14:332:437 Concepts in Digital Systems Design
Instructions for Using Verilog with the Synopsys System

M. L. Bushnell

January 20, 2005

1 Coding of Verilog for Synthesis

Please observe the following rules when you code Verilog for synthesis:

1. I have found that the Synopsys system does peculiar things when I update the stack pointer in a pushdown stack design from last year. This was due to the fact that I was updating the stack pointer, and then referencing into the stack with it. However, since the stack pointer was assigned to with the non-blocking signal assignment statement, it was not updating immediately, and therefore the stack pointer could never range from Boolean values 0 through 16. Instead, it could only have Boolean values 0 and 1. As a result, the logic synthesizer optimized away most of my stack registers! Therefore, in a process where you change a pointer, and then use the pointer to access/modify a register, assign to the pointer with the blocking assignment statement. This will allow you to change the pointer, and then use its new value to access other things, without having to account for logic delay. The synthesis tools should generate a correct design for this.

2. The Synopsys system has the property that when it reads in your Verilog description, it first generates reams of hardware implementing your design in a rote fashion, without logic optimization. As a result, the initial design looks ridiculous. It will boil down to something reasonable after you optimize the logic (but not always).

2 Status of the Synopsys System

The Synopsys logic synthesis tool, *design_analyzer*, and the verilog simulator, *vcs*, have been updated in the CAIP research center and in the ECE Department. The Synopsys system (Verilog logic synthesis, Verilog behavioral simulation, and Verilog logic simulation) is now fully operational. We have just added **virsim_exec**, the interactive viewer and debugger. In order to use it, you must also recopy Bushnell's files into your account. Also, the TAs and I will be glad to show you how to use the tool. In order to understand the complexity of the system, you must realize that one copy of it sells for $ 1,000,000. We have 100 copies, and this represents the largest cash donation ever made to the Rutgers University Foundation.

Both the new **design_analyzer** and the new verilog simulator **vcs** support the 2001 updated verilog syntax described in your textbook.

3 Operating the Synopsys Design Analyzer for Synthesis

- Log into any one of the Sun Sparcstations in the ECE Department laboratories. You can use *tomsriver*, *sandyhook*, *redbank*, or any of the other work stations. You need a powerful Sparcstation because the Synopsys system is very compute intensive. In order to find out the actual name of your client, type:

  ```
  echo $DISPLAY
  ```

- Initial Setup (Do this only during the first time that you use the Synopsys system), or when you wish to upgrade to the latest versions of *design_analyzer* and *vcs*, the verilog simulator.

  1. Before using Synopsys tools from an ECE machine, please first copy these files from Bushnell's account into your home directory:

     ```
     cp ~bushnell/.synopsys_sge.setup .
     cp ~bushnell/.synopsys_dc.setup .
     cp ~bushnell/.synopsys_vss.setup .
     cp ~bushnell/.cshrc .
     ```

- Synthesizing Hardware Using the Synopsys System

  1. Text edit or copy your verilog file in the directory of the account where you will do logic synthesis. Your verilog files must have the extender ".v".
  2. Maximize the size of the window that you will run the Synopsys system in.
  3. Type **design_analyzer** to start the Synopsys system on a Sun workstation. The **design_analyzer** is capable of Verilog error checking, Verilog logic synthesis, and Verilog hardware optimization.
  4. Click on "Setup" and then on "Defaults". In the form, blank out the "Link Library" and the "Symbol Library" blanks. Set the "Target Library" blank to "gtech.db", which means that you are synthesizing logic to be fabricated using the general technology logic library. Finally, click on the "ok" button.
  5. Click on "File" and "Read" to read in your Verilog source code file and check it for errors. Click on "Ok" to actually read the file. Expect many errors to be reported to you in your code. You must fix these errors with a text editor before proceeding further. We offer free consulting about what is wrong with your Verilog description. Repeat this Step until there are no errors in your Verilog file.
  6. Once you have gotten the Verilog read in without significant errors, click on "Cancel" to dismiss the form and go back to the main menu. There should be an *equation box* with the name of your topmost design entity showing in the main window.
  7. Double click on the *equation box* with your left mouse button to see the symbol (icon) for your top-most design entity. Make sure that the type and directionality of all of your design ports is correct. Double click on the entity symbol to see all of your logic gates. This will take some time, since logic must now be synthesized for your design. Expect to see a laughable amount of hardware being used to implement your design. This happens because no attempt has yet been made at logic synthesis – the Synopsys system just instantiates simple hardware templates for every Verilog statement. If you have trouble reading all of that hardware synthesized for your simple design, then click on "View"

and then on "Zoom In" to zoom in on it. There is also a "Zoom" command available in the background of the hardware window.

8. Now that you are thoroughly disgusted with the synthesized hardware, do logic optimization. Click on "Tools", and then on "Design Optimization". Call for "High Mapping Effort" and "Allow Boundary Optimization", except for large designs where this will take a prohibitive amount of time. This command will take a long time and generate voluminous output on the screen. Pay attention to all of the complaints about your hardware. We offer free consulting service to help you fix these hardware problems. You must correct ALL of these problems, or the synthesized hardware will not be correct. If the hardware Verilog description must be updated to correct errors, go back to the step where you read in the file. Repeat that and all subsequent steps.
9. Click on "Cancel" to dismiss this form.
10. If you do not now see your optimized hardware, repeat Step 8 until it appears. In many cases, this hardware will now be as good as what you could design by hand. In other cases, it is still unacceptably wasteful of hardware. If that is the case, you must now rewrite your Verilog description into a lower-level one (i.e., neither *behavioral* nor *data-flow*) in order to "command" the Synopsys system to design the hardware the way that you want it. Correct the Verilog file and repeat all previous steps. At this point, you can explore the design by using *design_analyzer*. You can see the critical path, get timing information, highlight all cells or references of a given type, change circuit constraints, group and ungroup subsets of the circuit, and get a variety of reports.
11. In the "File" menu, click on "Plot" to plot your hardware to a postscript file. If your design is huge, this will print the whole thing on one 8 1/2 X 11 sheet of paper, which is useless. There appears to be NO schematic tiling command. Instead, use the zoom command to magnify the design so that you can read it. Then, you must manually print out every sheet of the magnified design. Use the scroll bars at the bottom and right of your screen to position each part of the design on the screen. Then use the "File", "Plot", and "Current View" commands to print this out. Don't forget to edit the printer command blank in the form to put the correct printer command in for YOUR printer. Queue this file on your favorite black-and-white laser printer using the "Plot" command. Turn in a printout of your Verilog source code and the printed schematic of the synthesized hardware.
12. Write out the final correct logic design by using the "File"/"Save As" menu commands. Enter the new filename in the "File Name" field, for example, *decoderlogic.v*. Select the output format from the "File Format" pick list as "Verilog netlist" output format.
13. Exit the design analyzer.

4 Verilog Simulation Using the Synopsys Simulation Tools

Behavioral Simulation. Add a test bench to the behavioral verilog code, as described in the lecture notes. Without a test bench, the simulation will not yield any interesting results. Type this command:

```
vcs verilog_file
```

The verilog system will compile your hardware description into C code, link it, and leave an executable file called **simv** in the directory where you ran **vcs**. If you wish to use the latest syntax for verilog, you must type this, instead:

```
vcs +v2k verilog_file
```

In order to execute the simulation, type:

```
simv
```

The simulator will execute and produce a report on the screen showing all of your variables that you asked to be printed out with **$monitor**. In order to save this report to a disk file, type:

```
simv > file_name
```

I am still having trouble with **vcs** when using **#(parameter ...)** statements, but all other statements seem to work.

Example for Problem 1, Problem Set 1. We provide the correctly coded verilog test bench for the decoder for this problem as an example. The code has successfully compiled with the verilog for the decoder and correctly simulated.

```
module testdec (input  [7:0] a, output reg [2:0] s, output reg oe);

initial
  begin
    $monitor (10,, "s = %b, oe = %b, $a = %b", s, oe, a);
    s = 0;
    oe = 0;
    #10 s = 1;
    #10 s = 2;
    #10 s = 3;
    #10 s = 4;
    #10 s = 5;
    #10 s = 6;
    #10 s = 7;
    #10 oe = 1;
    #10 $finish;
  end
endmodule

module tbench;
  wire [7:0] a;
  wire [2:0] s;
  wire oe;

  decoder (s, oe, a);
  testdec (a, s, oe);
endmodule
```

Using the Graphical Waveform Display Module. We have also installed **virsim_exec**, the interactive waveform viewing module. In order to use it, please recopy the setup files into your account from Bushnell's account.

In order to see graphical waveforms, do the following:

1. In your testbench, insert the **$dumpvars;** statement and the **$dumpfile ("myfile");** statements (at the beginning of the **initial** block in the testbench). These will cause the simulator to dump all simulation variables when each **$monitor** statement executes into a file in your directory called *myfile.vpd* or it may just write to *myfile*.

2. Run **vcs** in the normal way.

3. Type **simv** to generate the normal output from **$monitor** statements and to create the waveform display file.

4. Type the command: **virsim_exec**

5. A window will pop up. Click on the **Waveform** button with the left mouse key.

6. Select **File Open** with the mouse.

7. In the file name box, type in *myfile.vpd* or *myfile* or select the waveforms file from the list of files, and click **OK**. The program may demand that you change the file format between **VCS** and **VCS2**.

8. Click on the **Hierarchy** button in the main window.

9. Click on your testbench module with the left mouse button in the *VirSim* window.

10. A list of your simulated signals will pop up. Click on each signal set that you want to see with the middle mouse key and drag its icon into the main display window.

11. The signal waveforms will appear in the waveform display window. Double click on each signal set that you want expanded with the left mouse button, and you will now see waveforms for the individual bits in the signal set.

12. The remaining keys are self-explanatory, and you can learn to use the tool by experimenting with it.

Logic Simulation. The logic simulation process proceeds almost exactly as for behavioral simulation, with one important exception. After you have read in your design and optimized it with **design_analyzer**, use the **File** $\rightarrow$ **Save As** menu item to save the logic that was synthesized in a verilog file. You must change the file format from **.db** to **Verilog** format, and you must give the file a unique name. After that, please edit the file and insert the contents of the disk file **/caip/u8/Synopsys/gtech_lib.v** at the very front of your logic file. This will define the behavior of every one of the logic gates in your logic level Verilog file, so that **vcs** can simulate it. After that, simply copy your testbench and your **system** module that instantiates both the design and the testbench into the end of the logic file. Finally, you can now run **vcs** just as for behavioral simulation, and it should give correct results. Warning: Logic simulation can run ten times longer than behavioral simulation on large designs.

5 VHDL Simulation Using the Synopsys Simulation Tools

Required Synthesis Steps for Simulation: Before going any further, it is assumed that you have done the following during synthesis:

1. There are new versions of *.synopsys_dc.setup* and *standard_config.vhdl* that are necessary to make simulation work. Please recopy this files from "~bushnell" into your directory.

2. Copy the file "~bushnell/vhdl/standard_config.vhdl" and use it for your configuration file for logic simulation. Please remember to change the file name to reflect the name of your design, and please change the configuration name of the file to reflect the name that you want for your configuration. Within the file, change *some_circuit* to the name of your design entity. If you intend to do only *behavioral* simulation, then change the name *SYN_some_architecture* to the name of your architecture, without the *SYN_* prefix. If you intend instead to do *logic* simulation, then change the name *SYN_some_architecture* to the name of your architecture, WITH the *SYN_* prefix. Any logic-level file produced by synthesis will have your original behavioral architecture name prefixed with *SYN_* as the architecture name of the *synthesized* logic.

3. Save the synthesized VHDL output file from the Synopsys synthesis tools in a file in VHDL format. Let us call it *decoder_logic.vhdl*

4. It is assumed that you have now created a configuration file for your design. Let us call it *decoder_config.vhdl.*

5. Ignore all complaints about the missing "gtech.sdb" file. That warning does not matter.

6. When you run *vhdlan* and *vhdldbx*, you must be in EXACTLY the same directory as the one where your VHDL source code was. Also, for each VHDL design, put it in a different subdirectory, so that VHDL does not get confused by the various different configuration files.

Setup Steps for Simulation: This need only be done once, and after that, all simulations will work correctly.

1. Enter the following commands into your *.cshrc* file before trying to use any of the simulation tools. You must also type the command *source .cshrc* to cause these changes to take effect.

```
set path = ($path $SYNOPSYS/sparcOS5/sim/bin)
setenv PATH ${PATH}:$SYNOPSYS/sparcOS5/sim/bin
source /caip/u8/synopsys.1997.08/admin/install/sim/environ.csh
```

   Alternatively, copy the *.cshrc* file from *~bushnell/.cshrc* This is the most painless way to get your account set up.

Behavioral Simulation: The Synopsys system can simulate your design at the *behavioral* or *data flow* level, which means that it simulates the original VHDL code that you entered **BEFORE** logic was synthesized. This is by far the easiest way to simulate your design. Follow these steps:

1. Run the VSS analyzer on your VHDL design files. The command is:

```
vhdlan decoder.vhdl
```

   Always analyze the files bottom up, i.e., the architecture files should be analyzed first and the top level design file should be analyzed last. For this, you mercifully do NOT need a configuration file.

2. Run the debugger on the top level VHDL configuration file. The command is:

   ```
   vhdldbx
   ```

3. Before the debugger comes up, you'll see a window showing the libraries in a text box on the left and the designs in a text box on the right. Select the library as "DEFAULT" and the design as the name of the top level design architecture within your design entity. Let us call it "your_architecture. Set the time units as NS (nanoseconds) and click on the "OK" button.

4. Now the debugger should come up EVENTUALLY. For reasons we do not understand, some idiot at Synopsys wrote this simulator to copy 1022 font files into your account. You may wish to go study for an exam while the work station laboriously copies all of those fonts.

   - The GUI for vhdldbx is very flakey. Do not use the GUI buttons, or use them as little as possible. Also, the waveform viewer is slightly fragile. If you unexpectedly click on buttons in a weird way, it goes away. You can try to restart it by clicking on "Trace" in the background menu, but if that does not work, then you must restart *vhdldbx*. If you have trouble with the waveform viewer crashing (which it is prone to do), please do not use the Graphical User Interface (GUI) to enter simulation commands. Instead, type the commands in the bottommost window on your screen. This will avoid GUI problems.
   - **Signal Tracing.** You can see the values in signal lines by selecting (with the left mouse button) the VHDL text declaring the ports in *entity* declaration for your VHDL design (shown in the window) that you wish to look at. Type *trace* <*signal_name*> in the bottommost box in your vhdldbx window to cause the signal to be traced. This will start up the Synopsys waveform viewer. Type as many trace commands as you like.
   - **Signal Values – Setting.** You can set values on a selected signal line by typing *assign (value) signal* in the bottommost box in your vhdldbx window. For *value*, use the appropriate VHDL expression for the value you wish to assign to *signal*. This is actually easier than using the GUI. You can release signal values by *release signal* in the box to release the signal from the value it was previously set to. For multiple-bit signal buses, you have to type the binary value for the bus in double quotes in the box.
   - **Running the Simulation.** Run the simulation by typing *run #* to simulate for that many nanoseconds of time units. For any other command that you wish to issue to the simulator, try it the first time with the GUI, and see what shows up in the bottommost box. After that, just type the command into the box. DO NOT USE THE GUI – IT WILL CRASH AFTER A SMALL NUMBER OF CLOCK PERIODS.
   - **Viewing Simulation Results.** Use the waveform viewer to see the results of the simulation. Click on "View" and "Full Fit" to see all of your waveforms. Double click on the green icon next to the bus that you wish to view in order to see an expanded version of the bus. You can also resize the waveform viewer to make it taller so that more information in captured in the window.
   - **Interactive Simulation.** You can do multiple logic simulations, by changing the signal values that you wish to change, and then typing "run" again. The timing diagram will show all of the signals that are the result of your simulation so far. You should simulate as many input signal combinations for the design as are necessary to convince yourself that the design hardware is working correctly.

- **Printing Waveform Traces.** In the "File" menu of the waveform viewer, click on "Print To: File:" to write out your timing waveform trace to a postscript disk file. Fill in the file name in the form provided. You should write out the waveform trace after every clock period, so that if the simulator GUI crashes, you still have the work that you have already done. Click on "ok" to finish writing the file.
- When you are done, exit the *vhdldbx* tool, and queue the postscript file to your favorite printer (ee103) with this command:
  lpr -P ee103 <postscript_file_name>
  Turn in waveform traces for all designs simulated as part of your homework. Since you cannot reload the waveforms after you have written them to a file, it is o.k. (if the waveform viewer crashes) to restart vhdldbx and produce a separate waveform timing diagram for the cases that you were unable to simulate earlier. Turn in timing diagrams on multiple sheets.

Logic Simulation. This is far more difficult than behavioral simulation, because it requires configuration files to map the synthesized logic produced by logic synthesis onto the *gtech* library.

1. **Configuration Files.** When simulating a synthesized logic file, you must include, in your configuration file, only those components that are actually used in the final logic. You must edit the configuration file so that the logic gates mentioned in it EXACTLY match the logic gates used in your logic level design. I apologize for this feature of VHDL. Simply comment out the logic gates in the configuration file that are not used in your design. You can find that out by looking at the *component* statements in the synthesized VHDL statements. The component name appears right after the word *component*. The simulation tools will give you trouble until your configuration file refers to exactly those components actually used in your synthesized logic. If any components used in the VHDL logic file are missing in the configuration file, then simulation also will not work. It will also not work if your configuration refers to components that are NOT in the VHDL logic file. This is a pain in the neck. Edit your configuration file to refer to exactly those components in your synthesized VHDL logic file.

2. **Test Benches.** You have the option of creating a test bench file for simulating your machine. A test bench gives instructions and input vectors for validating the behavior of your design. If you wish, instead, to use the interactive simulation system, then you must ensure that there is a *process* described in your synthesized logic VHDL file. Otherwise, the simulation system simulates the logic once for you, forgets the present state of the simulation, and then exits. Either edit the synthesized logic VHDL file to add a *process*, or create a test bench (to be described in a subsequent memo).

3. Run the VSS analyzer on your VHDL design files. The command is:

   ```
   vhdlan decoder.vhdl
   vhdlan decoder_config.vhdl
   ```

   Always analyze the files bottom up, i.e., the architecture files should be analyzed FIRST and the top level configuration file should be analyzed LAST.

4. Run the debugger on the top level VHDL configuration file. The command is:

```
vhdldbx
```

5. Before the debugger comes up, you'll see a window showing the libraries in a text box on the left and the designs in a text box on the right. Select the library as "DEFAULT" and the design as the name of the top level configuration. Let us call it "standard_config". If you do not select the configuration here, simulation will not work. Set the time units as NS (nanoseconds) and click on the "OK" button.

6. Now the debugger should come up.

   - **Signal Tracing.** This operates as described earlier.
   - **Signal Values – Setting.** This operates as described earlier.
   - **Running the Simulation.** This operates as described earlier.
   - **Viewing Simulation Results.** This operates as described earlier.
   - **Interactive Simulation.** This operates as described earlier, with the following exception. In logic simulation mode, you can only do one logic simulation, and then you must type *execute* and then *restart* to restart the simulation. You do not need to reselect lines for tracing, but you DO need to reselect the lines whose input values you wish to change for the next simulation. You should simulate as many input signal combinations for the design as are necessary to convince yourself that the design hardware is working correctly.
   - **Printing Waveform Traces.** This operates as described earlier.

Freeing Up Disk Space: If you are running short on disk space, you should delete all *.sim files (the compiled simulation models) and all *.mra files (the files stating how to link compiled simulation objects into **vhdldbx**). If you still have no disk space, delete *.db files, BUT MAKE SURE that you have saved all synthesized logic that you want in VHDL logic files.