1. Given the D-FF circuit shown in Figure 3.13 of Harris & Harris, and the clock and D waveforms of Figure 3.60, sketch waveforms that you would expect at nodes N1, N2, and Q.

2. Given the D-FF circuit above, modify it to include each of the following capabilities. Implement each separately. You may use only NMOS and PMOS transistors. You should not have two transistors pulling a node in opposite directions. No nodes should float except the output in part c.
   a. An asynchronous reset (it may be easiest to modify some of the existing inverters rather than adding additional gates).
   b. An ENable input. When ENable is true, the FF loads Q from D on the rising edge of the clock. When ENable is false, the FF maintains state Q regardless of the clock.
   c. An output_enable input. When OE is false, the output is high-Z.

3. Write the Verilog that describes a FF with all 3 of the capabilities that you implemented above.

4. Consider the fluidic device shown in this video http://www.youtube.com/watch?v=GTnVwyWaVQw
   a. Can you design a divide-by-16 circuit with this type of device?
   b. How about divide-by-3?

5. You have a digital to analog converter that takes in a 10 bit unsigned integer B and converts it to a voltage Vout = 5V * B / 2^{10}. You have an analog comparator which takes two analog input voltages V+ and V-, between 0 and 5V, and generates a single bit digital output Cout = (V+ > V-). You have a sample-and-hold circuit that captures an analog value and holds it when given a rising edge on its Sample input. Design the digital controller that uses these elements to generate a 10 bit ADC value every 10 clock cycles after reset. This digital circuit is called a successive approximation register (SAR). Here’s an example of such a digital chip http://www.ti.com/lit/ds/snos311a/snos311a.pdf
   a. Write a testbench for your SAR. Input to your DUT will be a sequence of 10 comparator outputs. The testbench should check if the value in the SAR is correct after each clock cycle. For example, a comparator input sequence of all 0s indicates that the SAR state should look very much like the output of a shift register with a single “1” going through it.
   b. Draw the block diagram of your SAR module implementation (something very close to the Wikipedia figure is fine http://en.wikipedia.org/wiki/Successive_approximation_ADC)
   c. Draw the state transition diagram
   d. Write the Verilog that implements your module.
   e. Turn in your diagrams, Verilog, and simulations.