EE4702 Informal Cadence Verilog Simulation Guide

Bryan Audiffred

February 19, 2004

1 Introduction

This brief guide should get you up and running with the Cadence Verilog simulator. It is by no means comprehensive. Please refer to the Cadence documentation for the exhaustive information. Part of being a good engineer is sifting through mountains of unreadable datasheets ☺. I can help you with getting started and performing the operations needed to succeed in the class, but please refer to the documentation first.

2 Goals

After reading this guide you should be able to:

1. Compile Verilog source
2. Simulate Verilog source
3. Interact with and debug a Verilog simulation
4. Analyze waveforms with SimVision

3 Setup

We will be using the following cadence tools for Verilog simulation, the NC-Verilog Compiler, SimVision interactive simulator, and SimVision Waves waveform viewer. Don't worry too much about the product names as they change every release cycle. All of the cadence software is located in the path `/opt/local/cadence` . The simulation tools are located in `/opt/local/cadence/LDV34`, and the documentation is in the `./doc` directory. A nifty documentation viewer is available by executing "cdsdoc".

All paths are now automatically set when you log on, so you should be able to execute all tools without trouble.

4 Creating Source Code

Verilog source is simply a text file. It goes without saying, but NC-Verilog will not read in formatted Microsoft Word documents. The standard text editor in the CDE dock will be dtpad. You may launch it from the dock or from the command line. Be aware that moving text documents from windows or mac may introduce undesired effects. You can use the `dos2unix` command if you would like to convert a windows text file to unix ASCII encoding. You may of course use any editor you wish, no doubt most of you will.

5 Compiling Source Code

We will be using NC-Verilog. It is Cadence's flagship HDL simulation product. The "NC" refers to native compiled, a technique used to greatly enhance simulation speed, a large concern of customers. As an example, we will compile the following inverter module, "myinv".

```
‘timescale 1ns/100ps

module myinv();
 reg a;
 wire b;

assign b = ~a;

 initial begin
   a=0;
   #10;
   $display("a=%d,  b=%d",a,b);
   a=1;
   #10;
   $display("a=%d,  b=%d",a,b);
 end
endmodule
```

You may compile the code with the command `ncverilog myinv.v`. The compiler (Verilog is really an interpreted language, but there's no point quibbling over semantics) will elaborate (interpret) the verilog then move onto a native compilation task that isn't very important to this guide. The `ncverilog` command actually calls three different commands in order. Please refer to the reference manual for all of the details. Your errors will occur in the elaboration stage. Since we have passed no command line arguments other than the file, a simulation will immediately run. This is obviously not the most useful way to debug a design. Also note that a myriad of files will be created in the current directory.

5.1 The Command Line Simulation

To engage in an interactive command line simulation, invoke:
`ncverilog -s +ncaccess+rwc <filename>`
This will stop the simulator at time zero. The access flag allows you to read, write, and change values in the design. No access will speed up the simulation, but this will not be an issue for this class. There are useful commands you may use to debug the design from the command line. A few of them are:

- `run` → "Wesely, engage." (How bad was that Star Trek reference?)
- `run -step` and `run -next` → step one line over or into subroutines
- `stop -time 5` → stop sim after 5 time units. also -show and -delete options
- `describe` → list info about a signal
- `force mySignal 1` → force mySignal to 1. opposite is release.
- `reset` → reset simulation to zero

You may find help on any command with the command `help`.

5.2 The Graphical Tools

Command line simulations are fine for short runs, but you will likely find the graphical environment more useful at first. Once you become a power user, you may revert back to manual commands, but you will find menus and buttons easier in the beginning. Predictably, the menus and buttons

execute commands in the program's TCL shell (you can see the commands). That is a good way to learn commands.

Invoke your simulation with the command:

```
ncverilog +gui +ncaccess+rwc <filename>
```

This will bring up the GUI environment. There will be three primary windows, the "Cadence NC Verilog" window that has your code and command shell, the "Navigator" window that shows the hierarchy and signals in your design, and the wave window to add signals. The wave window might not be present at first.

Let us walk through a simple example:

1. Invoke the tool on our sample inverter (make a file first - myinv.v) with the command line:
   ```
   ncverilog +gui +ncaccess+rwc myinv.v &
   ```
   You should see a window with code and a command shell. We will call this the main window. Press the massive VCR play button, and you will get two lines of output as expected.

2. Now have a look at the navigator window. If it is not visible, select it from the main window's "Windows" menu. You will see the module, and it has two signals. These are the current values of the signal. Use the "Step Over" button in the main window to step through the code. Note the time values in the command shell. You will notice something like "Stepped to 10 NS + 1". This means that you have gone through one evaluation cycle after 10 ns, Notice that b = a is not always true until time has advanced.

3. Now let us add some waves. Reset the simulation (menu or command line). Select both signals in the navigator and press the waveform viewer button. Simvision will start, and the signals "a" and "b" will be present. Press run and you will see a plot of the waveforms. I will leave the buttons up to you, but one useful one is the magnifying glass with the equals sign. It automatically zooms to fit the entire wave.

Despite the view of your code in the main window, it is only a cruel trick. You may not edit the source. There is an edit option from the file menu. The default editor is vi. You may change this in the options menu. Perhaps you would prefer dtpad, so enter "dtpad %F" for the editor command. After editing, you must perform the seemingly unnecessary step of reinvoking your simulation from the file menu.

6 Basic Debugging

For some simulations, debugging will consist of running the entire simulation and looking at all the waveforms. There are many more detailed options available. Two useful functions are forcing and breakpoints. A force statement will, as implied, force a signal to a value despite its real driver. Breakpoints operate like a normal program and stop the simulation when a particular statement or condition is reached.

To force signals, go to the Navigator window. Select the signal and press the force button (looks like a hammer). Choose any value. Also from the Navigator menu you may select the breakpoint button. This will stop the simulation whenever the signal changes - very convenient.

To breakpoint an arbitrary line you must add the option "+nclinedebug" to the end of your command line. You will get a performance warning, but you will be able to stop at arbitrary lines. You may right click on the line of source (actually I have only one mouse button, so I don't know if it is the right or middle button on a sun mouse) and set a breakpoint or use the Show→Breakpoints menu in the main window to set a breakpoint.

7 Conclusion

The best way to learn any software package is to play around with it. My recommendation is that you take this guide and spend an hour pressing buttons and trying out new functions of the software. Run your own sample design through it, and see what you can do.