14:332:431 Digital Systems Design
Instructions for Using the Synopsys System

Ganapathy Parthasarathy and M. L. Bushnell

September 3, 2001

1 Coding of VHDL for Synthesis

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

1. When you use *library ieee* in the Synopsys system, please put the statement *use ieee.std_logic_arith.all;* into your code. This will overload the + and - operators for **std_logic** signal types to do integer addition/subtraction. This differs from the arithmetic library used in your book.

2. Completely avoid using *ports* of type *buffer*. The problem is that many of the Synopsys synthesis component libraries cannot completely handle these. Instead, declare internal *variables* that will be set to the signals that you want to both read and write in your design. Declare the external *ports* that will be tied to these variables as type *out*. Then, at the appropriate points in your processes, copy the internal variables to these external ports. This eliminates the problem of using port type *buffer*.

3. 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 of type *signal*, 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, declare the pointer as a *variable* and not as a *signal*. 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. You can then copy the *variable* to a *signal*, and then the synthesis tools should generate a correct design.

4. There is a special clause that you can attach to an enumerated type state declaration to explicitly declare the state assignment. Normally you would say

   ```
   type states is (s0, s1, s2, s3, s4, u1, u2, u3);
   ```

   to define a set of states *s0* ... *u3* for your state machine. In this case, the Synopsys system chooses your state assignment. If, instead, you wish to control the assignment, do this:

   ```
   signal present_state, next_state: std_logic_vector (2 downto 0);
   constant s0: std_logic_vector (2 downto 0) := "000";
   ```

```
constant s1: std_logic_vector (2 downto 0) := "010";
constant s2: std_logic_vector (2 downto 0) := "001";
constant s3: std_logic_vector (2 downto 0) := "100";
constant s4: std_logic_vector (2 downto 0) := "011";
constant u1: std_logic_vector (2 downto 0) := "101";
constant u2: std_logic_vector (2 downto 0) := "110";
constant u3: std_logic_vector (2 downto 0) := "111";
```

5. The Synopsys system has the property that when it reads in your VHDL 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 system (logic synthesis, VHDL analysis, and VHDL logic simulation) is now fully operational in the computer engineering laboratories in the CORE Building and will shortly be operational in the ECE Building. Also, the TAs and I will be glad to show you how to use the tool. In order to understand the complexity of the program, you must realize that one copy of it sells for $ 1,000,000. We have 50 copies, and this represents the largest cash donation ever made to the Rutgers University Foundation (approximately $ 5 million).

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).*

  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 .
     ```

     In the sub-directory where you want to do logic synthesis, copy the file **decoder_synth_conf.vhdl** *from the* **b̃ushnell/vhdl** *home directory.*

  2. *If you do not want to use my .cshrc file, then you must add the paths* **/opt/synopsys/sparcOS5/syn/bin** *and* **/opt/synopsys/sparcOS5/sim/bin** *to your binary path search list. See file* **b̃ushnell/.cshrc** *for an example of how to do this.*

- *Synthesizing Hardware Using the Synopsys System*

1. *Text edit or copy your vhdl file in the directory of the account where you will do logic synthesis. Your VHDL files must have the extender ".vhdl" or ".vhd".*
2. *Edit the configuration file called* **decoder_synth_conf.vhdl** *in the directory and change the name of the "decoder" entity in the file to the name of the entity that you are describing in VHDL. Save the modified configuration file as* **<your-entity-name>_synth_conf.vhdl** *immediately. The configuration file tells the Synopsys system which Boolean functions are available as primitive logic gates. If you design something that requires a logic gate not present in your configuration file, expect problems and see me. The sample configuration file will be updated from time to time as we solve these problems.*
3. *Maximize the size of the window that you will run the Synopsys system in.*
4. *Type* **design_analyzer** *to start the Synopsys system on a Sun workstation. The* **design_analyzer** *is capable of VHDL error checking, VHDL logic synthesis, VHDL logic simulation, and VHDL hardware optimization.*
5. *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.*
6. *Click on "File" and "Read" to read in your VHDL 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 VHDL description. Repeat this Step until there are no errors in your VHDL file.*
7. *Once you have gotten the VHDL 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.*
8. *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 VHDL 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.*
9. *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 VHDL description must be updated to correct errors, go back to the step where you read in the file. Repeat that and all subsequent steps.*
10. *Click on "Cancel" to dismiss this form.*

11. *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 VHDL description into a lower-level one (i.e., neither* behavioral *nor* dataflow*) in order to "command" the Synopsys system to design the hardware the way that you want it. Correct the VHDL 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.*
12. *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 VHDL source code and the printed schematic of the synthesized hardware.*
13. *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.vhdl*. Select the output format from the "File Format" pick list as "VHDL netlist" output format.*
14. *Exit the design analyzer.*

4 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.