EE 109 Homework 5

Name: _____
Due: See class website

Score: _____

HW 5a (74 pts.) – Blackboard Submission

Binary Systems

1.) [BB] What are the corresponding decimal representations for the following binary strings? (2 pts. each)

Binary String	*8-bit unsigned* format	*8-bit 2's complement* format
10110110		
11011011		

2.) [**BB**] For each of the following decimal numbers find the corresponding 8-bit representation using the indicated systems. Note: Some numbers may NOT be representable w/ 8-bits. If this is the case, put **NA** for the answer. Also find the minimum bits needed to represent the number in the 2's complement system. (1 pt. ea.)

	2's Complement	Minimum bits needed using 2's complement
-128		
+31		
+59		
-16		

3.) [**BB**] Each C declaration of the variable x is initialized to a value in decimal. Show that value represented in hex using the appropriate size indicated by the variable type (e.g. char = 1-byte = 2 hex digits). Assume a 32-bit computer system where 'int' = 4-bytes. Use a calculator only if you have to. (2 pts. each)

a. int x = 256;
b. char x = -128;
c. unsigned char x = 250;
d. char x = -6;
e. int x = -1;
f. unsigned char x = 'a';
g. unsigned char x = 97;

4) **[BB]** (16 pts.) Perform the following addition and subtraction problems assuming 2's complement numbers. State whether overflow does or does not occur for each problem. Justify your answer for why overflow does or does not occur. (You can easily check your work by converting to decimal.)

a.) `1010 0110` + `1101 1011`

b.) `0010 0001` + `0111 1001`

c.) `1000 0010` − `1010 1111`

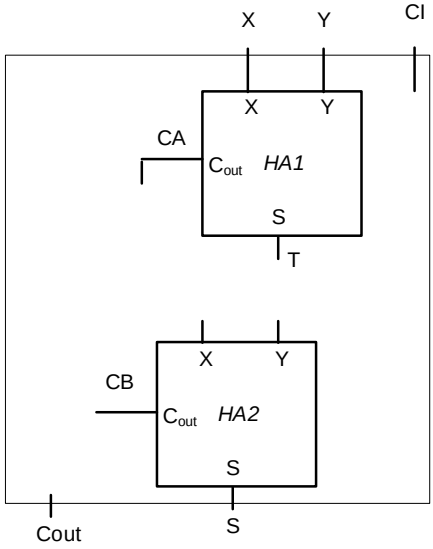
d.) `0101 1001` − `1010 0101`

5) **[BB]** (28 pts.) Design a circuit that takes in a 3-bit signed-magnitude number, A[2:0] ($A_2A_1A_0$), adds two to its decimal equivalent, and produces a 2's complement result, Z[n:0]. (i.e. Z = A + 2.) (Consider the signed magnitude combination representing –0 as an illegal input combination that should never occur.)

   a. Write out a truth table (consider how many outputs would be necessary for this problem)
   b. Use Karnaugh Maps to produce minimal SOP equations for each output bit

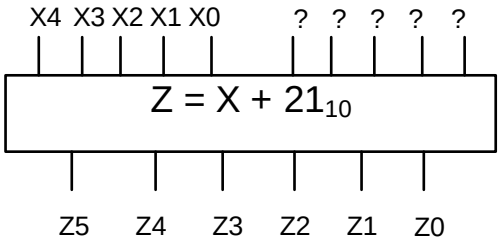
HW 5b (26 pts.) – BB Submission

6) [**BB**] (6 pts.) Build an equivalent full-adder using **two** half-adders as building blocks along with a **minimal** number of additional gate(s) if needed. On scratch paper, complete the schematic below. Then answer the questions on Blackboard.



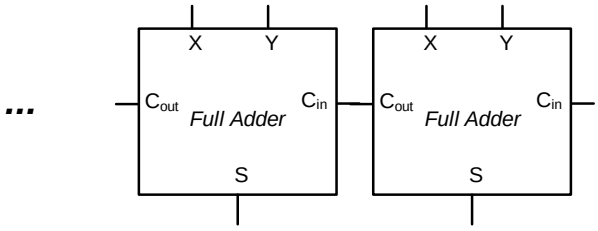
   a. What signals should you connect to HA2.X and HA2.Y?
   b. What additional gate should be used to produce the overall Cout signal (check all that will work correctly)?
   c. What inputs would you connect to that gate?

7) **[BB]** (10 pts.) Using **half**-adders and **full**-adder building blocks, design a circuit that takes in a 5-bit unsigned number, X ($X_4..X_0$) and produces an output equal to X + 21_{10}.



To do

To do this we would normally use a chain of full adders:

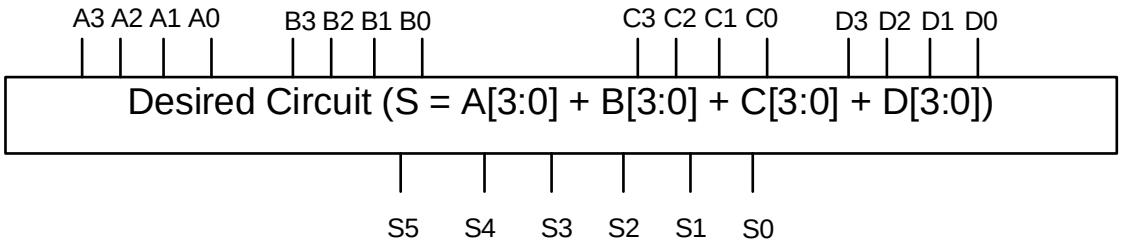


But our goal is to **minimize** the area required (i.e. use half-adders where possible.) On scratch paper, draw the schematic of your solution. Then answer the questions on Blackboard.

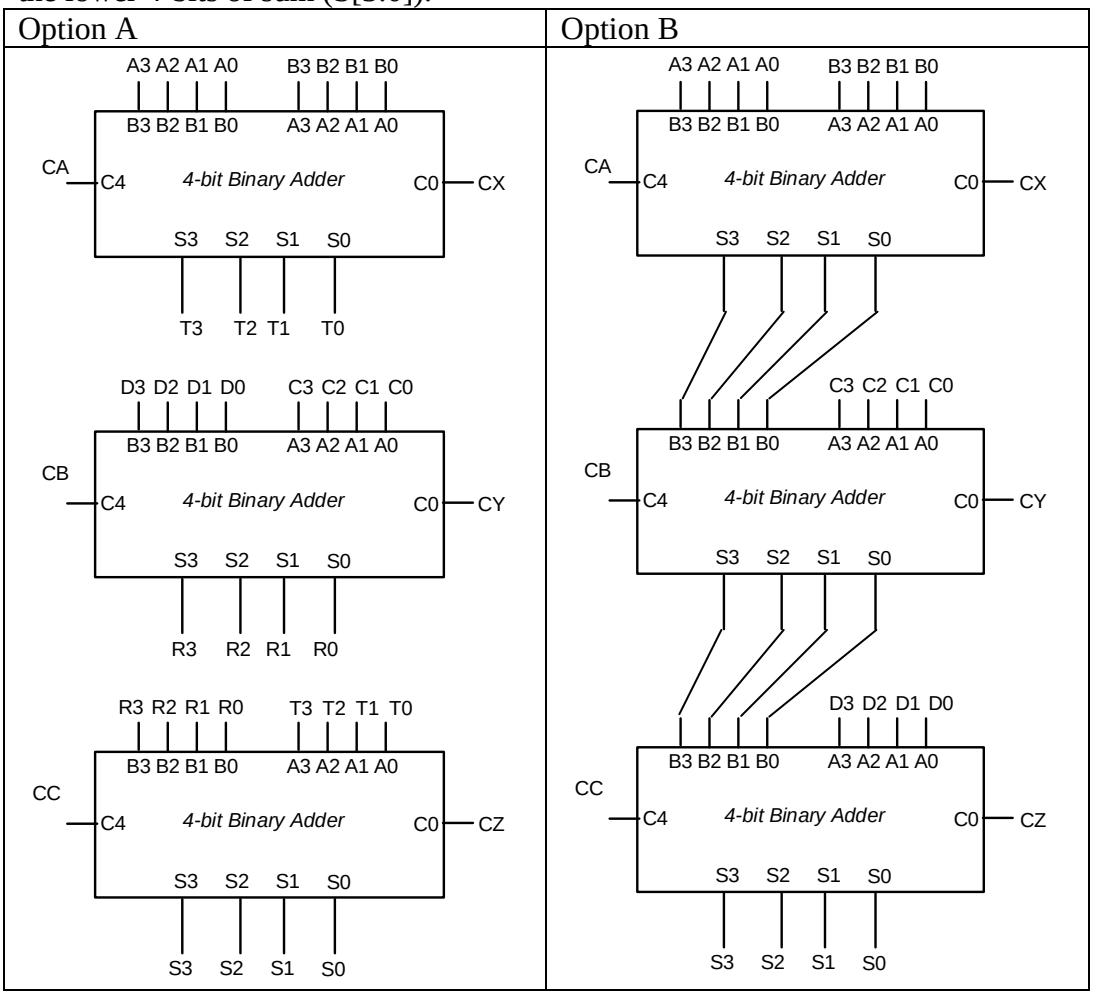
a. Check the boxes for which columns of addition can be performed using half-adders rather than full adders. Remember column 0 corresponds to the LS-bit.

8) **[BB]** (10 pts.) Design a circuit that takes in four 4-bit unsigned numbers, A ($A_3..A_0$), B ($B_3..B_0$), C ($C_3..C_0$), and D ($D_3..D_0$) and produces the 6-bit unsigned sum of those numbers. You should use **three** 4-bit adder blocks (74LS283's), and a **minimal** number of full adders or half adder build blocks. You should organize your adder circuits to perform as many additions in parallel (at the same time) as possible. Getting started: Write out the columns of addition and see where you can apply 4-bit adders. Use half and full adders for remaining addition operations.

Block Diagram of Desired Operation:



a. Which of the two initial set of connections will yield the smallest delay to produce the lower 4-bits of sum (S[3:0]).



b. Indicate what should be connected to the carry-in inputs of the 4-bit adders (CX, CY, CZ) for the option you selected.

c. Notice we have only provided 4-bits of sum, but that is not enough to produce the correct output for the addition of the four, 4-bit numbers. We must now add CA, CB, CC. Assuming we prefer use of full adders when necessary (rather than using multiple half-adders to mimic a full adder), how many full and half adders would you need to sum up CA, CB, CC?