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

Assignment #8 **Due: March 18, 2022**

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

1. (6 pts) Implement the following functions using only 74x138 binary decoders and NAND gates.

$$F = \prod_{A,B,C}(3,4,5,6,7)$$

$$F = \sum_{W,X,Y,Z}(2,3,4,5,8,10,12,14)$$

2. (6 pts) Design a 10-to-4 encoder with the inputs 1-out-of-10 code and outputs coded normally for 0 – 7 (binary 0000 - 0111) and 8 is coded as E (1110) and 9 is coded as F (1111). Show the internal circuit.

3. (6 pts) Implement the following function using only a single 4×1 multiplexer and inverters.

a	b	c	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

4. (6 pts) For the logic expressions given below, find all of the static hazards and design a hazard-free circuit that realizes the same logic function. Write the functions that are hazard free, you do not need to draw the circuit. (Hint: Use Karnaugh maps to find the timing hazards.)

   $F = W \cdot X + W' \cdot Y'$

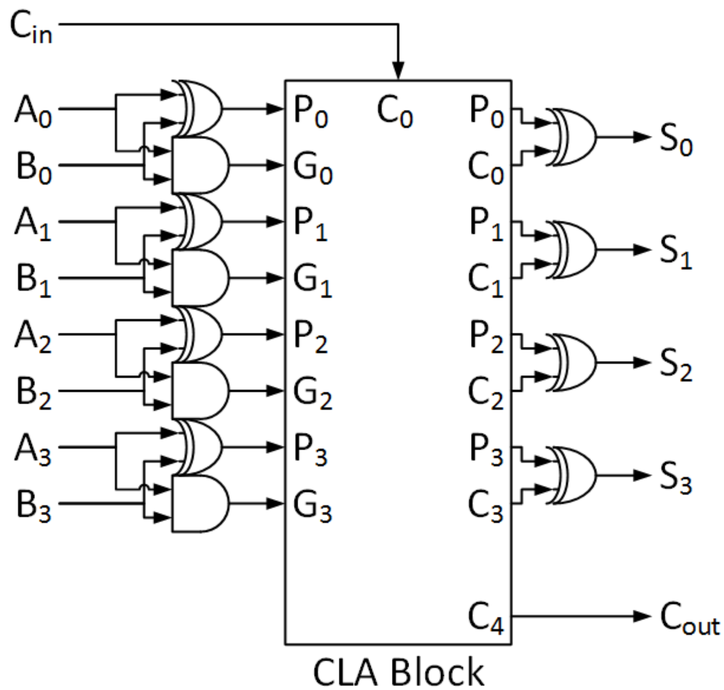
   $F = W \cdot Y + W' \cdot Z' + X \cdot Y' \cdot Z$

5. (6 pts) Design an 8x1 multiplexer (8 data sources / 1 bit data from each source) using 2x1 multiplexers only. You can use as many 2x1 multiplexers as needed. Clearly label all the inputs and outputs.

Part 2: Laboratory (20 pts)

N-bit Carry-Lookahead Adder

Over the past 2 weeks, we've looked at various ways to implement multi-bit adders using a ripple-carry topology. This week, we will look at how to design the Verilog code for a Carry-Lookahead adder which is a faster, but larger, way to generate the carry information in an adder. We will design the adder as a single module that can be parametrized and create a testbench for two adder sizes.



Step 1: Designing the CLA Adder Module

Create a new Modelsim project and call it abc123_eecs281_lab5 where abc123 is your Case ID. In it, create a new SystemVerilog file and name it "cla_adder.sv". Write a new module by the same name in this file which implements the entire CLA adder design as shown above. The module should have the following parameters and ports:

- one parameter ("N") which defines the width of the adder
- two N-bit input ports ("a" and "b") which hold the data to be added
- one 1-bit input port ("c_in") for the initial carry in
- one N-bit output port ("s") which holds the sum of the two inputs
- one 1-bit output port ("c_out") which holds the carry out from the most significant bit

The CLA block should be implemented as a single generate loop which expresses the carries iteratively as follows:

$$C_{i+1} = G_i + P_i C_i$$

You may either assign the initial carry (C_0) into the loop as a separate assign statement or through a conditional generate loop like was shown in recitation. The propagate, generate, and sum can all be generated through standard assign statements using bitwise operators.

Step 2: Testbench Design

Create a second file in the project called testbench.sv, and in it implement a testbench for the CLA adder module. Instantiate two copies of cla_adder, one with N=2, and one with N=8. Exhaustively test the N=2 adder. Test the N=8 adder by incrementing the A input by 3 every cycle, and the B input by 5 every cycle. You may fix the c_in to the 8-bit adder at 0. A should count in the following pattern (0, 3, 6, 9, 12, …) and B should count (0, 5, 10, 15, 20, …). You may generate the values of a, b, and c_in in any manner you wish. Terminate the test program once the exhaustive test is complete.

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 console output. You can take a screenshot and add it to your homework file.