CIS 240 Fall 2022 - Homework #2

Assignment Instructions:

You will submit this assignment by uploading it to Gradescope. You do not need to print out this document and put your answers on it, and may instead put your answers on their own dedicated pages, just be sure you properly indicate which question is being answered on all parts of the page.

You may write your answers and draw them by hand or type them and use an online program to draw the circuits. Whatever you do, please make sure your circuit is clear and readable. If you are drawing by hand, a straight-edge and/or graph paper may help with readability.

Many of the questions will ask you to create a "single CMOS transistor circuit". This means that you should not create multiple circuits and feed the output of one circuit into the input of another. This does **not** mean that you are restricted to only one CMOS transistor.

Please also note that this homework is graded manually and does not have “public test cases” like the other homeworks usually will have. It also means that **<u>we will not grant late extensions that exceed 72 hours</u>** aside from special circumstances.

Problem 1a (5 pts)

Design a single CMOS transistor circuit that has 2 inputs, A and B and produces a high output if and only if A is high (A = 1) and B is low (B = 0). In order to accomplish this, you can assume that you also have access to the inverses of A and B and that you can use those as inputs to your circuit. Please remember that to get full points your circuit must be a proper CMOS circuit, label the pull down and pull up networks and make sure that these two circuits are complementary to each other.

Problem 1b (5 points)

An interesting thing about digital circuits is that there are often multiple ways to implement a given function. Please provide a different CMOS circuit that implements the same function as the one you provided for the previous question. Once again your circuit must be a proper CMOS circuit with complementary pull up and pull down networks. You can assume access to all of the same input signals.

Problem 2 (10 pts)
Design a single CMOS transistor circuit that has 3 inputs, A, B and C, that represents a 3-bit unsigned integer where A is the MSB, B is the middle bit, and C is the LSB. For example, if the input was equivalent to the integer value 3 (0b011), A would be 0, B would be 1 and C would be 1. The circuit should produce a high output if and only if the unsigned integer value is a multiple of 3. 0, 3 and 6 are the possible multiples of 3 that can occur with a 3-bit unsigned integer.

In order to accomplish this, you can assume that you also have access to the inverses of A, B and C and that you can use those as inputs to your circuit. Please remember that to get full points your circuit must be a proper CMOS circuit, label the pull down and pull up networks and make sure that these two circuits are complementary to each other.

Problem 3
Consider the following boolean function that takes 3 inputs (A, B, C): Output = (A OR B) AND C.

Problem 3a (5 pts)
Design a single CMOS transistor circuit that has 3 inputs, A, B and C that implements the boolean function: Output = (A OR B) AND C.

In order to accomplish this, you can assume that you also have access to the inverses of A, B and C and that you can use those as inputs to your circuit. Please remember that to get full points your circuit must be a proper CMOS circuit, label the pull down and pull up networks and make sure that these two circuits are complementary to each other.

Problem 3b (5 pts)
Implement a gate-level circuit that has 3 inputs, A, B and C that implements the boolean function: Output = (A OR B) AND C. Be sure to properly label all inputs and outputs to the circuit and make sure you are drawing a readable & legal gate level circuit

Problem 3c (5 pts)
In class we briefly went over the pros and cons of working at different abstraction levels. These can include things like development time, readability, transistor delay, cost, etc. If you had to regularly design circuits as part of your job, which abstraction level (CMOS-level or Gate-level) would you want to work with? Please briefly explain your reasoning in 2-3 sentences.

(Note: the choice here is subjective, so we care about your explanation for your choice)

Problem 4 (5 pts)
The idea of Computational Sprinting was developed a few years ago here at Penn by a former Penn Professor, Milo Martin. Here is a link to a short article that describes the basic idea. You can find more information via a simple search if you are interested.

https://www.scientificamerican.com/article/computational-sprinting/

Explain in 2 or 3 sentences the basic idea of computational sprinting.

Problem 5 (15 pts)
NAND gates are sometimes termed universal gates since it turns out that you can implement any other logical function using only 2 input NAND gates. To show that this is true, produce circuits showing how you could implement the fundamental logical operations, NOT, AND and OR using only NAND gates. Since any logical function can be expressed in terms of these three this suffices to prove the result.

Which of the following gates are universal? If you think a gate is universal, show an implementation of the three fundamental logic operations using only that gate. If you think a gate is not universal, state which of the three fundamental logical operations you can implement, which of the three fundamental logic operations you cannot implement and brief (2-3 sentences) on why you cannot implement that fundamental logic operation.

- NOT gate
- AND gate
- OR gate
- NOR gate
- XOR gate

Problem 6 (5 pts)
Design a PLA circuit that takes a 4 bit input, I, representing a 4 bit unsigned number and outputs a logical 1 if and only if the input is a power of 2 (e.g., 1, 2, 4, 8). Please make sure to clearly label your diagram to indicate which signal lines represent the four inputs: I_3, I_2, I_1, and I_0 where I_3 corresponds to the MSB of the input and I_0 to the LSB.

(please refer to the course slides for a description of a PLA circuit, it's a gate level circuit with one layer of AND gates and a final OR gate that implements a sum of products function directly).