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 $\text{select}=0$, 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 $a,b,\text{select},f$ for all inputs $(a,b,\text{select}) = 000, 001, \ldots$. 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 $a,b,\text{cin},\text{sum},\text{cout}$ for all inputs $(a,b,\text{cin}) = 000, 001, \ldots$.
Write a 8-to-1 multiplexor \texttt{mux3} based on \texttt{mux}. Write a testbench to illustrate the behavior of this module.

Write a 32bit adder \texttt{addern} based on \texttt{add}. Write a testbench to illustrate the behavior of this module.

## 2 Getting Started

Hints by Praveen Bhojwani on the usage of Synopsis Verilog.

1. Log on to your CS Unix account
2. Launch the bash command shell (bash). At the bash command line
   source the \texttt{'synopsys.shell.source.bash'} script, as follows:
   
   
   \begin{verbatim}
   bash-2.03$ . /usr/local/bin/synopsys.shell.source.bash
   \end{verbatim}

   Note: do this \textbf{ONCE} as soon as you enter the bash shell.

3. You are ready to run the Verilog compiler \texttt{vcs} or the graphical debugging system \texttt{virsim}
   - use
   
   \begin{verbatim}
   bash-2.03$ vcs -help
   or
   bash-2.03$ virsim -help
   \end{verbatim}

   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
     \texttt{/usr/local/synopsys/vcs6.2/doc/UserGuide/}
     for \texttt{vcs}, and
     \texttt{/usr/local/synopsys/virsim_4.2.R2/doc/}
     for \texttt{virsim}

4. 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 \texttt{PATH} greatly.

5. To restart using Synopsys tools you need to start from \textbf{(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.