COE 0147 Spring 2014
Lab #10: K-Maps

Each of you should submit your own solution via email, according to the instructions at your TA's website. Each person must turn in their own copy of the lab. If you choose to work with a neighbor/partner, put your partner's name on your submitted copy of the lab.

There are two questions in this lab. You will have to zip your circuit diagram files (.circ files) into a .zip archive and submit it through the lab submission website as directed on the lab website. This circuit diagram portion is due before midnight on next Friday after your scheduled recitation. Submission timestamps will be checked. Late submissions will not be accepted. Follow instructions so that your program produces output in the correct format.

For the truth table and K-Map portions, you should turn in a hard copy of this assignment at the beginning of the following recitation meeting either in recitation or in your TA's mailbox. Staple multiple pages together and do not forget to put your name on. Do not present answers out of order.

Part 1: Developing a Combination Lock

In this section, you have to develop a combination lock. Your combination lock works with a 4-bit binary combination and can be unlocked only when the combination is 0101.

This circuit is built using two components: **LockDetect** and **PopCount**. Please use the circuit diagram from http://www.pitt.edu/~dbd12/teaching/files/lab10part1.circ to start with. In this circuit diagram, you can see the combination lock detector which uses two components. But the circuit is incomplete because the components LockDetect and PopCount are not yet drawn. You have to draw the circuits for **LockDetect** and **PopCount** components according to their description below. Save the circuit in **lab10part1.circ** and submit inside the zipped archive.

LockDetect will output a 1 only when the input is the correct unlock code (i.e., 1101). Otherwise LockDetect will output 0. Let's assume the input bits are: 3^{rd} bit is A, 2^{nd} bit is B, 1^{st} bit is C and 0^{th} bit is D. If the output of LockDetect is Y, then Y = A.B.C'.D which can be simply implemented using AND and NOT gates.

PopCount is a small helper circuit which will show how many 1's are correctly placed on the input. Do not consider the 0's which are correctly placed. For example, if the input is 1001, then the output from PopCount will be 2 because the 0^{th} bit and 3^{rd} bit are a 1's and are in the correct position according to the correct unlock code 1101. Similarly, if the input is 1111, then the output of PopCount will be 3. If the input is 1101 (the correct unlock code), the output of PopCount will be 2 because three 1's are correctly placed on the input.

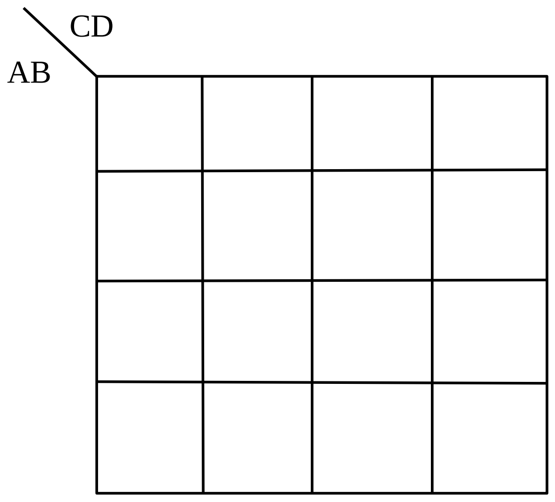
Assume the input bits are as follows: 3^{rd} bit is A, 2^{nd} bit is B, 1^{st} bit is C and 0^{th} bit is D. The output from PopCount ranges from 0 to 3 in decimal. So, you will need a 2-bit output from PopCount. Assume the output bits are as follows: 1^{st} bit is X1, 0^{th} bit is X0.

Print and fill out the truth table and the K-Maps for X1 and X0 on the next page. After you have filled out the K-Maps, circle the minterms and write your simplified equation as a sum of products. Be sure to show all of your work. Turn in the hard copy of the page as directed before.

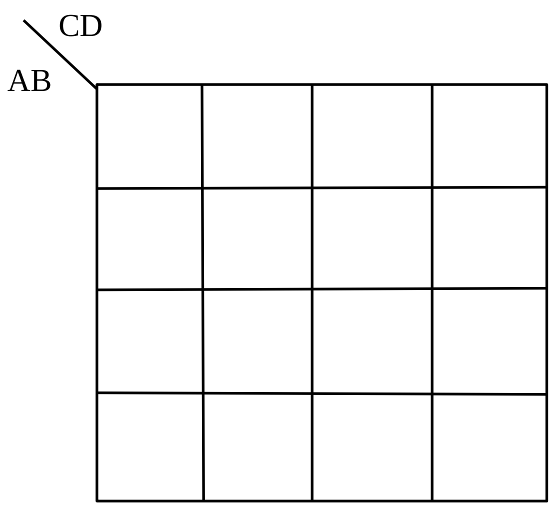
(CS 447/COE 147 Spring 2012 Lab 10 Part 1)

Name: Date:

Input					Output	
A	B	C	D		X1	X0
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1			



X1 =



X0 =

Part 2: Parity Generation

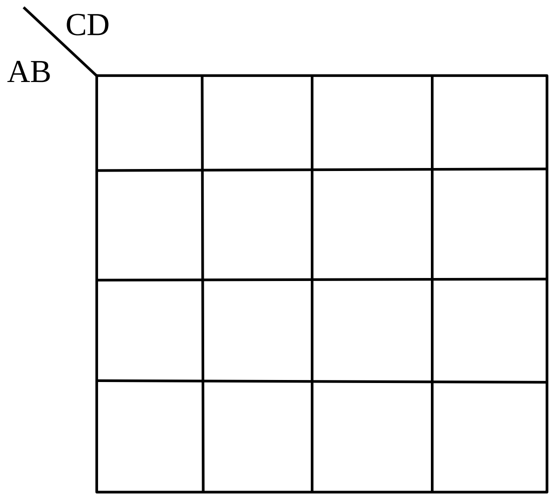
This circuit is built using one component: **ParityGen**. Please use the circuit diagram from http://www.pitt.edu/~dbd12/teaching/files/lab10part2.circ to start with. In this circuit diagram, you can see the parity generator module. But the circuit is incomplete because the components for ParityGen are not yet drawn. You have to draw the circuits for **ParityGen** according to their description below. Save the circuit in **lab10part2.circ** and submit inside the zipped archive.

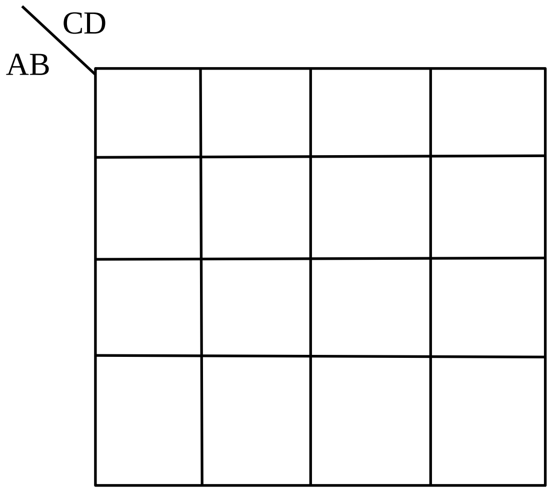
ParityGen is a circuit which generates the parity for the input. We count the number of 1's (T) in the input. Then we calculate the input parity (P) and add the parity into total number of 1's. Finally, we output the sum (S). Let's assume the input bits are: 3rd bit is A, 2nd bit is B, 1st bit is C and 0th bit is D, input parity is calculated by the equation P = A XOR B XOR C XOR D. For example, if the 4-bit input is 1011, the total number of 1's (T) is 3, input parity (P) is 1 by calculating 1 XOR 0 XOR 1 XOR 1. The total sum (S) is 100.
The sum output is at most 3 bits and can be represented by X2 X1 X0. X2 is 2nd bit, X1 is 1st bit and X0 is 0th bit.
Print and fill out the truth table and the K-Maps for X2 X1 and X0 on the next page. After you have filled out the K-Maps, circle the minterms and write your simplified equation as a sum of products. Be sure to show all of your work. Turn in the hard copy of the page as directed before.

(CS 447 Spring 2012 Lab 10 Part 2)

Name: Date:

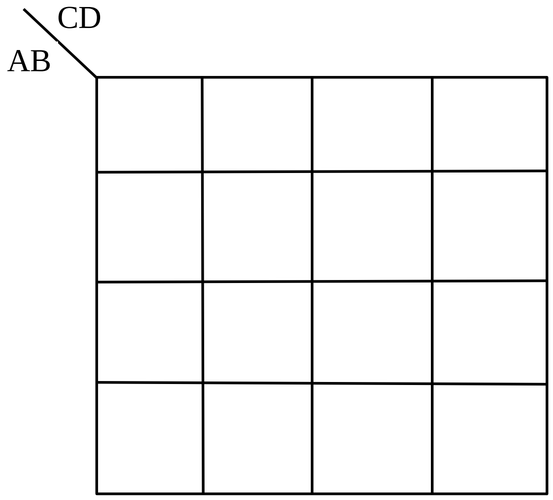
Input					Output		
A	B	C	D		X2	X1	X0





X2 =

X1 =



X0 =