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

Assignment #11 **Due: April 16, 2022**

PLEASE SUBMIT YOUR WORK AS A SINGLE pdf DOCUMENT ON CANVAS.
Part 1: Problems (30 pts)

Please show your work.

1) (6 pts) Design a 4-bit ripple down counter using only four D flip-flops.
2) (6 pts) For the following inputs that are applied to a 74x163 counter, draw a logic diagram and write the output (QD QC QB QA) for the next 15 clock ticks. Assume that the current count is 0000.

   Inputs:

   ENP and ENT and D are always HIGH

   A, B and C are always LOW

   LD_L = (QA·QC)'

   CLR_L = (QB·QD)'

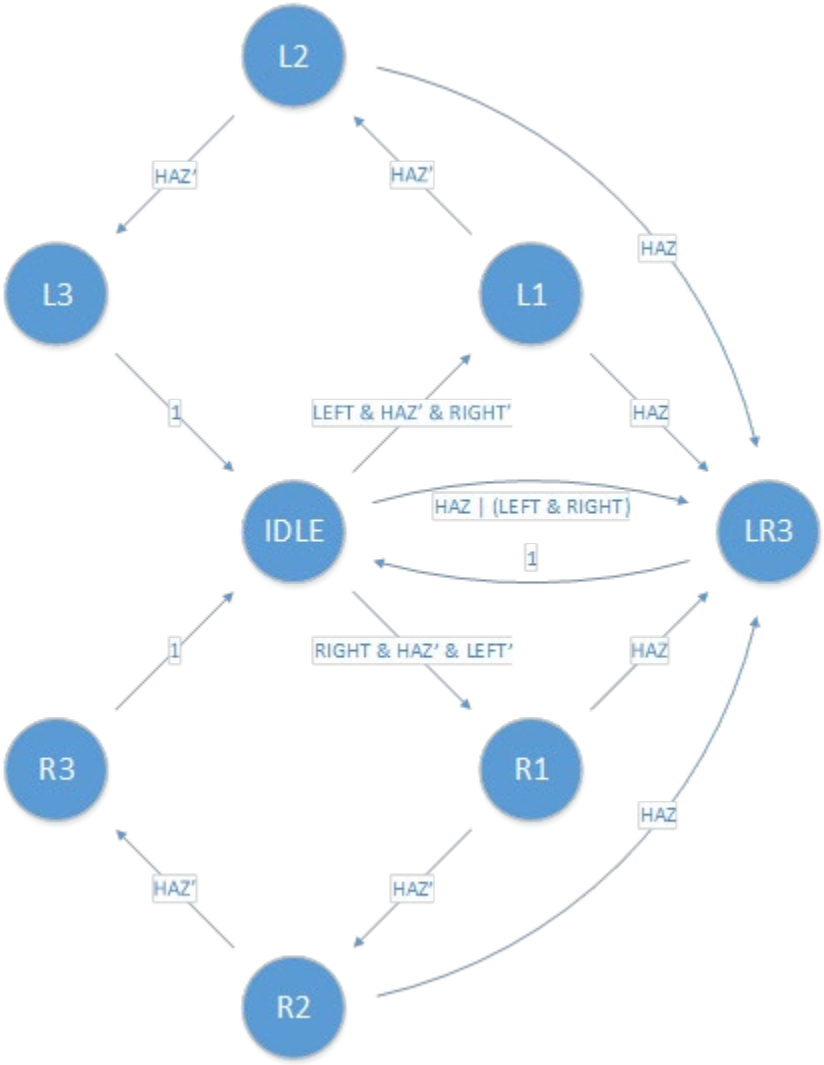
   CLK input is connected to a free-running clock signal.

3) (6 pts) Using a 74x163 and external gate(s), design a modulo-10 counter circuit with the counting sequence 3,4,5,6,…, 12, 3,4,5,6, …
4) (6pts) Using two 74x163 counters, design a counter with counting sequence 0, 128, 129,…, 254, 255, 0, 128, 129, … , 254, 255. Logic 0 and 1 are available.
5) (6pts) Using one 74x169 and three inverters, design a counter with the counting sequence 4, 3, 2, 1, 0, 11, 12, 13, 14, 15, 4, 3 ...

Part 2: Laboratory (20 pts)

Finite State Machines

This week, we will implement the finite state machine (FSM) for the Thunderbird tail-lights. A visual representation of the state diagram is given below. However, instead of manually synthesizing the excitation and output equations, like presented in the textbook's analysis, we will create a behavioral model for the FSM in Verilog using case statements and enumerated types as discussed in the recitation.



Step 1: Designing the FSM

Create a new Modelsim project and call it abc123_eecs281_lab8 where abc123 is your Case ID. In it, create a new SystemVerilog file and name it "tbird_fsm.sv". First, define a new enumerated type with the state assignments shown in Table 9-7 of the textbook (page 476) (also can be found in lecture notes). You should use the same state names from the table as your enumerated labels. Then, in the same file, create a new module named tbird_fsm which will implement the entire FSM. The module should have the following ports (no parameters required this time):

- one 1-bit input port ("clk") for the clock signal
- one 1-bit input port ("rst_b") for the active low reset signal
- one 1-bit input port ("left") to request a left turn
- one 1-bit input port ("right") to request a right turn
- one 1-bit input port ("haz") to request the hazard lights
- one 3-bit output port ("l_lights") to indicate which of the left-side lights are on
- one 3-bit output port ("r_lights") to indicate which of the right-side lights are on

Inside the module, implement the computation of the next state using a behavioral case statement inside of an always_ff block like shown in the recitation. Also remember that, in practice, the flip-flops that define the state will start in an unknown state (i.e. X), so you will need to include code in the always_ff block to initially reset the flip-flops into the IDLE state based on the rst_b input.

Additionally, you should include a separate always_comb block to set the values of the output signals (l_lights and r_lights) based on the current state of the FSM according to the output table from page 475 of the text (Figure 9-24) (you can also find the output table in lecture notes). You will probably also find this most convenient to implement the output logic using a case statement.

Step 2: Testbench Design

Create a second file in the project called testbench.sv, and in it implement a testbench for the Thunderbird tail light FSM. You should test every possible transition in the FSM to make sure that it works properly. One possible input sequence to achieve this would be:

1. reset the FSM
2. wait 1 cycle to ensure the FSM remains in IDLE
3. apply the right input for 4 clocks to test a complete cycle of the right turn
4. apply the left input for 4 clock to test a complete cycle of the left turn
5. apply haz for 2 clocks to test a complete cycle of the hazard lights
6. apply left for 1 clock and then haz for 1 clock
7. apply left for 2 clocks and then haz for 1 clock
8. apply right for 1 clock and then haz for 1 clock
9. apply right for 2 clocks and then haz for 1 clock

During each phase of the test, ensure that the output signals behave as defined by the output table from the book using the waveform viewer. You should also observe the FSM state throughout the test to ensure it follows all the appropriate transitions.

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.