Due Tuesday October 1, 2015. To be done individually.

1. In this problem you will synthesize a digital design of an 8b×8b multiplier from behavioral HDL code to a netlist. If you have another design that you would prefer to use instead of the multiplier, there may be flexibility; email the instructor. You may use Verilog or VHDL to implement your design. However, I will be better able to offer help on Verilog designs.

As you work through the homework, you are encouraged to add/modify items on the wiki page that could benefit from edits, additions, or clarifications.

(a) Implement your design in the HDL of your choice. For a multiplier, this should be a very simple file.
(b) Write a tester for your design that will allow you to provide digital stimuli to test either a verilog or transistor-level implementation of your design.
(c) Simulate the test bench to verify that the HDL design works correctly. Include a plot of this simulation in what you turn in.
(d) Following the synthesis instructions on the class wiki, synthesize your design to a structural gate-level design. From within Synopsys, print a schematic view of your synthesized design.
(e) Import this design into Cadence. Run the same simulation from step 1c but using your gate-level design. For this simulation, the gates should be simulated using their HDL descriptions. So the multiplier is modeled as a collection of gates, but each gate is modeled by an HDL description rather than transistors. Verify that the design still works. Include a print-out of the simulation results and note the simulation time. At this point, you should also be able to measure delay.
(f) Repeat the simulation from the previous step, but this time, edit your configuration to have the gates modeled at the transistor level. Measure the delay and note the simulation time.
(g) Turn in the multiplier schematic from Synopsys, a schematic of your top-level test bench, a plot of the simulation results, and a print-out of the HDL code for both your multiplier and the test driver.

2. Draw the schematic for a 4b×4b radix-4 Booth multiplier. For the final carry-propagate adder, you can use a carry-ripple adder. Use logical effort to estimate the delay of your multiplier in terms of fanout-of-four (FO4) delays.

- One FO4 delay is the delay required for an inverter to drive four identical inverters. It is a commonly-used metric of the speed of a process as well as a commonly-used normalization to describe the speed of a block in a technology-independent way. Your logical effort analysis will find the delay in terms of “unit delays,” where an unloaded inverter has a delay of two units. So you will need to estimate the FO4 configuration in unit delays in order to convert your multiplier delay from unit delays to FO4 delays.

3. Design an SRAM cell. Demonstrate that it both a '0' or a '1' can be written to and read from the cell. You may choose to write an HDL tester for the cell, but this is not required. Devise a test bench to measure the static noise margin, the read margin, and the write margin. You can use a pair of inverters as the sense amp.