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

EECS150, Spring 2013

Homework Assignment 3: Behavioral Verilog, Synthesis and Simulation and High-level Design
Due Feburary 19^{th}, 2pm

1. In homework 2, you designed a 3-bit $\overline{Up}/Down$ counter, now you are to implement it as an FSM. You can assume the counter output wraps around when it overflows/underflows.
   (a) Draw the state transition diagram
   (b) Implement your FSM using Verilog.
2. 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.
3. You are given three registers R_1, R_2 and R_3, each of which has an enable signal. A register would keep its old value at positive edge of the clock if the enable signal is not high. You will design a circuit capable of swapping the values in any two of the registers.
   (a) How would you connect the registers such that you can swap the values between any two of them? Make sure you minimize the number of clock cycles needed for the swap. Feel free to use any additional components. Draw the circuit diagram and label clearly the signals you would use to control the actions of each component.
   (b) Briefly describe the sequence of actions needed to swap the values in R_1 and R_2, referring to control signals you have in part 3a.
4. You have four registers each holding a number, you are to design a circuit to find the greatest number and put it into all four registers. You can use comparators, registers and other components in your implementation. Input s would start the operation and when the task is completed, a $done$ signal should be asserted.
   (a) For the first version of the circuit your are allowed to employ only one comparator.
      i. Arrange the four registers, the comparator, and any other necessary components into a datapath that can accomplish the described task. Draw a circuit diagram and label the signals you would use to control the actions of the datapath.
      ii. Draw the state transition diagram for a FSM to control your circuit in part 4(a)i to accomplish the desired task. State clearly the value of each control signal in every state.

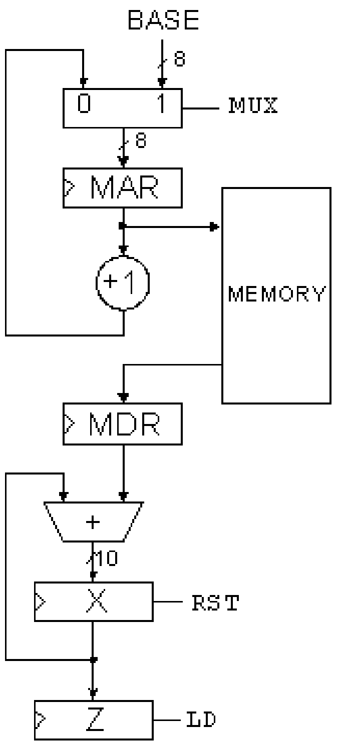
(b) In this part, devise a circuit that minimizes the number of clock cycles necessary. You are allowed to use more comparators and other components as necessary. (Hint: It is possible in one clock cycle.) Draw your resulting circuit.

5. Create a testbench for the full adder circuit you created during lab 0, make sure you cover all possible input combinations. If you are given a 16-bit adder, how many test cases would be needed for exhaustive testing?

6. Create a testbench for the FSM in question 1, make sure all state transitions are covered.

   it works.

7. A simple processor is designed to add the contents of blocks of 4 bytes in consecutive memory locations. The datapath is shown below



The processor has one data input (8-bit wide) named **BASE**, an input control signal named **ENABLE**, and 3 internal control signals - **MUX**,**LD**, and **RST**. The datapath contains three data registers - **MAR**, **MDR**, and **X**. After the processor performs its operation, the **Z** register is left with the sum of memory locations **BASE**, **BASE + 1**, **BASE + 2**, and **BASE + 3**. We assume that a controller (not shown) will take as input the **ENABLE** signal and generate **MUX**, **RST**, and **LD**. To begin the addition operation, an external circuit asserts **ENABLE** for 1 clock cycle then lowers it for a minimum of 12 cycles.

Write the register transfer language level description for the sequence of transfers that must occur after the **ENABLE** signal is asserted. Try to minimize the total number of cycles.