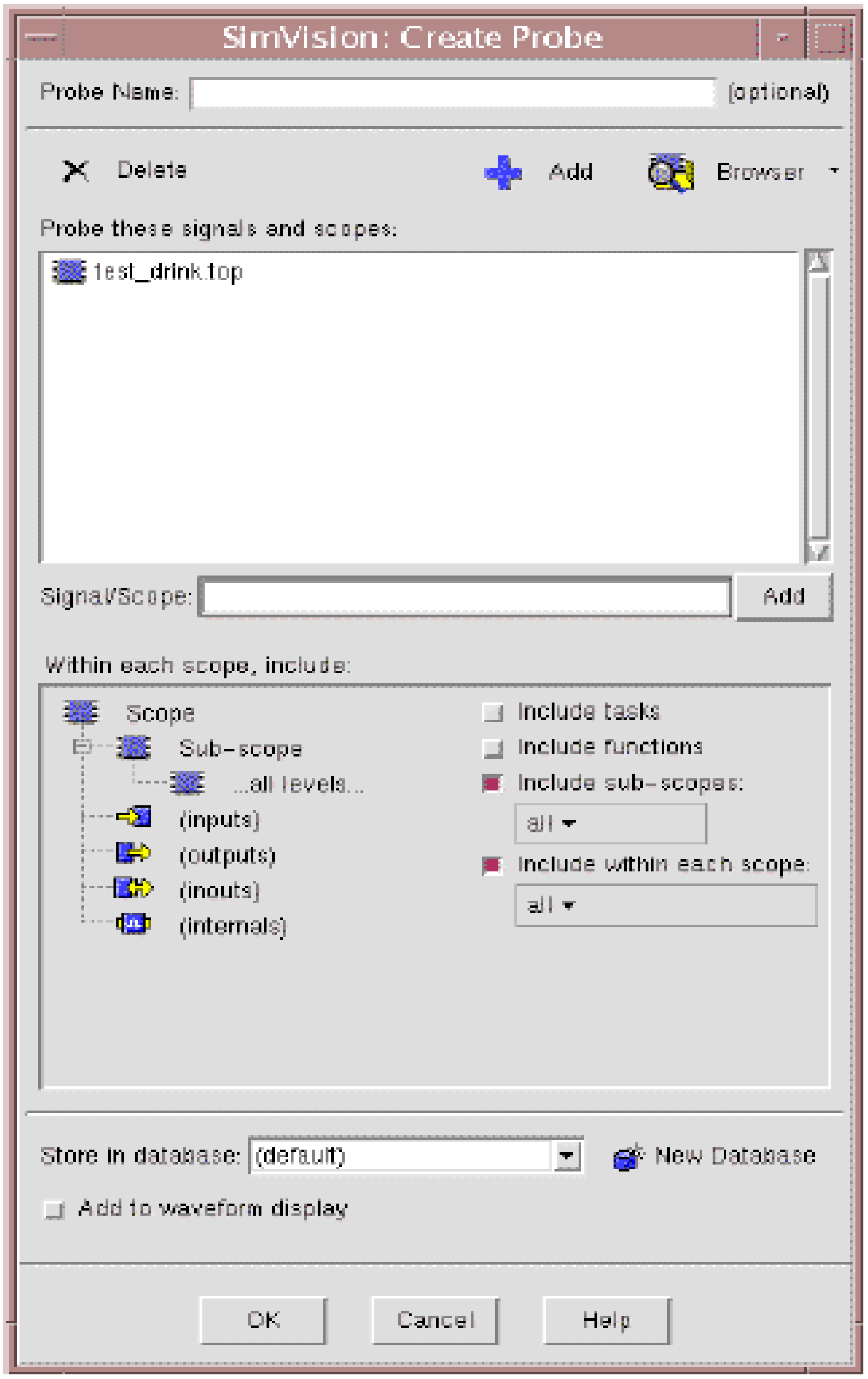
❑ Select *Include sub-scopes* and choose *all* from the drop-down list to include all the subscopes in the design.

❑ Select *Include within each scope* and choose *all* from the drop-down list to include all inputs, outputs, and ports.

❑ Deselect *Add to waveform display*.

The form should have the settings shown in Figure 3-2 on page 3.

Figure 3-2 Set Probe Form



SimVision: Create Probe

Probe Name: (optional)

Probe these signals and scopes:

test_drink.top

Signal/Scope:

Within each scope, include:

Include tasks
Include functions
Include sub-scopes: all
Include within each scope: all

Store in database: (default)

Add to waveform display

5. Click *OK* to set the probe and close the form.

Tip

If you see the following message in the Console window, you have successfully created the probe:

```
ncsim> database -open waves -into waves.shm -default
Created default SHM database waves
ncsim> probe -create test_drink.top -depth all -all -shm
Created probe 1
```

Running the Simulation

To run the simulation:

1. From the any SimVision window, choose *Simulation – Run*. SimVision simulates the design and saves the simulation data in a default database. As it runs, the simulator displays the following messages in the Console window:

```
ncsim> run
                 400 loading machine with   5 cans
                 400 *** machine empty! ***
                 700 enter nickel
                 900 enter dime
                1100 enter quarter
                1300 enter dime
                1500 enter quarter
                1500 -> drink dispensed
-------------------------------
                1800 enter nickel
                2100 enter nickel
                2300 enter dime
                2500 enter dime
                2600 *** machine empty! ***
                2700 enter quarter
                2700 nickel changed
                3000 enter nickel
                3200 enter dime
                3400 enter quarter
                3600 enter dime
                3800 enter quarter
                3900 -> drink dispensed
-------------------------------
                  .
                  .
                  .
-------------------------------
               68400 enter quarter
               68700 enter nickel
               69000 enter nickel
               69200 enter dime
               69400 enter dime
Simulation complete via $finish(1) at time 69600 NS + 0
./test_drink.v:48                               $finish;
ncsim>
```

Tip

When you have completed these steps, your working directory should contain a new directory named `waves.shm`. The `waves.shm` directory should contain two files: `waves.dsn` and `waves.trn`. If these files are significantly smaller than 65,500 and 360 bytes, respectively, you did not probe all of the necessary objects during simulation.

To correct any problems, restart the simulator by choosing *Simulation – Reinvoke Simulator* from the Console window or by exiting the simulator and restarting it with the following command:

```
ncsim -gui worklib.test_drink:module
```

Then rerun the steps in this chapter.

Video

Now try this interactive exercise, Simulating the Design.

For More Information

This chapter describes how to set a probe and use the Console window to run the simulation. However, there are other ways to perform these steps, as follows:

Tool or Feature	Description

`$shm_open`
`$shm_probe`
`$shm_close`
	You can include calls to these system tasks in your Verilog source files. When you simulate the design, these system tasks open your database, set the objects that you want to probe, and close the database after the simulation has completed.
	
	See “Displaying Waveforms with the SimVision Waveform Viewer” in the *NC-Verilog Simulator Help* for details on these system tasks.

`database -open`
`probe -create`
`run`
	You can call these simulator commands from the Tcl command-line interface, or include them in a command file to simulate the design in batch mode.
	
	See the probe, database, and run commands in the *NC-Verilog Simulator Help*.

Tool or Feature	**Description**
Simulation database management	You can create different simulation databases for individual components of a design to help with debugging, or you can open simulation databases from other tools. See Chapter 6, “Managing Simulation Databases” in the *SimVision User Guide*.
Probes	You can enable, disable, and delete probes, or create new probes from the Properties window, and create probes automatically by sending signals to the Waveform window. See Chapter 7, “Creating and Managing Probes” in the *SimVision User Guide*.
Breakpoints	You might want to control where simulation stops by setting breakpoints. See Chapter 8, “Setting and Managing Breakpoints” in the *SimVision User Guide*.
Simulation control	You can run a simulation, stop at breakpoints, and step into and over subprogram calls. You can also save the state of a simulation and restart the simulation from that point. See Chapter 10, “Controlling the Simulation” in the *SimVision User Guide*.
Simulation Cycle Debugger	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 for Verilog-XL or AMS Designer. See Chapter 11, “Debugging at the Delta Cycle Level,” in the *SimVision User Guide*.

4

Displaying Simulation Data

Waveforms show the values of signals at any time during simulation. They can help you understand the behavior of your design.

To open a Waveform window:

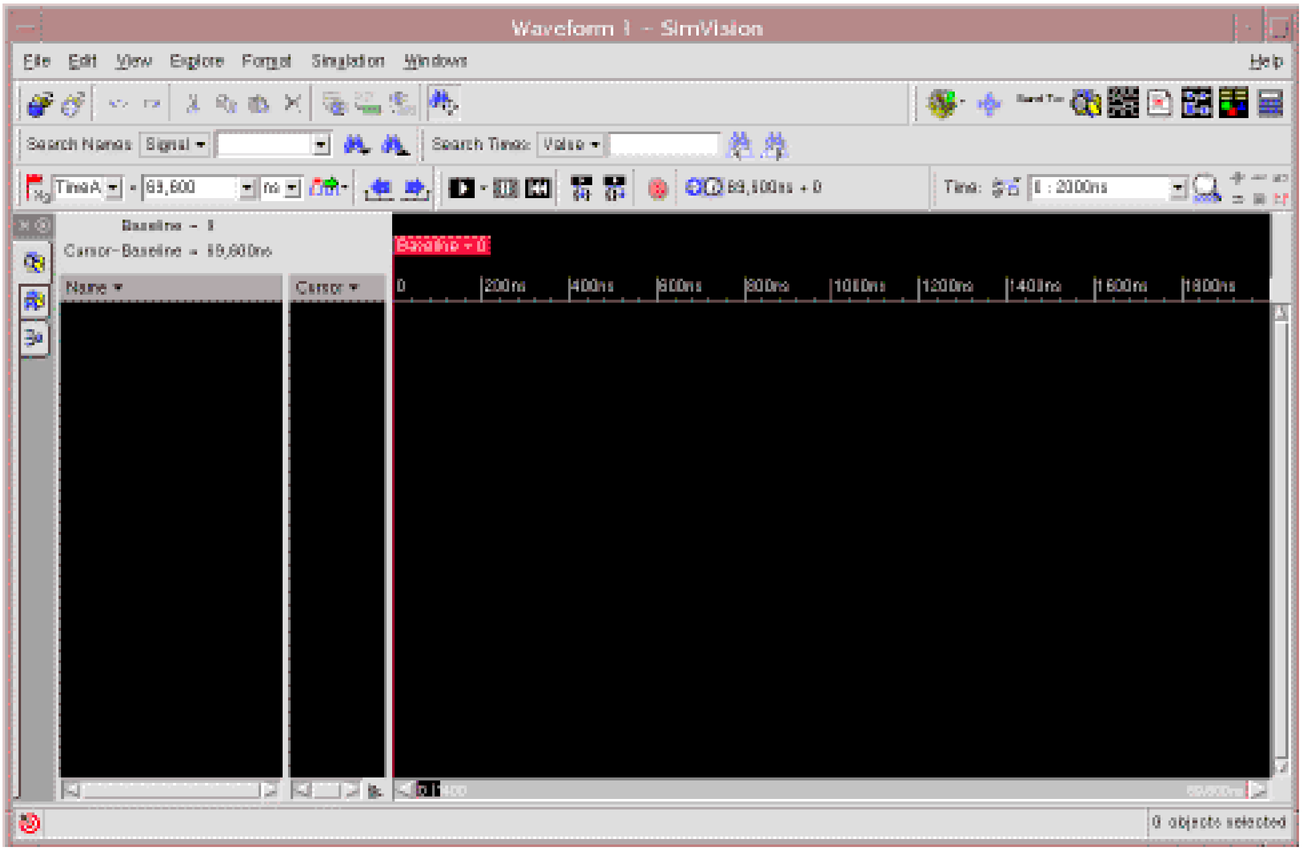
Deselect the `top` scope in the Design Browser sidebar, then click the *Waveform* button in the *Send to* toolbar. You can deselect the scope by pressing `Control` while clicking on the selected scope.

SimVision opens a blank Waveform window, as shown in Figure 4-1 on page 2.

Tip

If you select a scope before clicking the *Waveform* button, all of the signals in that scope are added to the Waveform window. If the scope has many signals, this may add more signals to the window that you want, and it may take a long time to load all of those signals and their waveforms into the window.

Figure 4-1 Opening a Blank Waveform Window



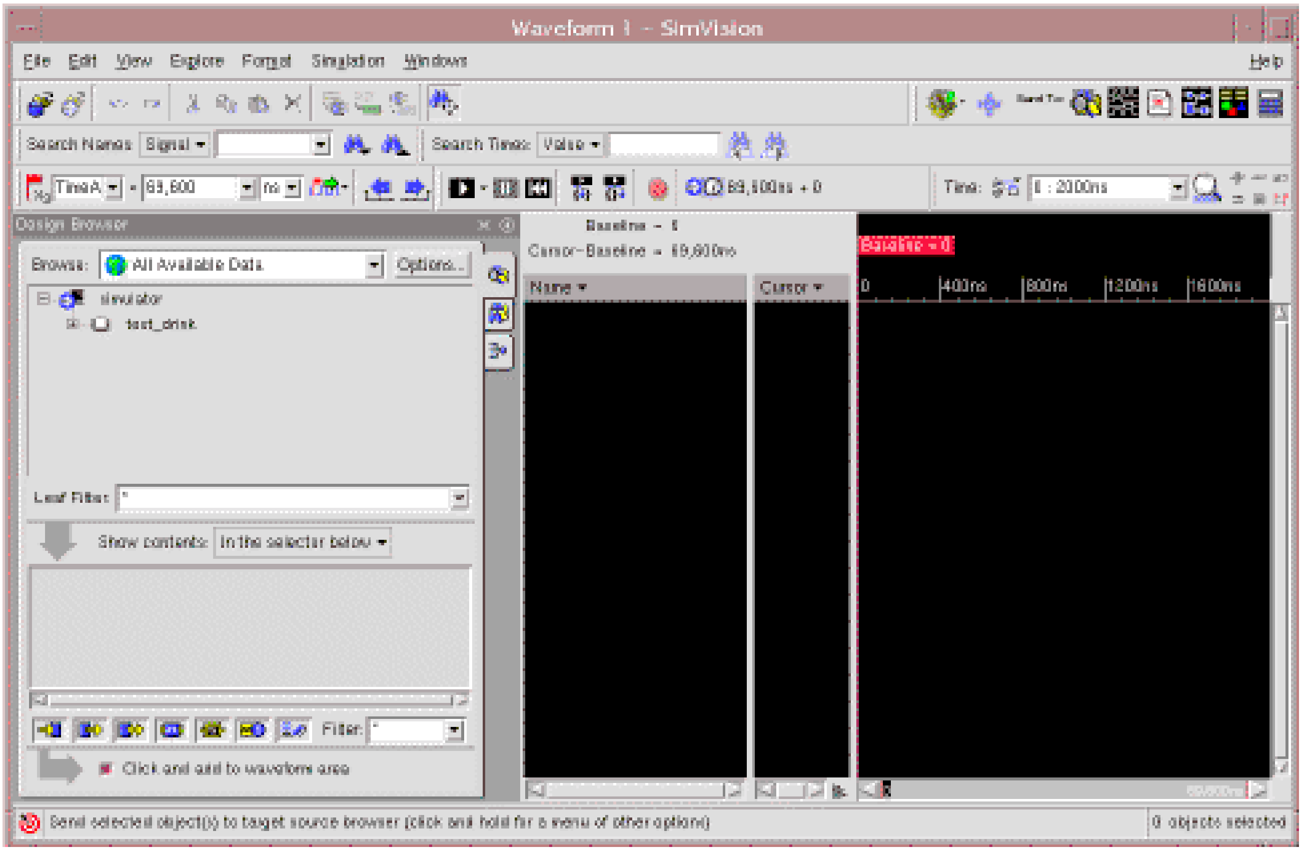
Selecting the Signals to Display

In the Design Browser sidebar, you can select objects from one scope at a time and send them to the Waveform window.

To select the signals that you want to display in the Waveform window:

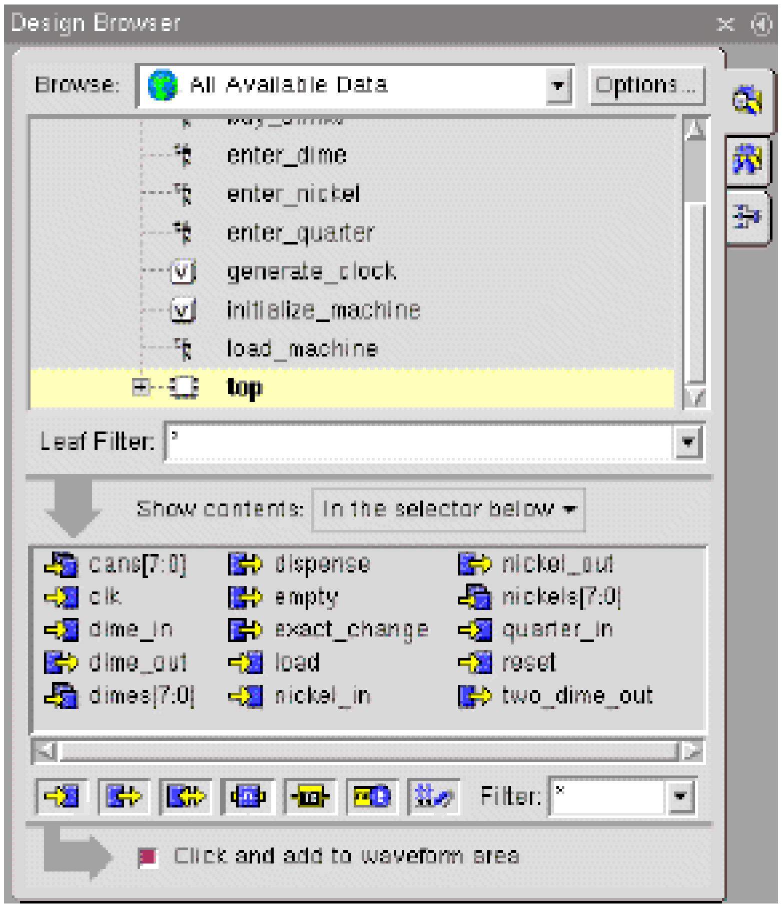
1. Expand the Design Browser sidebar by clicking the *Expand* button in the sidebar tab. SimVision adds the sidebar to the window, as shown in Figure 4-2 on page 3.

Figure 4-2 Expanding the Design Browser Sidebar


Waveform 1 - SimVision
File Edit View Explore Format Simulation Windows Help
Search Names Signal
Search Times: Value
TimeA 69,600 ns
69,600ns + 0
Time: 0 : 2000ns
Design Browser
Browse: All Available Data
Options...
simulator
test_drink
Leaf Filter: *
Show contents: In the selector below
Filter: *
Click and add to waveform area
Baseline = 0
Cursor-Baseline = 69,600ns
Name
Cursor
Baseline = 0
0
400ns
800ns
1200ns
1600ns
Send selected object(s) to target source browser (click and hold for a menu of other options)
0 objects selected

2. Expand the `test_drink` scope by clicking on the + button, then select the `top` scope. The sidebar displays the signals for that scope in the selector area, as shown in Figure 4-3 on page 4.

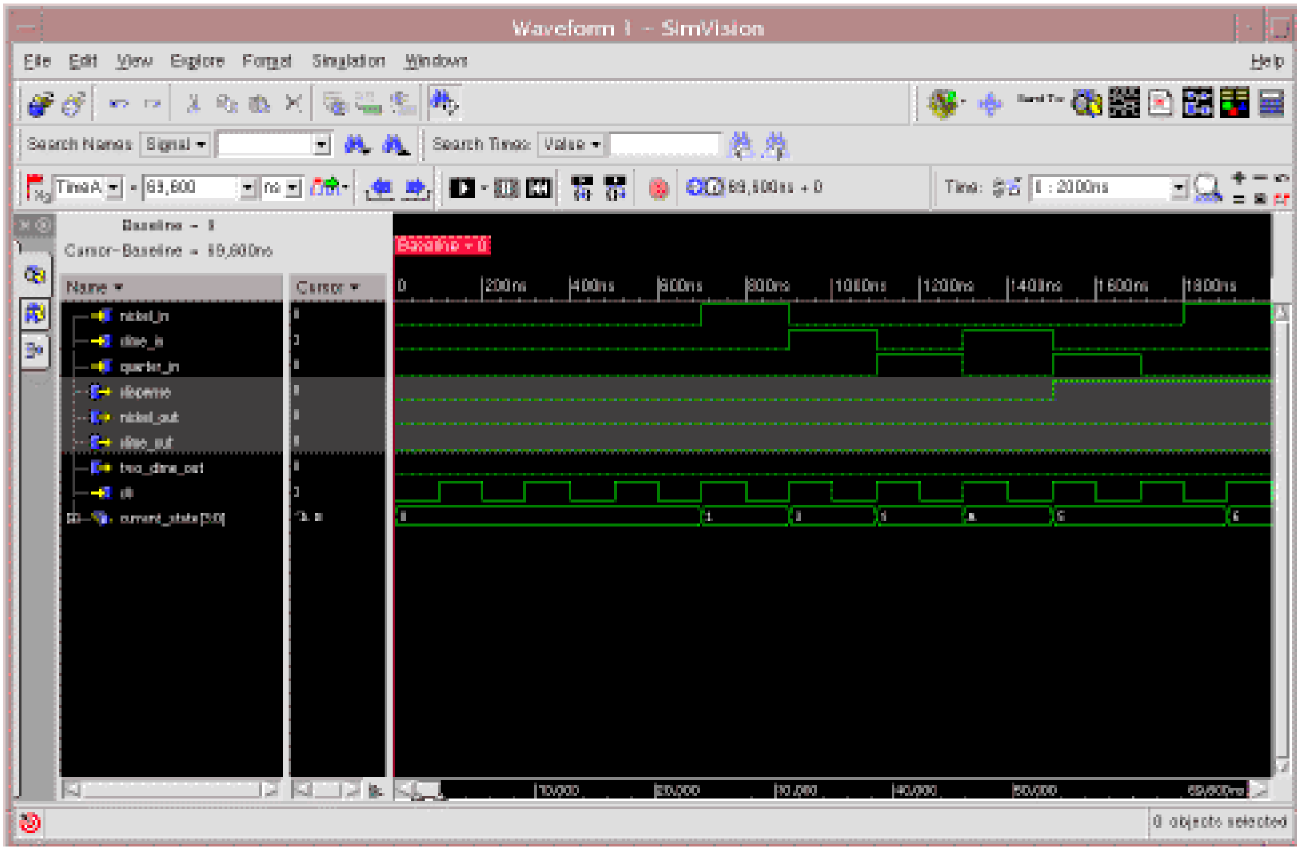
Figure 4-3 Showing the Contents of the top Scope



3. In the selector area, select the signals that you want to add to the Waveform window. For this sample session, select `nickel_in`, `dime_in`, `quarter_in`, `dispense`, `nickel_out`, `dime_out`, `two_dime_out`, and `clk`.
4. Expand the `top` module and select the `vending` scope. In the selector area, click on `current_state` to add that signal to the Waveform window.
5. Collapse the sidebar by clicking the *Collapse* button.

The Waveform window displays the signals and waveforms, as shown in Figure 4-4 on page 5. Signal names and values are displayed on the left; their waveforms are displayed on the right.

Figure 4-4 Displaying Data in the Waveform Window



Tip

SimVision adds the signals in the order in which you select them, but you can rearrange them. Select the signal that you want to move, then press and hold the middle mouse button. As you move the cursor, SimVision displays a red insertion bar. Place the insertion bar where you want the signal to appear, and release the mouse button.

Video

Now try this interactive exerecise, Selecting the Signals to Display.

Moving through Simulation Time

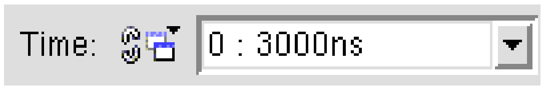
Above the waveform data, you can see the beginning and ending times for the simulation data currently displayed. Below the waveform data, the scroll bar shows the entire simulation time. You can adjust the amount of waveform data displayed in the window by entering a new time range.

To enter a new time range:

1. Enter a range in the *Time* field.

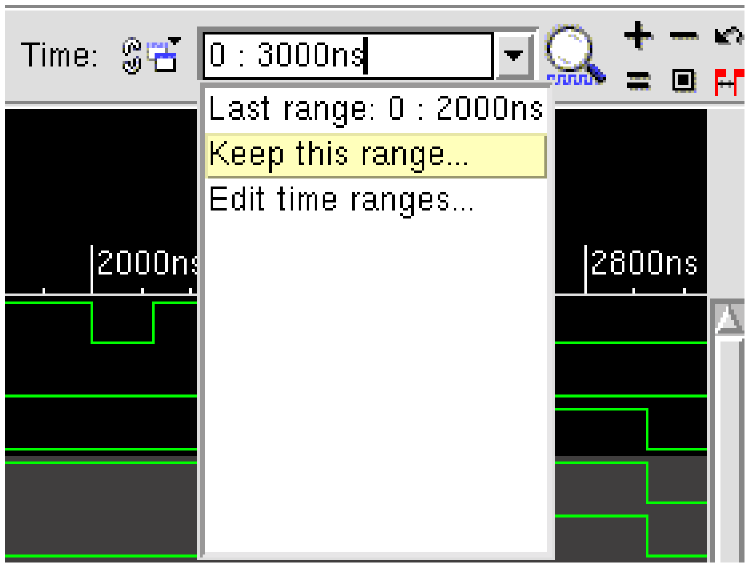
   For this example, enter `0:3000`, as shown in Figure 4-5 on page 6, and press `Return` to apply the time range.

Figure 4-5 Entering a New Time Range



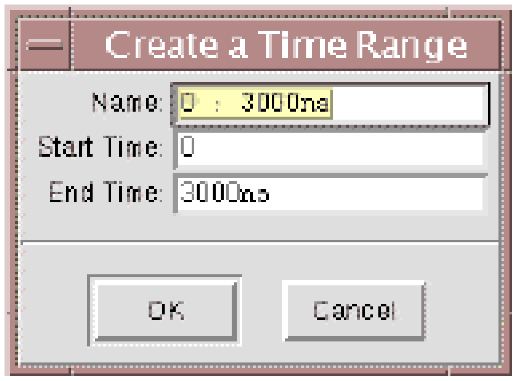
2. Save these settings by selecting *Keep this range* from the *Time* drop-down menu, as shown in Figure 4-6 on page 6.

Figure 4-6 Saving a Time Range



SimVision opens the Create a Time Range form, as shown in Figure 4-7 on page 7.

Figure 4-7 Create a Time Range Form



Name: 0 : 3000ns
Start Time: 0
End Time: 3000ns

3. You can either assign a name to the time range, or accept the default name. This name appears in the drop-down list of time ranges. At any time, you can quickly return to a view by selecting it from the drop-down list.

Video

Now try this interactive exercise, Managing Time in the Waveform Window.

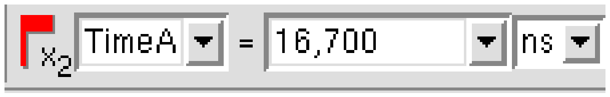
Moving the Cursors

The Waveform window contains two cursors, named `TimeA` and `Baseline`. You can move these cursors to any point in simulation time and use them as reference points. You can also create any number of additional cursors. However, for this example, you need only these two.

To move a cursor:

➤ Either drag the cursor to the desired time or enter a simulation time in the cursor time text field. For this example, change the simulation time of `TimeA` to `16,700` ns, as shown in Figure 4-8 on page 7.

Figure 4-8 Setting the Cursor Time



TimeA = 16,700 ns

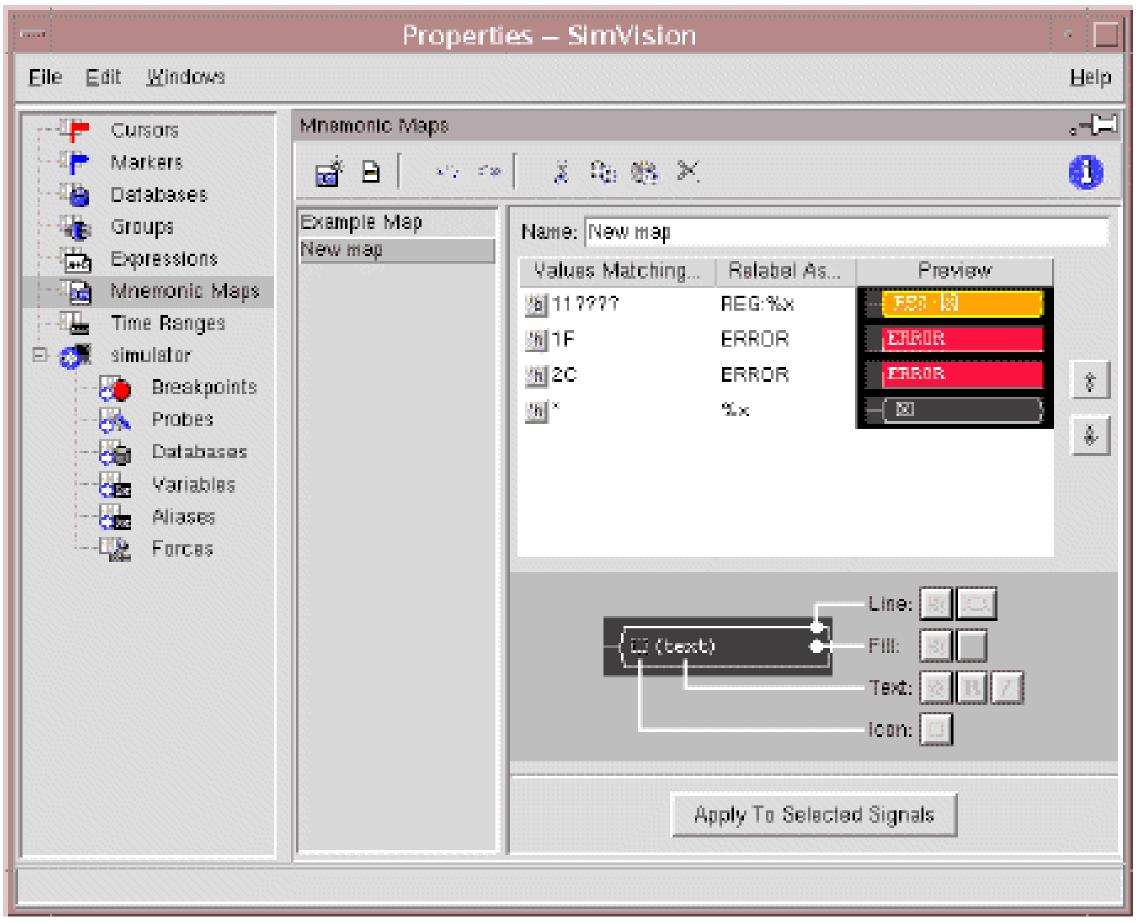
Controlling the Appearance of Waveform Data

When looking at a waveform, it is sometimes helpful to display the signal values as ASCII strings. For example, when the value of `current_state` is `1`, the user has deposited 5 cents in the machine; when `current_state` is 2, the user has deposited 10 cents. It can be helpful when viewing the waveform to display the `current_state` as `five` and `ten` rather than `1` and 2.

To map signal values to ASCII strings, define a mnemonic map:

1. Choose *Windows – Tools – Mnemonic Maps* to open the Properties window for Mnemonic Maps, as shown in Figure 4-9 on page 8.

Figure 4-9 Properties Window for Mnemonic Maps



Name: New map

Values Matching...	Relabel As...	Preview
11????	REG:%x	REG:%x
1F	ERROR	ERROR
2C	ERROR	ERROR
*	%x	%x

When you define a mnemonic map, you can 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.

2. Click the *New Map* button to create a new mnemonic map.

3. Define the first mnemonic map entry as follows:

a. To change the default radix, click on the `'h` entry and select `'d`.

b. Click in the *Values Matching* field, enter the value `0`, and then press Tab to move to the *Relabel As* field.

c. In the *Relabel As* field, enter the string `idle` and then press Tab to move to the *Values Matching* field of the next entry.

The *Preview* field shows what the Waveform window will display when the value of `current_state` is `0`. That is, it displays a box containing the string `idle`.

4. Define the following states:

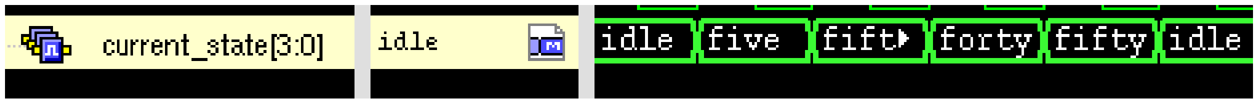
Values Matching	**Relabel As**
`0`	`idle`
`1`	`five`
`2`	`ten`
`3`	`fifteen`
`4`	`twenty`
`5`	`twenty_five`
`6`	`thirty`
`7`	`thirty_five`
`8`	`forty`
`9`	`forty_five`
`10`	`fifty`
`11`	`nickel_out`
`12`	`dime_out`
`13`	`nickel_dime_out`

Values Matching	Relabel As
14	`two_dime_out`

5. In the *Name* field, change `New map` to `Current State`.
6. Click *Apply To Selected Signals*.

   Now the values of `current_state` are displayed as `idle`, `five`, `fifty`, and so on, rather than as their decimal equivalents, `0`, `1`, `10`, and so on, as shown in Figure 4-10 on page 10.

Figure 4-10 Displaying Signal Values with a Mnemonic Map



7. Choose *File – Close Window* to close the Properties window.

Video

Now try this interactive exercise, Creating a Mnemonic Map.

For More Information

This chapter describes a few of the ways that you can display and organize information in the Waveform window. Other tools and features are available to help you display waveforms.

Tool/Feature	Description
Design Search sidebar	You can search for signals, variables, and scopes in multiple databases, without regard for the design hierarchy. See Chapter 4, "Accessing Design Objects," in the *SimVision User Guide*.
Groups and expressions	You can collect signals into groups and expressions. These features let you treat a set of signals as a single entity. See Chapter 15, "Organizing Signals into Groups, Expressions, and Comparisons," in the *SimVision User Guide*.

Tool/Feature	Description
Markers	You can place markers, similar to cursors, on the simulation timeline. You can jump from marker to marker to compare simulation results at different times, but you cannot use markers for measurement purposes. See Chapter 16, “Navigating through Simulation Time,” in the *SimVision User Guide*.
Measurement window	You can create a table of measurements to examine the characteristics of variables over a specified simulation time. For example, you can look at their slopes and minimum and maximum values. This is useful for measuring the properties of analog signal waveforms. See Chapter 20, “Measuring Signal Values,” in the *SimVision User Guide*.
Preferences	You can specify the style of toolbars in SimVision windows, set Waveform window defaults, select the text editor for the Source Browser, and otherwise customize SimVision. See Chapter 21, “Setting Preferences,” in the *SimVision User Guide*.

5

Debugging the Design

By defining conditions and analyzing the waveform data, you can see where errors occur in your design. When you have located an error, you can edit the design source code and reinvoke the simulator to test your changes.

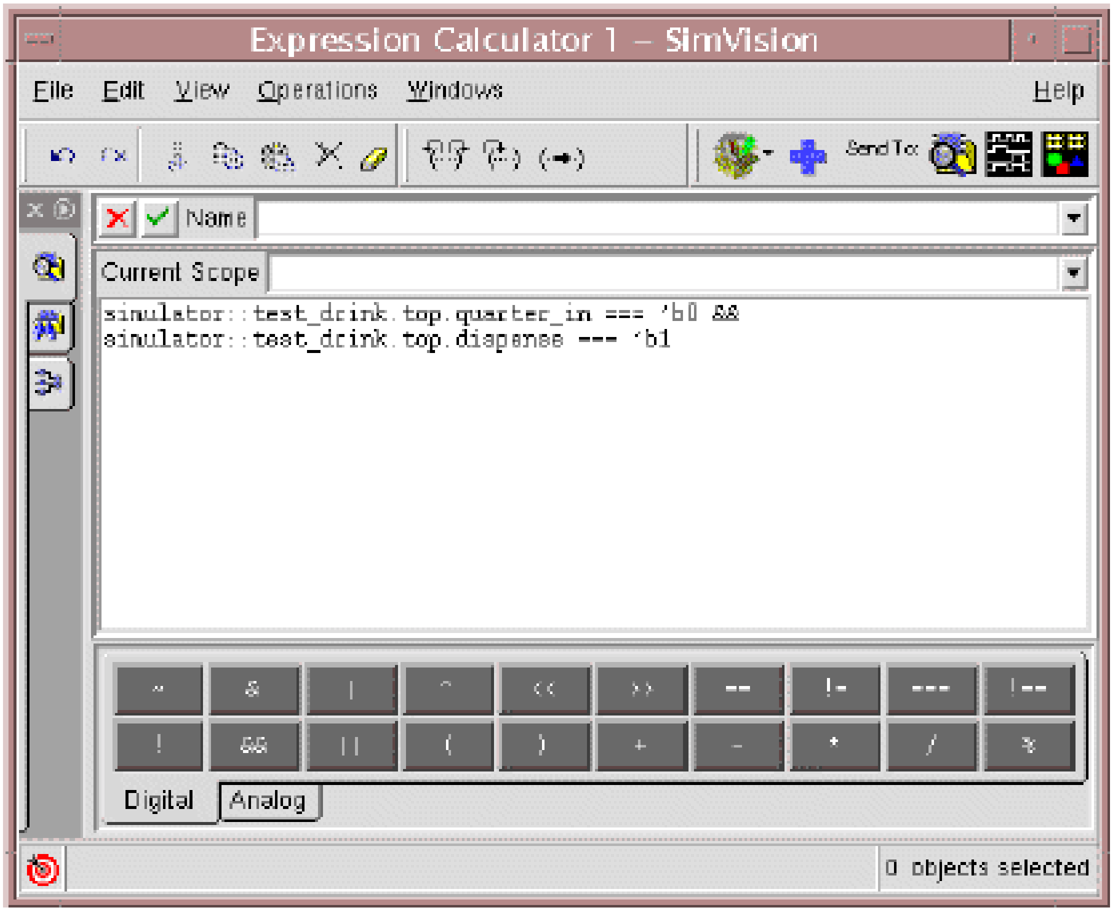
Searching for Conditions

A condition is a combination of signal values that you want to search for in the Waveform window. For example, you can define a condition that occurs whenever the `dispense` and `quarter_in` signals have the value 1, as follows:

1. Select the `quarter_in` and `dispense` signals in the Waveform window and click the *Next Edge* button until both signals have the value `1`.
2. Choose *Edit – Create – Condition*.

   SimVision opens the Expression Calculator, as shown in Figure 5-1 on page 2.

Figure 5-1 Expression Calculator



```
simulator::test_drink.top.quarter_in === 'b0 &&
simulator::test_drink.top.dispense === 'b1
```

The Expression Calculator creates a default AND expression. This expression is true whenever both signals have the value 1. You can edit this expression if you want to look for a different condition.

3. Enter a name for the condition expression in the *Name* field, such as `quarter_inANDdispense`.

4. Click the *Waveform* button to add the condition to the Waveform window.

   When you add a condition to the Waveform window, you can treat it like a signal and jump from edge to edge.

5. Choose *File – Close Window* to close the Expression Calculator.

6. Search for the expression to locate where the condition occurs.

Video

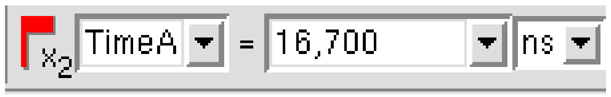
Now try this interactive exercise, Creating an Expression.

Analyzing Simulation Data in the Waveform Window

Analyzing waveforms can help you find problems in a design. For example:

1. Set the simulation time to 16,700 ns, as shown in Figure 5-2 on page 3.

Figure 5-2 Setting the Simulation Time



At this time, the machine is in the `idle` state.

Tip

There are several ways to set the simulation time:

❑ Enter the desired time in the cursor time field.

❑ Drag the primary cursor until it reaches the desired time.

❑ Select the `dispense` signal and click the *Next Edge* button until you reach the desired time.

2. Select the `current_state` signal and click the *Next Edge* button to follow the sequence of events from clock cycle to clock cycle:

❑ The user adds a nickel, and the machine transitions to the next state, `five`.

❑ The user adds a dime, and the machine transitions to the state `fifteen`.

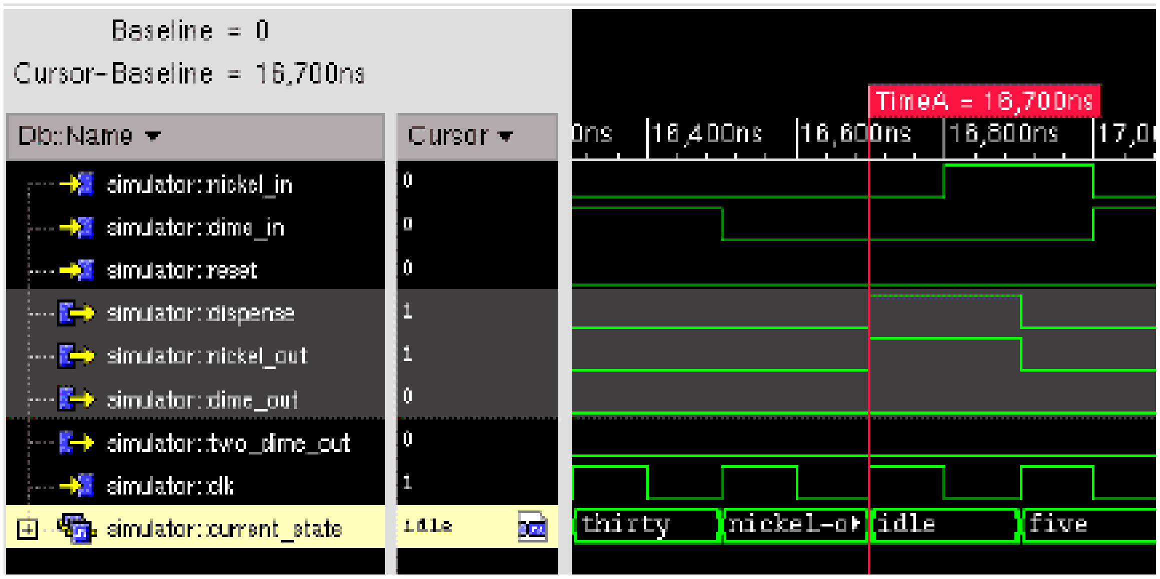
❑ The user adds a quarter, and the machine transitions to the state `forty`.

❑ The user adds a dime, the machine transitions to the state `fifty`.

During this clock cycle, the user also adds a quarter. You would expect the machine to transition immediately to the state `twenty_five`, but it does not. The machine transitions to the `idle` state, and sets the `dispense` signal to 1; the quarter is ignored. During the next clock cycle, the user deposits a nickel, and the machine transitions to the state `five`.

Figure 5-3 on page 4 shows the waveform at the point where the error occurs. Between the time the machine dispenses a drink and the machine returns to the `idle` state, the `quarter_in` signal goes high. The machine transitions to the state `five` in the next clock cycle, when the user deposits a nickel.

Figure 5-3 Locating the Error in the Design



3. From the *Time* field, choose *Keep this View* from the drop-down list so that you can easily go back to this view later.

Video

Now try this interactive exercise, Analyzing Waveforms to Find an Error.

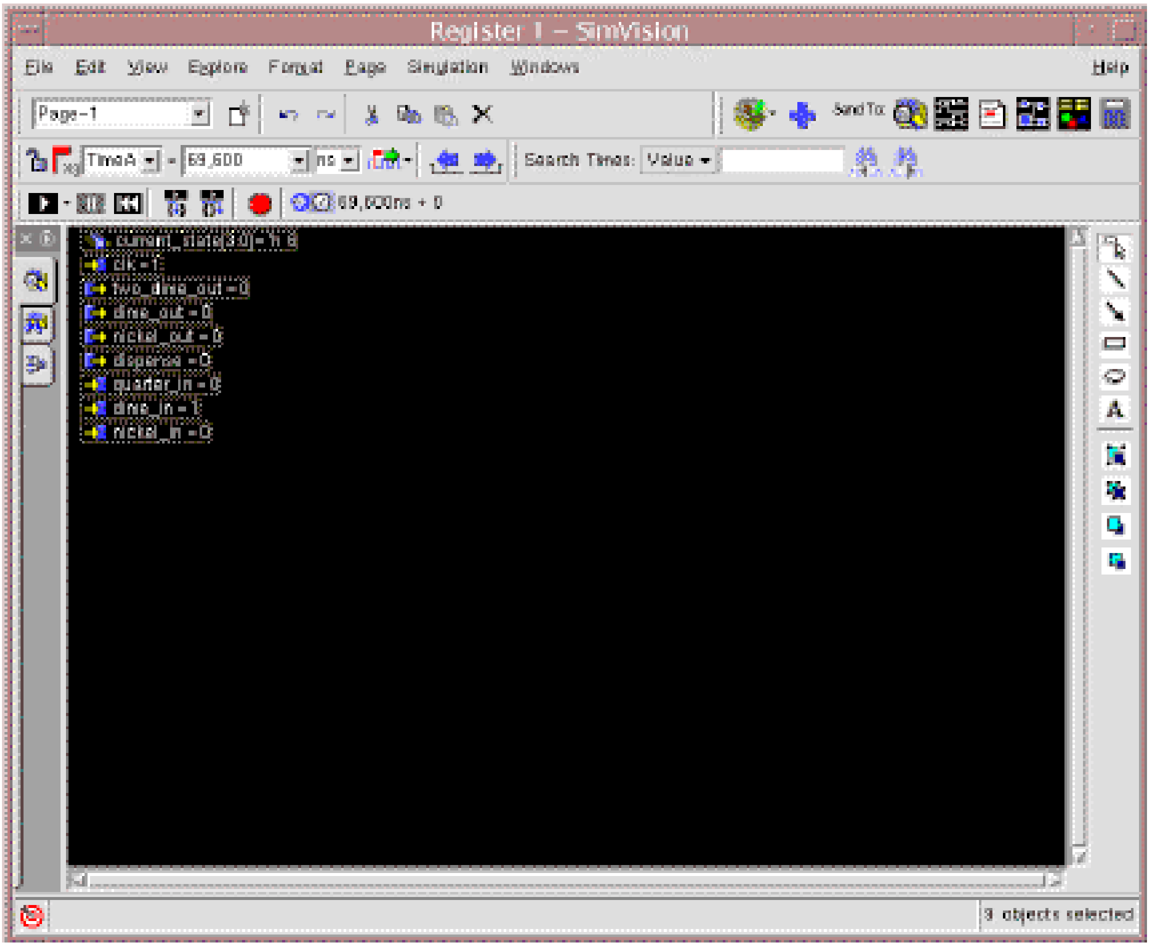
Analyzing Simulation Data in the Register Window

Another way to analyze simulation results is through the Register window, where you can create custom views of the simulation data, including freeform text and graphical elements. A Register window can have several pages, each with its own view.

To create a page in a Register window:

1. From the Waveform window, select the signals that you want to analyze, such as the `nickel_in`, `dime_in`, `quarter_in`, `dispense`, `nickel_out`, `dime_out`, `two_dime_out`, `clk`, and `current_state`.
2. Click the *Register* button to send these signals to a Register window, as shown in Figure 5-4 on page 5.

Figure 5-4 Adding Signals to a Register Window

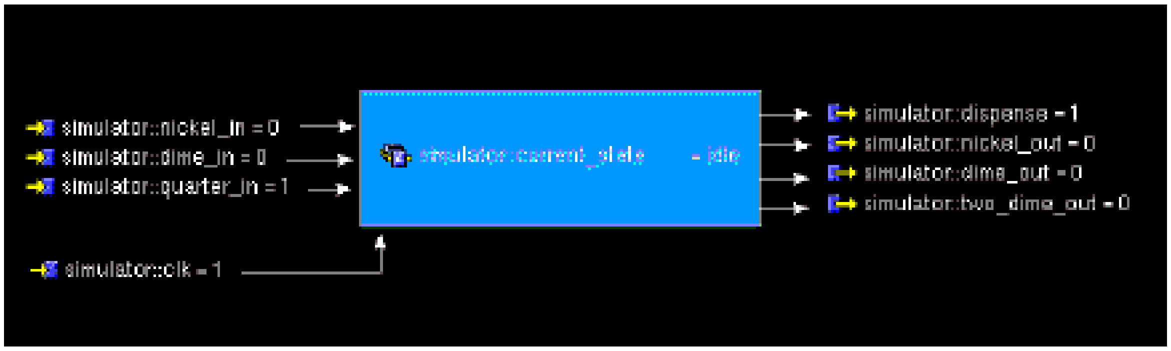


Along the right side of the window are buttons that let you draw graphical objects, add text, and manage the layout of the objects in the window. Tool tips pop up when you place the cursor over these buttons, telling you what functions they perform.

3. To apply the mnemonic map to the Register window, select the `current_state` signal and choose *Format – Radix/Mnemonic – Current State*.

4. Arrange the objects any way that you want. For example, the layout in Figure 5-5 on page 6 shows the relationship between the inputs—`nickel_in`, `dime_in`, and `quarter_in,` and `clk`—the state machine variable, `current_state`, and the outputs—`dispense`, `nickel_out`, `dime_out`, and `two_dime_out`.

Figure 5-5 A Custom Layout



5. Enter a simulation time, such as 16,700 ns, in the *Cursor Time* field.

   The Register window updates the signals to show their values at that time.

6. Select a signal, such as `current_state`, and click the *Next Edge* button.

   The time progresses to the next edge of that signal and the Register window updates all of the signals to show their values at that time.

7. Click the *Previous Edge* button to move the simulation time back to the previous edge of the selected signal.

8. Start at simulation time 16,700 ns, select the `current_state` signal, and click the *Next Edge* button to see the same sequence of events in the Register window that was shown in the Waveform window. That is:

   ❑ The user adds a nickel, and the machine transitions to the next state, `five`.

   ❑ The user adds a dime, and the machine transitions to the state `fifteen`.

   ❑ The user adds a quarter, and the machine transitions to the state `forty`.

   ❑ The user adds a dime, the machine transitions to the state `fifty`.

Video

Now try this interactive exercise, Analyzing Simulation Data in the Register Window.

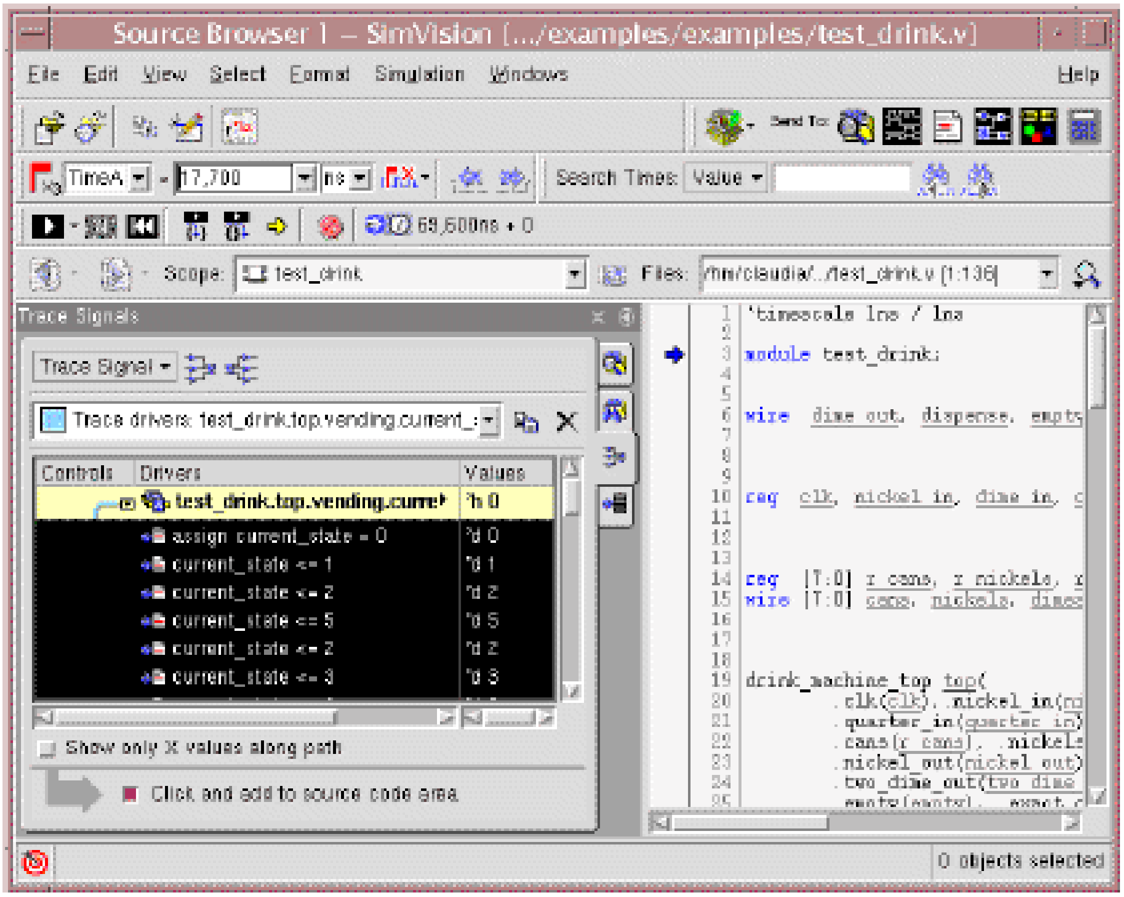
Fixing an Error in the Source Code

You can use the Trace Signals sidebar and Source Browser to locate the line in the source file where an error occurs, as follows:

1. In the Waveform window, and with the primary curset set to 17,700 ns, select the `current_state` variable and choose *Explore – Go To – Cause*.

   SimVision opens the Source Browser with the Trace Signals sidebar containing the signal you have selected. The sidebar lists the signal's drivers, as shown in Figure 5-6 on page 7. The Source Browser points to the beginning of the module where this signal is defined.

Figure 5-6 Displaying Drivers in the Trace Signals Sidebar


Source Browser 1 – SimVision [.../examples/examples/test_drink.v]
File Edit View Select Format Simulation Windows Help
Send To:
TimeA = 17,700 ns
Search Times: Value
69,600ns + 0
Scope: test_drink
Files: /hm/claudia/.../test_drink.v [1:136]
Trace Signals
Trace Signal
Trace drivers: test_drink.top.vending.current_s
Show only X values along path
Click and add to source code area
0 objects selected

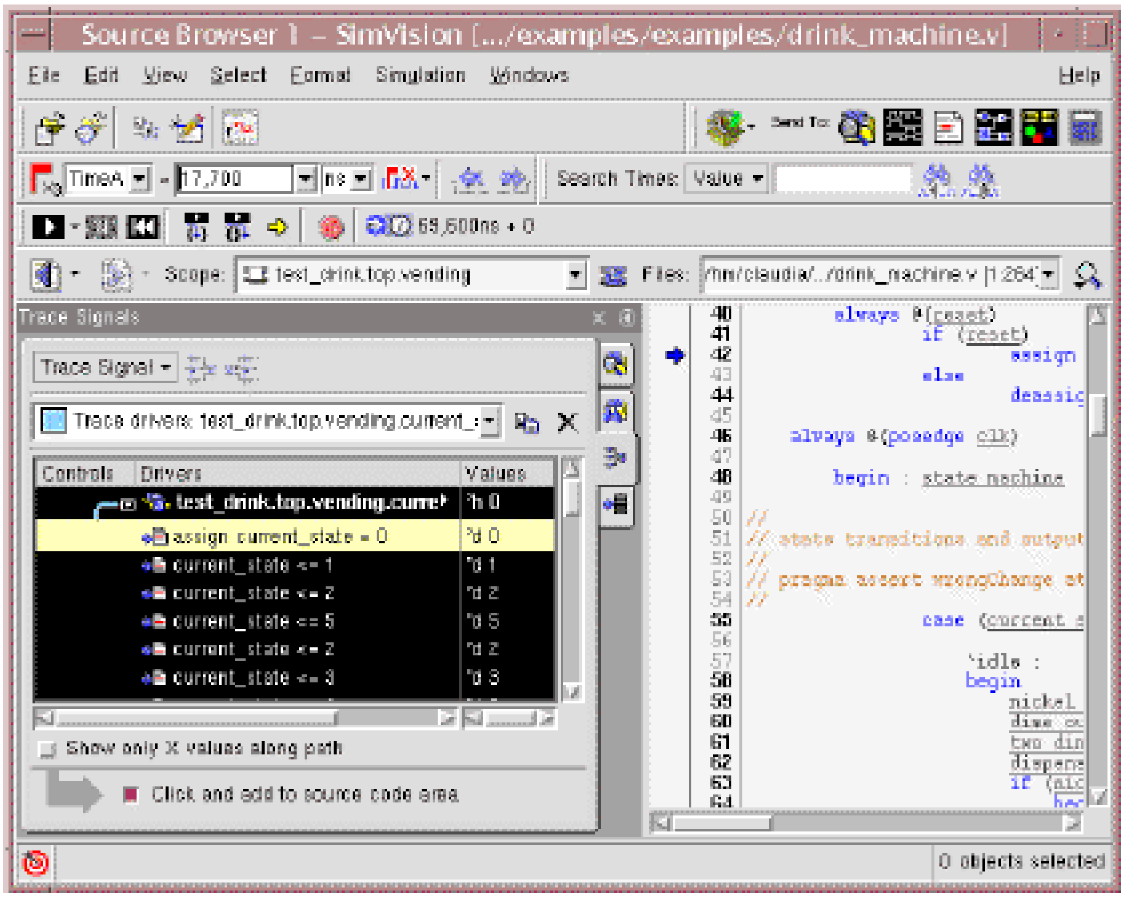
Controls	Drivers	Values
	test_drink.top.vending.curre	'h 0
	assign current_state = 0	'd 0
	current_state <= 1	'd 1
	current_state <= 2	'd 2
	current_state <= 5	'd 5
	current_state <= 2	'd 2
	current_state <= 3	'd 3

```
`timescale 1ns / 1ns

module test_drink;


wire  dime_out, dispense, empty



reg  clk, nickel_in, dime_in, c



reg  [7:0] r_cans, r_nickels, r
wire [7:0] cans, nickels, dimes



drink_machine_top top(
           .clk(clk), .nickel_in(ni
           .quarter_in(quarter_in)
           .cans(r_cans), .nickels
           .nickel_out(nickel_out)
           .two_dime_out(two_dime_
           empty(empty), exact_c
```

2. In the Trace Signals sidebar, select the first driver: `assign current_state = 0`.

   The Source Browser now points to the line in the source file where `current_state` is set to `'idle`, as shown in Figure 5-7 on page 8.

Figure 5-7 Displaying Source Code in the Source Browser


Source Browser 1 – SimVision [.../examples/examples/drink_machine.v]
File Edit View Select Format Simulation Windows Help
Send To:
Trace Signals
Trace Signal
Click and add to source code area

TimeA = 17,700 ns

Search Times: Value

59,600ns + 0

Scope: test_drink.top.vending

Files: /hm/claudia/.../drink_machine.v [1:264]

Trace drivers: test_drink.top.vending.current_s

Controls	Drivers	Values
	test_drink.top.vending.curre	'h 0
	assign current_state = 0	'd 0
	current_state <= 1	'd 1
	current_state <= 2	'd 2
	current_state <= 5	'd 5
	current_state <= 2	'd 2
	current_state <= 3	'd 3

Show only X values along path

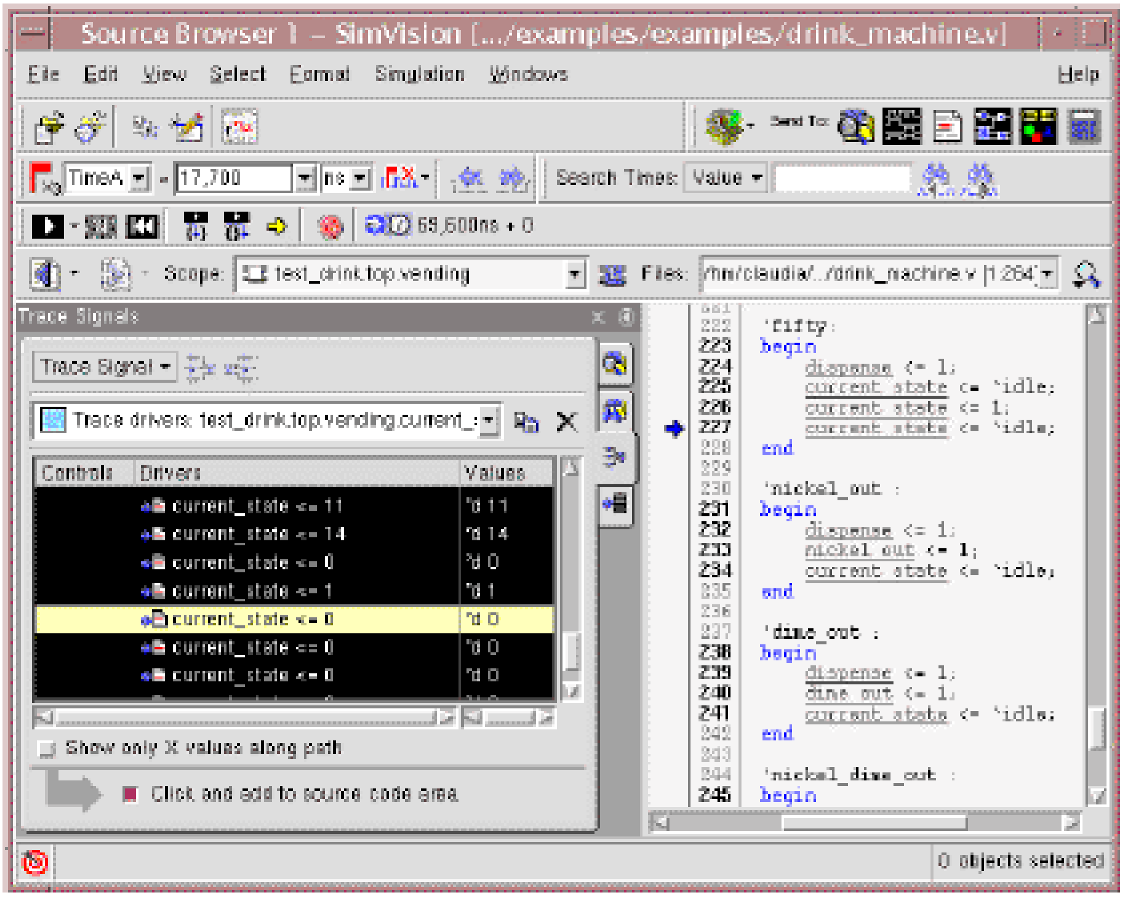
```
        always @(reset)
            if (reset)
                assign
            else
                deassig

    always @(posedge clk)

        begin : state_machine

//
// state transitions and output
//
// pragma assert wrongChange at
//
            case (current_s

                'idle :
                begin
                    nickel
                    dime
                    two din
                    dispens
                    if (nic
                    hex
```

0 objects selected

3. Scroll through the list of drivers in the Trace Signals sidebar, and you can see that there are several places where `current_state` is set to `0`. Click on these drivers to see the lines in the source code where these state transitions occur.

   Notice that the code for `'fifty`, shown in Figure 5-8 on page 9, sets the `current_state` to `idle`, but does not check to see if additional coins have been added.

Figure 5-8 Source Code for the Incorrect Case Statement


Source Browser 1 – SimVision [.../examples/examples/drink_machine.v]
File Edit View Select Format Simulation Windows Help
Send To:
TimeA = 17,700 ns
Search Times: Value
69,600ns + 0
Scope: test_drink.top.vending
Files: /hm/claudia/../drink_machine.v [1:264]
Trace Signals
Trace Signal
Trace drivers: test_drink.top.vending.current_
Show only X values along path
Click and add to source code area
0 objects selected

Controls	Drivers	Values
	current_state <= 11	'd 11
	current_state <= 14	'd 14
	current_state <= 0	'd 0
	current_state <= 1	'd 1
	current_state <= 0	'd 0
	current_state <= 0	'd 0
	current_state <= 0	'd 0

```
'fifty :
begin
    dispense <= 1;
    current_state <= 'idle;
    current_state <= 1;
    current_state <= 'idle;
end

'nickel_out :
begin
    dispense <= 1;
    nickel_out <= 1;
    current_state <= 'idle;
end

'dime_out :
begin
    dispense <= 1;
    dime_out <= 1;
    current_state <= 'idle;
end

'nickel_dime_out :
begin
```

4. To fix the error in the drink machine, choose *Edit – Edit File* from the Source Browser and add the necessary logic to the `case` statement for `'fifty`. For example:

```
'fifty :                                    //4'd10
begin
      dispense <= 1;
      if (nickel_in == 1)
           begin
                current_state <= 'five;
           end
      else if (dime_in == 1)
           begin
                current_state <= 'ten;
           end
      else if (quarter_in == 1)
           begin
                current_state <= 'twenty_five;
           end
      else
           begin
                current_state <= 'idle;
           end
end
```

Important

You can copy the above code and paste the solution into the source file. However, make sure that the tick marks face the correct direction after being pasted; otherwise the simulation will fail.

5. Save these changes to the file.
6. Choose *Simulation – Reinvoke Simulator* from the Source Browser.

   Note: If the Reinvoke dialog box appears, click *Yes*.

   SimVision compiles and elaborates the design, and restarts the simulator. All of the SimVision windows that were opened in the previous session are opened again, and the mnemonic map and condition that you created are still defined.
7. In any SimVision window, choose *Simulation – Run* to generate new simulation data.
8. In the Waveform window, display the view that you previously saved; you can see that the machine transitions correctly to state `twenty_five`.

Video

Now try this interactive exercise, Fixing an Error in the Source Code.

Ending a SimVision Session

To exit from SimVision:

1. Choose *File – Exit SimVision* from any SimVision window.
2. If the Waveform window remains open, choose *File – Exit SimVision* from the Waveform window.

   SimVision displays a confirmation message.
3. Click *Yes* to exit and close all SimVision windows.

For More Information

This chapter describes some basic steps for debugging a design. You may also find these other debugging tools and features helpful, as follows:

Tool/Feature	Description
Measurement window	You create a table of measurements for selected variables during a specified period of simulation time. For example, you can look at the value, slope, and minimum and maximum values of the variables, which is useful for measuring the properties of analog signal waveforms. See Chapter 20, “Measuring Signal Values,” in the *SimVision User Guide*.