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

Assignment #3 **Due: September 26, 2024**

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

Please attempt to solve all the problems. We will randomly select a question from the Problems part and grade it. The solutions of the homework will be posted once the submissions are completed in order for you to check your work for the ungraded questions. The lab part will be graded completely.

Problems (10 pts): Please show your work in detail.

1) Perform the following operations by first representing the decimal numbers in two's complement number system. (Hint: the subtraction will become addition). Use 6 bits to represent the numbers.

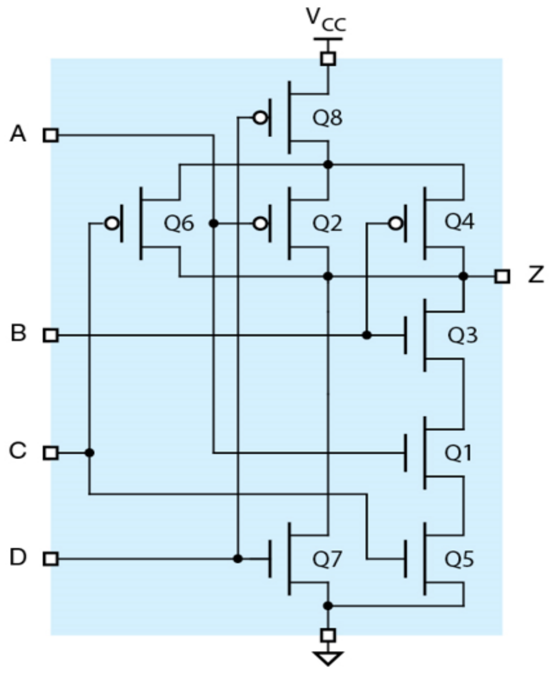
   14 – 16 = ?

   15 – 10 = ?

   22 – 4 = ?

   9 – 21 = ?

2) For the circuit given below, complete the table below for Z.



A	B	C	D	Z
L	L	L	L	
L	L	L	H	
L	H	L	H	
H	L	L	L	
H	H	L	H	
H	H	H	L	

3) Simplify the following expressions using the switching algebra theorems. Indicate all the theorems you use at each step.

   (X'·Y'+X) · ((X+Y) ' + X'·Y) simplifies to X'·Y'

   A'·B'·(D'+C'·D)+B'·(A+A'·C·D)+B'·C' simplifies to B'

4) Write the truth table for the following function and draw the circuit using AND, OR and NOT gates:

   F = a·b' + a'·c + a'·b·c'

5) Write the truth table for the full adder and find the canonical sum for the carry-out output of the full adder.

Laboratory (20 pts): Intro to Verilog

In this assignment, we will learn the basics of a hardware description language (HDL) called SystemVerilog. We will use SystemVerilog to describe a simple logic circuit shown in which has three logic gate (AND, OR, and XOR), two inputs ("a" and "b") and three outputs ("y1", "y2", and "y3"). We will implement this circuit using the three different Verilog coding styles (structural, dataflow, and behavioral) and then create a behavioral Testbench module to test the functionality of the three modules.

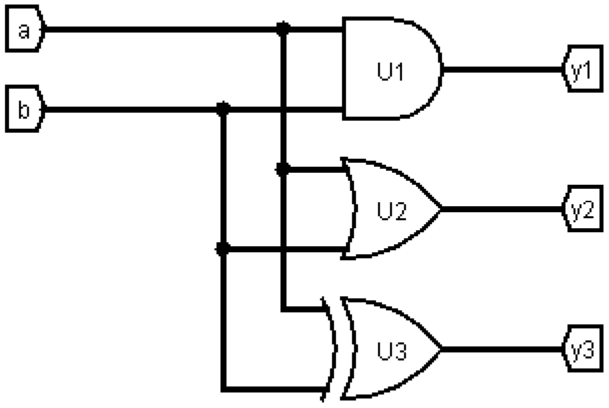


Figure 1: Logic circuit to be implemented in this assignment

You can refer to the recitation notes posted to Canvas under Pages / Verilog Notes on Verilog for reference.

Step 1: Create a new Modelsim project

Create a new project as described in tutorial and name it "abc123_eecs281_lab2", where abc123 is your Case ID.

Step 2: Create a structural version of the circuit

Create a new SystemVerilog file as described in the tutorial and name it "module_s.sv" and in it create a module that implements a structural description of the circuit in Figure 1. First, you will need to create a module header like described in the recitation. Make all your ports the logic type and name and set their directions as shown in Figure 1. You may order the ports however you like, but be sure to keep track of the ordering if you plan to do port connections by position.

The body of this module will contain one gate instantiation per gate shown in Figure 1 like in the example from recitation.

Step 3: Create a dataflow version of the circuit

Create another new SystemVerilog file and call it "module_d.sv" and in it create a module that implements a dataflow description of the circuit in Figure 1. The module interface will be the same as what you created in Step 2, just change the name of the module from "module_s" to "module_d". The body should be modified to use continuous assignment statements to model the circuit instead of gate instances though.

Step 4: Create a behavioral version of the circuit

Create a third SystemVerilog file and call it “module_b.sv” and in it create a module that implements a behavioral description of the circuit in Figure 1. Again, the only part of the module header that needs to change is the module name. Change the body of the module to use procedural assignments inside of an always_comb block.

Step 5: Create a Testbench for all modules

Create one final new SystemVerilog file and name it “testbench.sv”. Your Testbench module should instantiate one copy of each of the modules created in steps 2-4. Also include code to generate all possible combinations of inputs ‘a’ and ‘b’ (00, 01, 10, and 11). You may do this however you like. Two good approaches will be to either use always blocks to create toggling waveforms with different periods, or to use initial blocks and specify the values of ‘a’ and ‘b’ at each step (examples of both shown in the recitation notes). Finally include an output monitor to print the response of each module as simulation progresses. This can easily be done with a $monitor statement. You should also use a $display statement to label each column of the monitor statement to make it easier to read.

Don’t forget to also set your simulation timescale. You may use 1ns/10ps for this lab and all future assignments unless specifically directed otherwise.

Step 6: Running the simulation

Once all the modules have been written, compile the code and run a simulation with the testbench as the top-level module. Confirm that the output to the console for all three modules matches the truth tables for the three logic gates. Also create a waveform view and confirm that it too shows the correct response.

Step 7: 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 and the waveform. You can take a screenshot and add it to your homework file. **Please make sure to include your name on the screenshots of both the console output and the waveform by entering your name on the transcript window. Screenshots without the names clearly displayed on the transcript window will not be accepted.**