CASE WESTERN RESERVE UNIVERSITY
Case School of Engineering
Department of Electrical, Computer and Systems Engineering
ECSE 281. Logic Design and Computer Organization (4)

Assignment #10 **Due: April 2, 2022**

PLEASE SUBMIT YOUR WORK AS A SINGLE pdf DOCUMENT ON CANVAS.

Part 1: Problems (30 pts)

Please show your work.

1) (6 pts) Draw the state diagram for the following problem: There is one input A and two outputs X and Y. X becomes one if A has been 1 for at least three cycles altogether (not necessarily consecutively). Y becomes 1 if A has been 1 for at least two consecutive cycles.

2) (6 pts) Given the following excitation and output equations, write the transition and output tables. X is the input, Z is the output.

   $D1 = Q1' + Q2$

   $D2 = Q2' \cdot X$

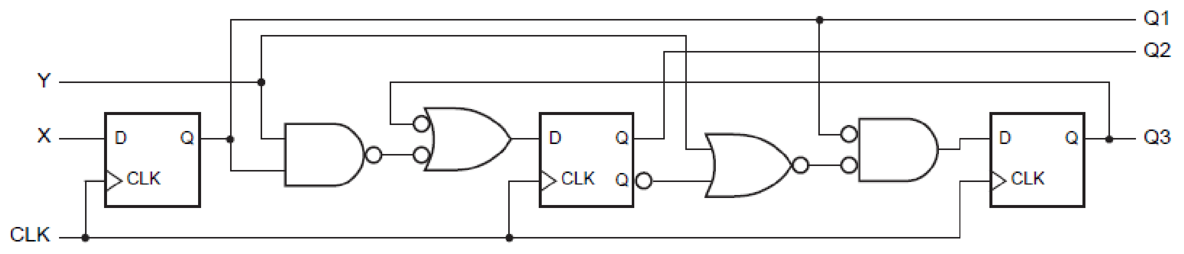
   $Z = Q1 + Q2'$

3) (6 pts) Draw the sequential circuit that has the state transition equations given below. Use D flip-flops and combinational logic gates. In the equations X is the input. The outputs of the circuit are the states Q1 and Q0.

   $Q1^* = X' \cdot Q1 + Q1 \cdot Q0' + X \cdot Q1 \cdot Q0$
   $Q0^* = X' \cdot Q0 + X \cdot Q0'$

   Is this a Mealy or a Moore machine? Explain briefly.

4) (6 pts) Write the excitation equations for the following state machine:



5) (12 pts) Design a 3-bit binary counter with counting sequence 0, 1, 2, 3, 4, 5, 0, 1, …
   The counter will increment at each clock tick.
   Use binary state assignments. Use D flip-flops, and combinational logic gates.
   Follow the steps below:
   - **a.** Draw a state diagram.
   - **b.** Write a state transition table.
   - **c.** Write state transition, and excitation equations.
   - **d.** Draw the circuit.

Part 2: Laboratory (20 pts)

N-bit Up/Down Counter

So far, we've designed purely combinational circuits in these lab assignments. Starting this week, we will begin integrating some sequential elements into our designs. This week, we will explore a simple sequential circuit known as a counter which simply increments (or decrements) its stored value by one every clock cycle that it is enabled. Our design will be based on the 74x169 up/down, however we will modify it to support an arbitrarily sized number of bits in the counter using parameters.

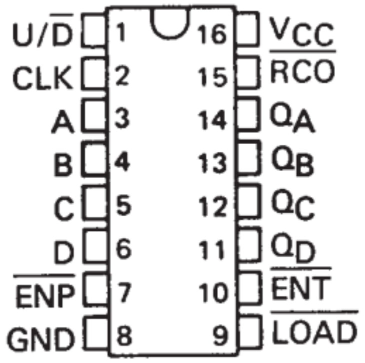


Figure 1: Pin-out for a 74x169

The 74x169 our design will be based on is shown above and has a couple features beyond just incrementing a counter with each clock cycle.

1. Pin 1 ($U/\overline{D}$): When this pin is high, the counter will count up by one on each positive edge of CLK. Otherwise, it will count down by one with edge positive edge of CLK.
2. Pin 9 ($\overline{LOAD}$): When this pin is low, the values of A-D will be latched into the counter flip-flops instead of counting normally.
3. Pin 15 ($\overline{RCO}$): Goes low when the next increment of the counter would cause an overflow condition (Q goes counts "up" from 15 to 0 or "down" from 0 to 15).

Step 1: Designing the Counter Module

Create a new Modelsim project and call it abc123_eecs281_lab7 where abc123 is your Case ID. In it, create a new SystemVerilog file and name it "up_down_counter.sv". Write a new module by the same name in this file which will implement the entire counter design. The module should have the following parameters and ports:

- one parameter ("N") which defines the number of bits the counter can hold
- one 1-bit input port ("clk") for the clock signal
- one 1-bit input port ("en_b") for the active low enable signal
- one 1-bit input port ("load_b") for the active low load signal
- one 1-bit input port ("up") to indicate the counting direction
- one N-bit input port ("load_in") holding the data to be loaded in when load_b is asserted
- one N-bit output port ("q") holding the current value of the counter
- one 1-bit output port ("rco_b") to indicate if the next increment will cause an overflow

Inside the module implement the counting behavior inside of an always_ff block. Your module should have the following behavior:

- when en_b is not asserted, the values of Q should not change
- at every positive edge of clk where en_b is asserted and load_b isn't, q should count up by 1 if up is high or down by 1 if up is low
- at every positive edge of clk where en_b and load_b are asserted, q should be set to the current value of load_in

In addition to the always_ff block which implement the actual counting function, you should have a separate always_comb block which sets the value of the rco_b output (alternately you can implement the rco_b functionality as a series of assign statements). Your rco_b logic should drive the output low whenever the counter would roll over from its maximum value (2^N-1) to 0 or vice-versa. This should work for any value of the N parameter used. As an important note, rco_b should be active on the cycle before the roll-over occurs (e.g. for a 4-bit counter, when Q=15 and up=1), not the cycle after the roll-over occurred

Step 2: Testbench Design

Create a second file in the project called testbench.sv, and in it implement a testbench for the counter module. Instantiate two copies of up_down_counter, one with N=4, and one with N=5. For both instances, it's important to note that there is no reset signal for this counter, so the values of q will initially be X (undefined) in the simulation. Before you can actually start counting, you will need to do a load to get the counter into a known state. As with the previous lab, you may generate the test inputs however you like, but you should follow this general pattern:

1. load 0 into both counters
2. count up until the counters roll over back to 0 (ensure that rco_b behaves as expected)
3. load the maximum value into both counters
4. count down until the counters roll over back to their maximum value
5. try loading a couple values in the middle of the counting ranges to each counter
6. make sure that when en_b is high, the values of q do not change (even on a load).

Step 3: Deliverables

To turn in your lab code, submit all your code as part of your homework pdf file. You can copy and paste the code into a word processor and then add it to your homework pdf. You also need to submit the waveform. You can take a screenshot and add it to your homework file.