Computer Architecture
CPSC 32, Fall Semester 2003
Lab Assignment #3
Due: Week of October 27–October 31, demonstrate in your lab,
complete by yourself

Make yourself familiar with the Verilog programming environment on the CS UNIX machines. Attending the lab sessions is **strongly recommended**. The class homepage contains links to some introductions to Verilog. Appendix A of the book by Brown and Vranesic contains an overview as well.

The first assignment is mainly based on structural Verilog, meaning that the modules are composed of elementary logic gates such as AND, OR, NAND, etc. A hierarchical design builds new components from old ones. For instance, you can build a multiplexor from NAND gates, and then a full-adder using these multiplexors.

Note that you can write your testbenches in behavioral Verilog.

1 Assignment

[**30 points**] Describe a multiplexor based on NAND gates, without using any other gates, such that `f=a` if `select=0`, and `f=b` otherwise.

```
module mux(a, b, select, f)
input a, b, select;
output f;
...
endmodule
```

Write a testmodule using a `initial begin` ... `end` block to display the function table `a,b,select,f` for all inputs `(a,b,select) = 000`, `001`,.... Hint: Use `$display` or `$monitor`.

[**30 points**] Design a one-bit full-adder based on two `xor` gates and one `mux`.

```
module add(a, b, cin, sum, cout)
input a, b, cin;
output sum, cout;
...
endmodule
```

Write a testmodule using a `initial begin` ... `end` block to display the function table `a,b,cin,sum,cout` for all inputs `(a,b,cin) = 000`, `001`,....

[**20 points**] Write a 8-to-1 multiplexor `mux3` based on `mux`. Write a testbench to illustrate the behavior of this module.

[**20 points**] Write a 32bit adder `addern` based on `add`. Write a testbench to illustrate the behavior of this module.

2 Getting Started

Hints by Praveen Bhojwani on the usage of Synopsis Verilog.

```
(0) Log on to your CS Unix account
(1) Launch the bash command shell (bash). At the bash command line
    source the 'synopsys.shell.source.bash' script, as follows:
bash-2.03$ . /usr/local/bin/synopsys.shell.source.bash

 Note: do this ONCE as soon as you enter the bash shell.

(2) You are ready to run the Verilog compiler 'vcs' or the graphical
debugging system 'virsim'
 - use
bash-2.03$ vcs -help
   or
bash-2.03$ virsim -help
   to obtain a review of the command line options and arguments to
   invoke these programs. VirSim is a graphical tool that helps us
   debug/simulate and inspect graphically the internal state of our
   models as they are being simulated.

 - Documentation can be found under
/usr/local/synopsys/vcs6.2/doc/UserGuide/
   for vcs, and
/usr/local/synopsys/virsim_4.2.R2/doc/
   for virsim
(3) Exiting
Simply exit the bash shell. Note that the init script sets up a number of
environment variables that Synopsys tools use and it extends the $PATH
greatly.
(4) To restart using Synopsys tools you need to start from (1) above.
```

3 Dishonesty

Make sure that you complete the assignment by yourself. Do not copy the code from others, nor provide others with your code. Refrain from copying and modifying the code from other sources.