Homework 6

Homework 6 is due on Wednesday, April 1, at the start of the lecture. Remember to include your *Discussions section* (e.g. ED1) and follow the complete Homework submission guidelines.

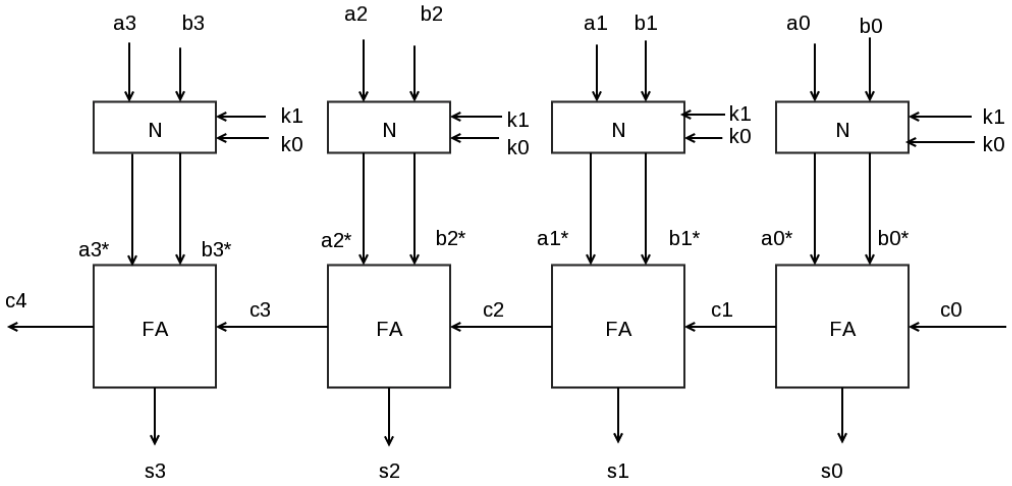
Please ask all questions about this assignment during the office hours, or post them on piazza.

Design of Combinational Logic Circuits

1. Design of an arithmetic logical unit

In this problem you will design an arithmetic logical unit: the arithmetic unit in part a, the logical unit in part b, and the complete ALU in part c.

a. Arithmetic unit (AU). Let A = $a_3a_2a_1a_0$ and B = $b_3b_2b_1b_0$ be 4-bit 2's complement numbers. Shown below is the structure of our arithmetic unit, consisting of four 1-bit Full Adders (FAs), four copies of a circuit N, and interconnections as shown.



As specified in the table below, our AU will compute an arithmetic function, depending on the values of control signals k_1 and k_0:

k_1	k_0	Function
0	0	A MINUS 1
0	1	B PLUS 1
1	0	A MINUS B
1	1	B MINUS A

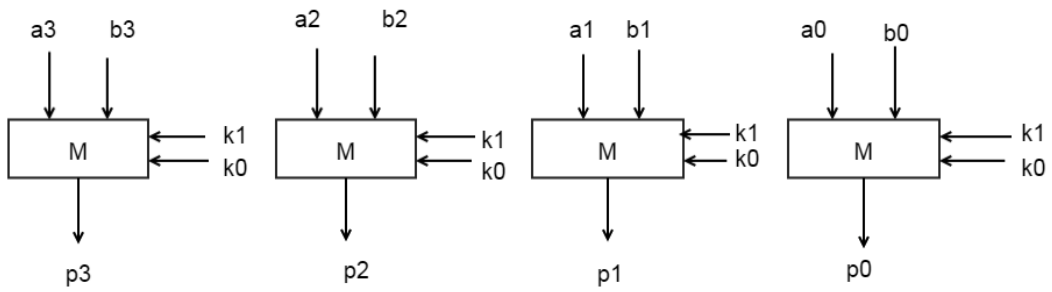
Provide the missing circuitry so that the AU performs as specified.

1. Express a_i* and b_i* in minimal SOP form. Use K-maps to show your work.
2. Draw the circuitry for N, using only NAND gates. Assume that complemented inputs are available.
3. Give a Boolean expression for c_0 as a function of k_1, k_0.

b. Logical unit (LU). In this problem you will design a 4-bit logical unit, which takes binary inputs A=a_3, ..., a_0, and B=b_3, ..., b_0 and control inputs k_1, k_0. The unit should perform the *bitwise* logical operations specified in the table below.

k_1	k_0	Function
0	0	A XOR B
0	1	A AND B
1	0	A OR B
1	1	A

Design circuit M so that the LU drawn below works as desired.



1. Express p_i in minimal POS form. Use a K-map to show your work.
2. Draw the circuitry for M, using only NOR gates. Assume that complemented inputs are available.

c. Arithmetic logical unit (ALU). In this part you will complete the design of an ALU that computes the arithmetic and logical operations specified in the table below. Since the ALU can perform any of 8 operations, there are three control lines k_2, k_1, k_0.

k_2	k_1	k_0	Function
0	0	0	A MINUS 1
0	0	1	B PLUS 1
0	1	0	A MINUS B
0	1	1	B MINUS A
1	0	0	A XOR B
1	0	1	A AND B
1	1	0	A OR B
1	1	1	A

The ALU has inputs A = $a_3a_2a_1a_0$, B = $b_3b_2b_1b_0$ and $k_2k_1k_0$. Denote the ALU outputs by Z=$z_3z_2z_1z_0$. In this part you will *use* (not redesign) the AU and LU circuits.

1. List all AU inputs and outputs. List all LU inputs and outputs.
2. Draw a "box" to represent the AU and another "box" to represent the LU. Each box must have all inputs and outputs labeled, and the box itself must be labeled. Complete the ALU circuit by interconnecting the AU and LU (be sure to label all wires). Use MUXes as needed and as few additional gates as possible.

Note: You may wish to use a dual (or 3-ary or quad) MUX, which is a chip that has 2 (or 3 or 4) MUXes - all of which share the same select inputs. E.g. a dual 4:1 MUX or a quad 2:1 MUX.

Notation: Use "slash N" notation to denote that a wire is a bundle of N wires (see e.g. the figure on p. 51 of Lumetta's notes).

2. Bit-slice design

Using bit-slice design, implement a circuit which checks if an *n*-bit binary number A has an odd number of 1s in its binary representation. The circuit should produce an output of **0** if the number of 1s is even, or **1** otherwise.

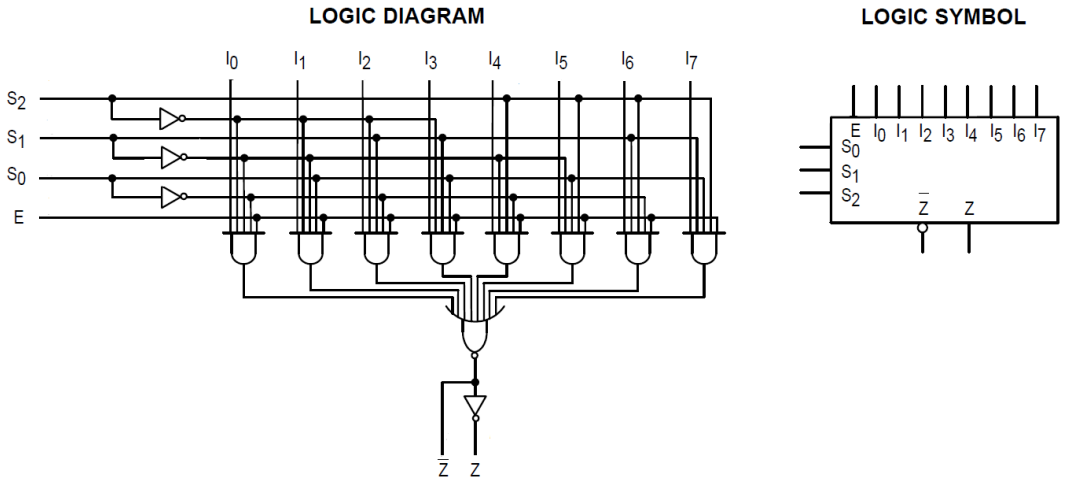
1. Draw the overall circuit consisting of *n* identical bit-slices. Label all inputs and outputs.
2. List the possible 'answers' (or information) that your bit slice may need to communicate to the next bit slice (and receive from the previous bit slice). Choose a representation for these 'answers'.
3. Write truth-table for output bit(s) from your bit-slice in terms of all inputs to your bit-slice.
4. Convert the truth table to Karnaugh map(s) and find minimal logical expression(s) for output(s).
5. Draw gate-level diagram for your bit-slice using NAND or NOR gates only. You may assume you have inverted inputs available.

3. Implementation of functions with MUXes

1. Using only one 8:1 MUX, implement the function f(a,b,c)=a + bc + a'b'c'.
2. Using only one 4:1 MUX and one inverter, implement the same function f(a,b,c) as in part 1.
3. Compare your designs from part 1 and part 2 in terms of number of gates. How many *n*-input gates are needed in part 1 and how many *n*-input gates are needed in part 2? (*Hint*: Lookup MUX implementation in the lecture notes.)

4. Building larger MUXes from smaller MUXes

Shown below is a schematic diagram of an 8:1 MUX.



1. Analyze the logic diagram and write Boolean expression for output Z as a function of its inputs S_2-S_0, I_7-I_0, and E.
2. Using two 8:1 MUXes and as few additional gates as possible, implement a 16:1 MUX. Clearly label all signals and ports.

5. Programming in C

Using the code from Homework 5 as a starting point, write a program to print the K--map for a 4 -variable function. Demonstrate its work using **f(w,x,y,z) = xy'+w'z** and **g(w,x,y,z) = w'xyz'+ w + x'** as examples. Submit the printed source code and the printout of the output of the program for the above function examples. Your program should only print the K-map using the above functions: there is no need for user input.