University of California at Berkeley
Department of Electrical Engineering and Computer Science

EECS150/EE141/EE241A, Fall 2014

Homework 3: Finite State Machines and More Verilog

Due: Friday Sep 26, 5pm

Please include your name, SID and specify either CS150, EE141 or EE241A at the top of your homework handin. Homeworks must be submitted electronically.

1. You are to design (using behavioral Verilog) a serial one hot code to binary converter which converts 4-bit one hot code to binary numbers. Instead of having all 4 bits of the one hot code at the same time, you would see one bit each clock cycle. Your circuit should have two outputs, one for the binary number, and another one indicating if the current binary number is valid. The valid signal should only be high after all 4 bits are seen and they constitute a legal one hot code. After the circuit decodes one 4-bit code or detect an invalid code, it would be reset before the next code is presented.

2. Describe the differences between how the synthesis tools will interpret a case statement vs. a series of if-else statements. How does this affect the circuit that will be inferred?

3. Design a FSM which controls a vending machine. The vending machine would dispense a can of coke if it receives 25 cents, and it does not give back change. Besides `clk` and `reset`, there are three inputs to the vending machine: `quarter`, `dime`, and `nickel`, which would be turned on high for 1 clock cycle if the corresponding coin is inserted. The FSM would produce 1 output, which is high for a cycle when 25 cents or more are received, after which it returns to the initial state. You can assume only one of the three inputs would be high at a particular clock cycle. You are not allowed to use a counter for your design.

   (a) Draw the state transition diagram and write behavioral Verilog for a Mealy machine implementation.

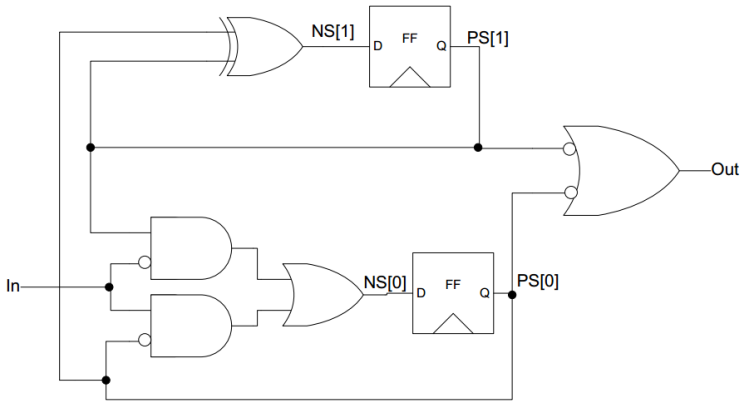
   (b) Draw the state transition diagram and write behavioral Verilog for a Moore machine implementation.

4. Given below is some random guy's attempt at the Verilog describing a D flip-flop with an asynchronous reset and an enable:

```
module dff{input clk, input rst, input en, input d, output q};
    reg q_reg;
    assign q = q_reg;
    always @(*) begin
        if(rst)
            q_reg = 1'b0;
        else begin
            q_reg = d;
        end
    end
endmodule
```

   What is wrong with his code? Fix it so that the D flip-flop functions as expected.

5. Consider the following finite state machine (FSM) circuit:



   (a) Write a Verilog description of the circuit using continuous assignment for the `NS` and `OUT` signals.

   (b) Draw a state transition diagram describing the behavior of the circuit. Within each state bubble indicate the bit encoding for that state. Remember to label the arcs with input values and state with output values.

6. Build an edge detector circuit with a parametrizable delay in Verilog. The module should have a single bit input `in`, single bit output `out`, and a parameter `delay`. After a rising edge is detected, your circuit should wait for the number of cycles specified by `delay` before setting the output high for a single cycle. Implement this in Verilog using a generate statement.

7. You are given three registers R1, R2 and R3, each of which has an enable signal. A register will keep its old value at positive edge of the clock if this signal is not asserted.

(a) Draw the datapath for a circuit that can swap the values in any two of the registers. Minimize the number of cycles needed to perform the swap. You may use primitive gates and MUX's of any size, as well as any additional registers you may need. Clearly label the control signals you would use to regulate the swaps.

(b) Augment your datapath so that it has three input values (V1, V2, and V3), and a `load` input signal. When `load` is asserted on a rising clock edge, the three values will be loaded into the registers. The circuit should still maintain its swap functionality.

(c) Express your datapath as a Verilog module with your control signals, `load` signal, and load values as inputs, and the three register values as outputs. Your module should also have clock and reset inputs.

(d) Briefly describe the sequence of actions needed to swap the values in R1 and R2, referring to control signals you have in part a.

(e) Design a control FSM that will sort the three values, with the greatest value in register R1. Draw the state transition diagram. Make sure to indicate the behavior of your control signals. You may use a comparator block and any other components you need in your logic.

(f) Express your control FSM as a Verilog module with the register values as inputs and the control signals as outputs. The module should also have a `done` output, which will be asserted once the three values are sorted. Your module should also have clock and reset inputs.

8. Design a circuit that has an input x, and an output y. The output is 1 when the input sequence "1010" is detected.

   (a) Draw the state transition diagram for your FSM and show how you will encode each of your states.

   (b) Show the truth table showing x and the current state as inputs, and next state as output.

   (c) Draw the circuit of the FSM, using D-flip-flops and two-input gates.

   (d) Assuming clock-to-q delay of 1ns, 1ns delay for each gate, and 0.8ns setup time for all D flip flops, how fast can you clock the FSM? Highlight the critical path.

9. **For EE241A only:** Imagine you have a random stream of bits coming in, one bit per clock cycle, to a 64 bit shift register that holds an integer value. The bits come in and get shifted in to the LSB. Every 64 cycles, a `reset` signal is asserted and the shift register is reset to 0. Design the FSM for a circuit that will output 1 when the value held in the register is divisible by 5. *Hint: You should be able to do it with 5 states.*