CDA 3201 Computer Logic Design
Summer 2017
Homework 2

Total points: 100

Notes:

1. **All homework should be done and submitted individually**
2. **Show all steps for each question to get full points**
3. **Submit electronically in canvas as a single pdf file**
4. **Follow instructions for each questions**
5. **A' is the complement of A**

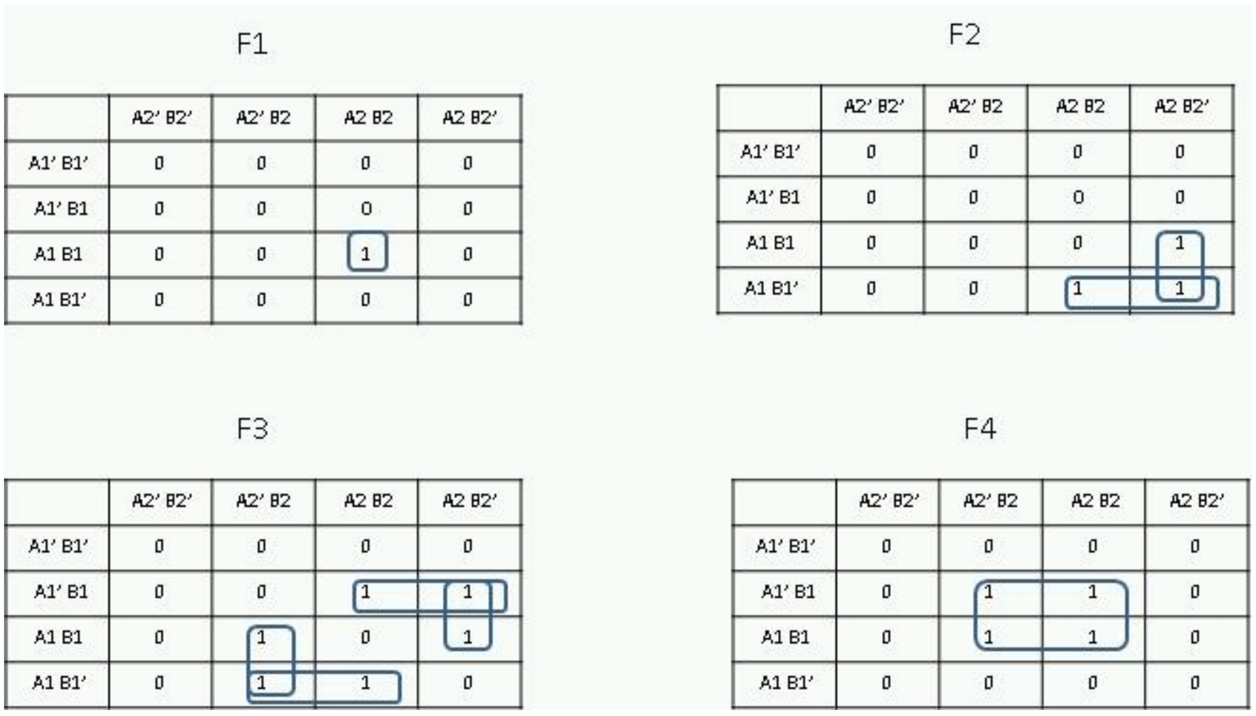
Questions

1. Derive the Boolean expressions for 2-bit multiplier. Each output bit should be represented by a different Boolean expression. (10 points)

The two input numbers are represented as A1 B1 and A2 B2 and their product is represented as F1 F2 F3 F4

Input 1		**Input 2**		**Output**			
A1	B1	A2	B2	F1	F2	F3	F4
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

The Karnaugh Map for the output variables are the following:



F1

	A2' B2'	A2' B2	A2 B2	A2 B2'
A1' B1'	0	0	0	0
A1' B1	0	0	0	0
A1 B1	0	0	1	0
A1 B1'	0	0	0	0

F2

	A2' B2'	A2' B2	A2 B2	A2 B2'
A1' B1'	0	0	0	0
A1' B1	0	0	0	0
A1 B1	0	0	0	1
A1 B1'	0	0	1	1

F3

	A2' B2'	A2' B2	A2 B2	A2 B2'
A1' B1'	0	0	0	0
A1' B1	0	0	1	1
A1 B1	0	1	0	1
A1 B1'	0	1	1	0

F4

	A2' B2'	A2' B2	A2 B2	A2 B2'
A1' B1'	0	0	0	0
A1' B1	0	1	1	0
A1 B1	0	1	1	0
A1 B1'	0	0	0	0

$$F1 = A1\,B1\,A2\,B2$$

$$F2 = A1\,A2\,\overline{B2} + A1\,\overline{B1}\,A2$$

$$F3 = A1\,\overline{A2}\,B2 + A1\,\overline{B1}\,B2 + \overline{A1}\,B1\,A2 + B1\,A2\,\overline{B2}$$

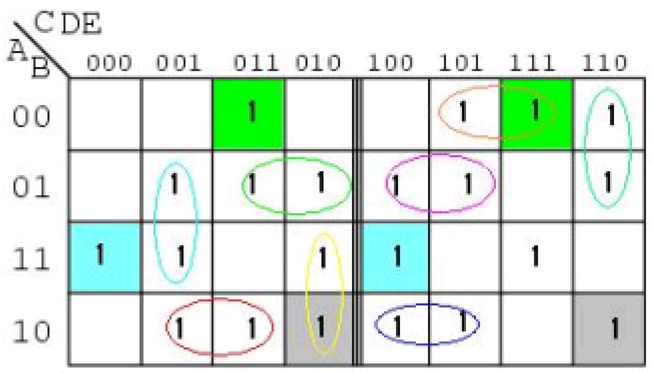
$$F4 = B1\,B2$$

2. Design a circuit of 5 input variables that generates output 1 if and only if the number of 1's in the input is prime (i.e., 2, 3 or 5). (10 points)

Truth table for the output function is given as follows:

Table 1. Truth table

Decimal	Binary	# of 1's	Function
0	00000	0	0
1	00001	1	0
2	00010	1	0
3	00011	2	1
4	00100	1	0
5	00101	2	1
6	00110	2	1
7	00111	3	1
8	01000	1	0
9	01001	2	1
10	01010	2	1
11	01011	3	1
12	01100	2	1
13	01101	3	1
14	01110	3	1
15	01111	4	0
16	10000	1	0
17	10001	2	1
18	10010	2	1
19	10011	3	1
20	10100	2	1
21	10101	3	1
22	10110	3	1
23	10111	4	0
24	11000	2	1
25	11001	3	1
26	11010	3	1
27	11011	4	0
28	11100	3	1
29	11101	4	0
30	11110	4	0
31	11111	5	1



5- variable Karnaugh map (overlay)

Figure 1. Truth table in overlap map

In the overlay mode, the K-maps are laid one above the other just as shown in Figure 2, and then the grouping is done.

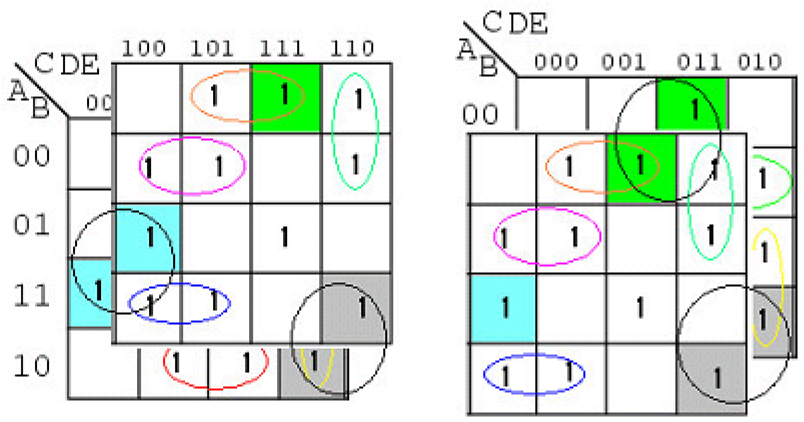


Figure 2. K-maps in overlay mode

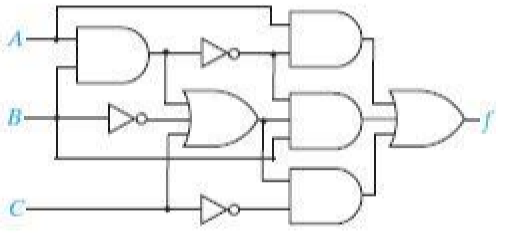
Then, the simplified expression becomes
Y = BC'D'E + A'BC'D + AC'DE' + AB'C'D + A'B'CE + A'CDE' + A'BCD + AB'CD' + ABD'E + AB'DE' + A'B'DE + ABCDE

3. Please use the Quine-McCluskey algorithm to find all of the prime implicants of the function F(A, B, C, D) = Σm(2, 4, 5, 6, 9, 10, 11, 12, 13, 15). (10 points)

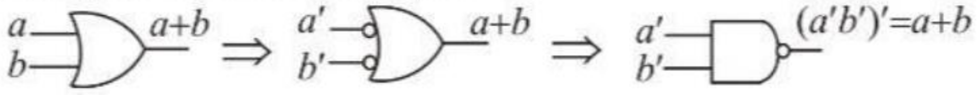
2	0010✓	2, 6	0-10 a'cd'	4, 5, 12, 13	-10- bc'
4	0100✓	2, 10	-010 b'cd'	~~4, 12, 5, 13~~	~~-10-~~
5	0101✓	4, 5	010-✓	9, 11, 13, 15	1--1 ad
6	0110✓	4, 6	01-0 a'bd'	~~9, 13, 11, 15~~	~~1--1~~
9	1001✓	4, 12	-100✓		
10	1010✓	5, 13	-101✓		
12	1100✓	9, 11	10-1✓		
11	1011✓	9, 13	1-01✓		
13	1101✓	10, 11	101- ab'c		
15	1111✓	12, 13	110-✓		
		11, 15	1-11✓		
		13, 15	11-1✓		

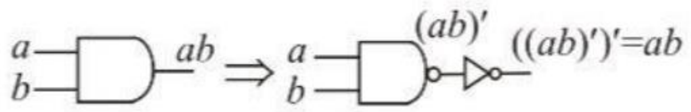
Prime implicants: *ad, bc', a'cd', b'cd', a'bd', ab'c*

4. Implement the below logic circuit by using only NAND gates. (Do not simplify except to delete two inverters in series, if any). (10 points)

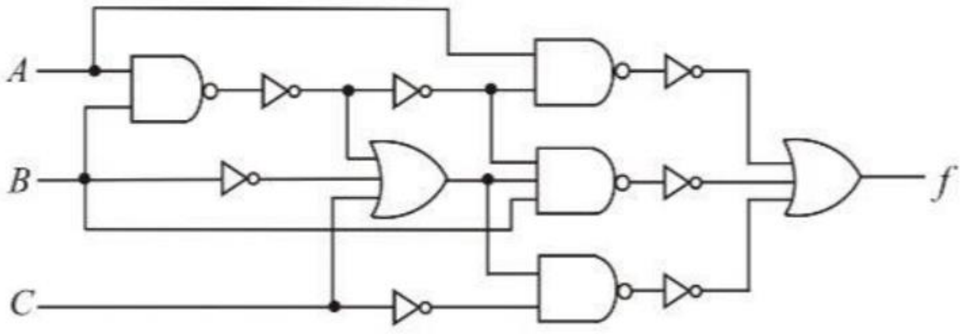


Using the gate equivalent to convert the given circuit into a four-level circuit containing only NAND gates. Convert all AND gates into NAND gates by adding an inversion bubbles at the outputs. Convert all OR gates into NAND gates by adding inversion bubbles at the inputs.

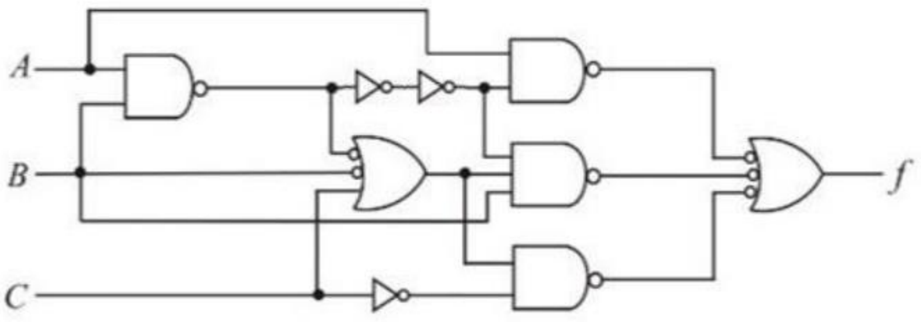




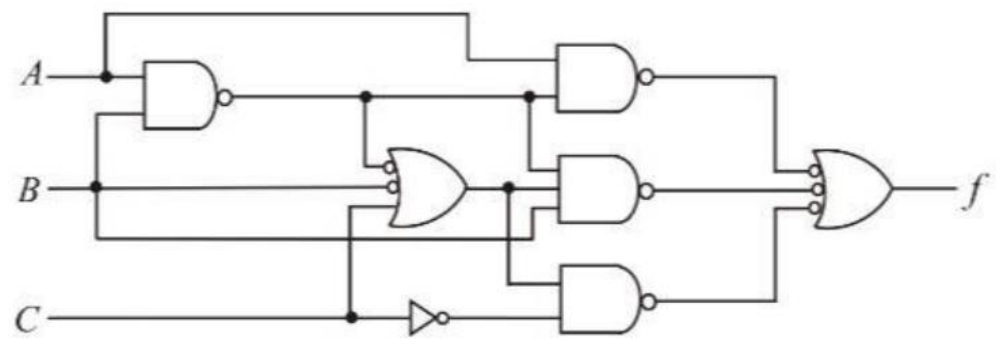
Thus, the AND gates are replaced by equivalent NAND gates in the given four-level circuit.
Then the circuit is redrawn as

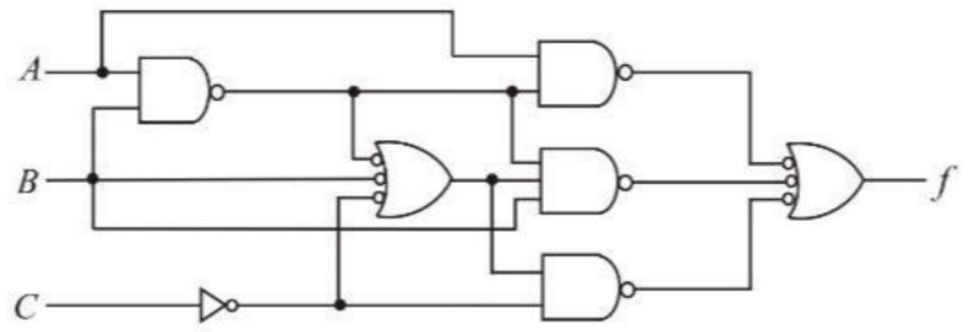


the above circuit is redrawn as

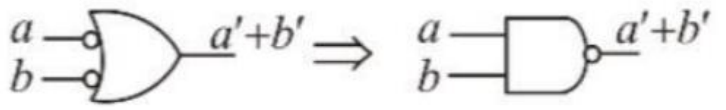


Modified circuit diagram is as follows:

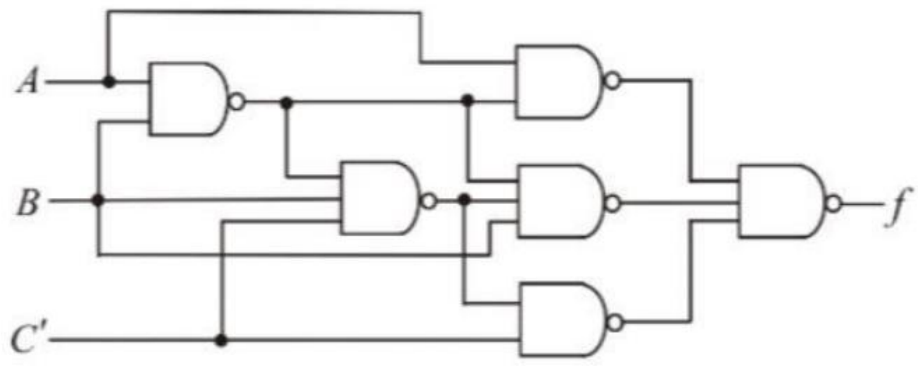




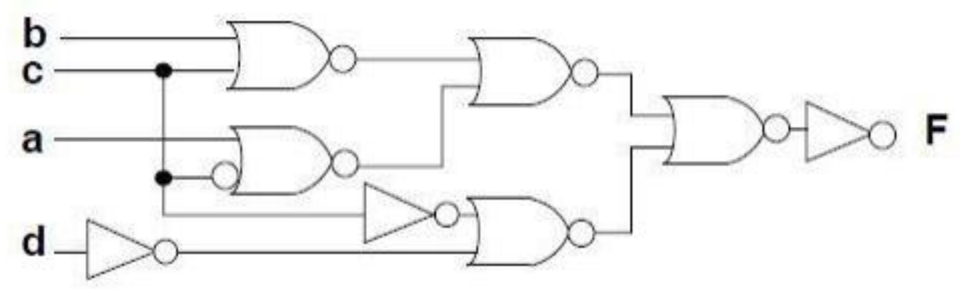
As we know that



Then, the bubbled OR gate is replaced with an NAND gate.
Simplified circuit diagram is as follows:



5. a) Find the hazards in the below circuit. (5 points)
   b) Modify the circuit so that it is hazard free. (10 points)

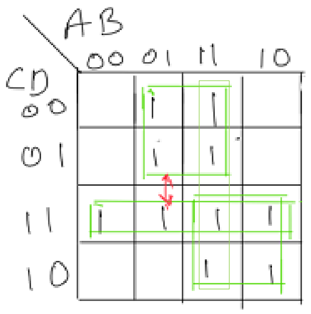


a) $F = \overline{\overline{\overline{\overline{b+c} + \overline{a+\bar{c}}} + \overline{\bar{c}+\bar{d}}}}$

$= (b + c)(a + \bar{c}) + cd$

$= ab + ac + b\bar{c} + cd$

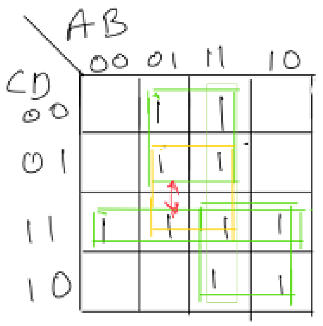
K-Map



CD \ AB	00	01	11	10
00		1	1	
01		1	1	
11	1	1	1	1
10			1	1

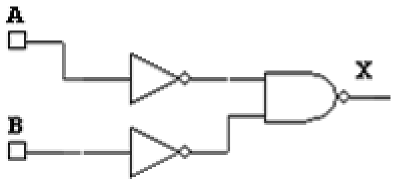
The above K-Map shows the error.

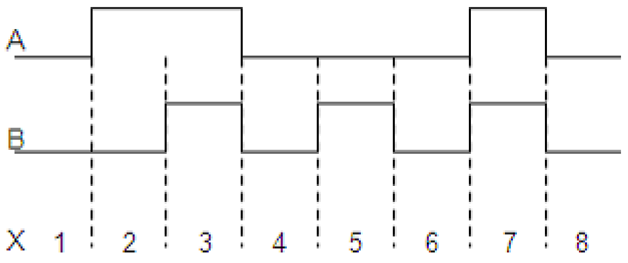
b) The K-map below removes the hazard by adding the term $b\bar{d}$

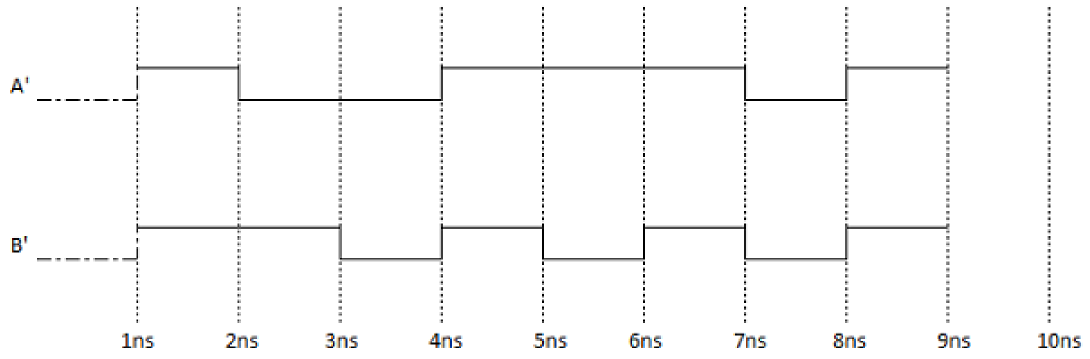


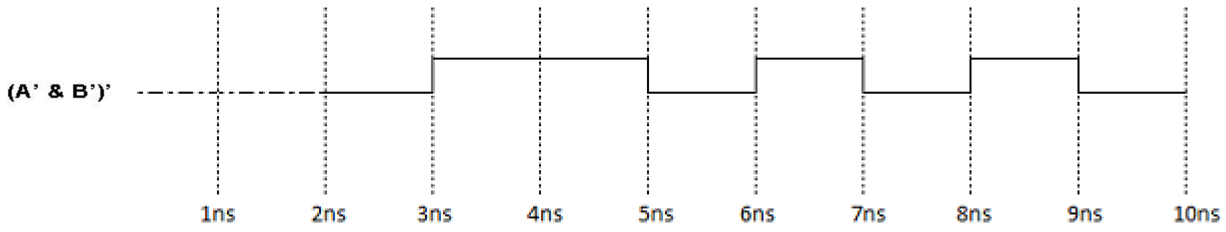
CD \ AB	00	01	11	10
00		1	1	
01		1	1	
11	1	1	1	1
10			1	1

6. Complete the timing diagram for the given circuit. Assume that all gates have a propagation delay of 1 ns. (10 points)









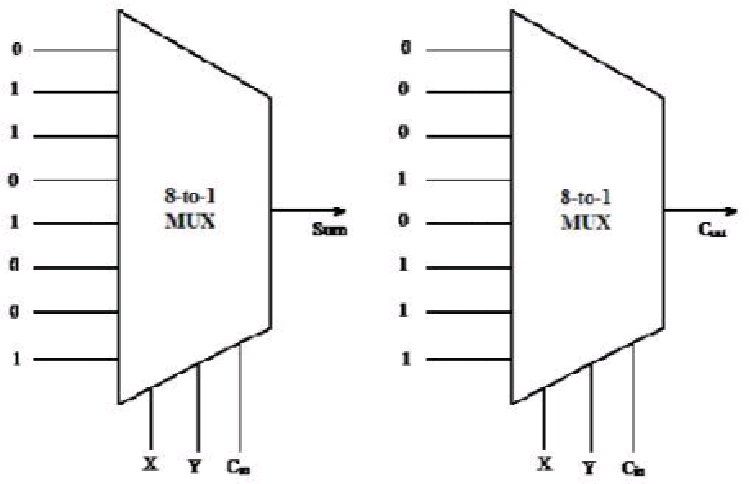
7. Implement a full adder
   (a) using two 8-to-1 MUXes. Connect X,Y, and Cin to the control inputs of the MUXes and connect 1 or 0 to each data input. (5 points)

Let X, Y and C_{in} be the inputs to the multiplexer and Sum and C_{out} be the outputs.

Truth table:

Inputs			Outputs	
X	Y	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Implementing Sum and C_{out} using 8-to-1 MUX's:



   (b) using two 4-to-1 MUXes and one inverter. Connect X and Y to the control inputs of the MUXes, and connect 1's, 0's, Cin, or C′in to each data input. (5 points)

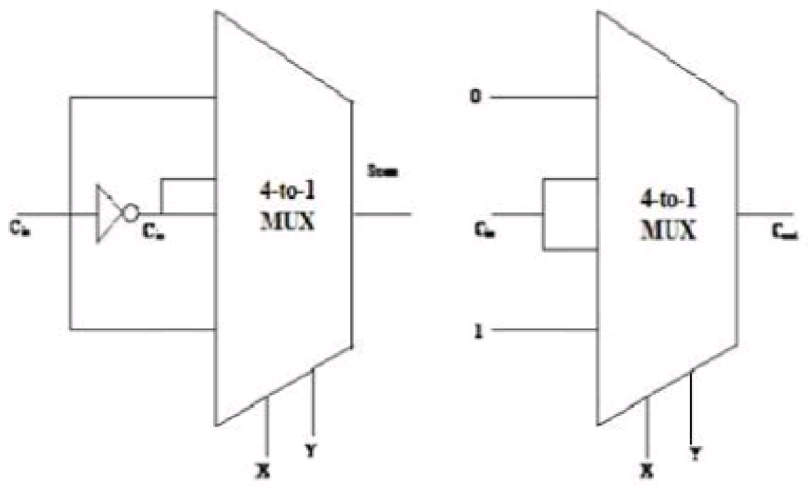
Implementing Sum and C_{out} using 4-to-1 MUX's:

When $X = 0, Y = 0$, then Sum depends on C_{in}.

When $X = 0, Y = 1$, then Sum depends on C_{in}'.

When $X = 1, Y = 0$, then Sum depends on C_{in}'.

When $X = 1, Y = 1$, then Sum depends on C_{in}.



When $X = 0, Y = 0$, no carry occurs whether C_{in} is 0 or 1.

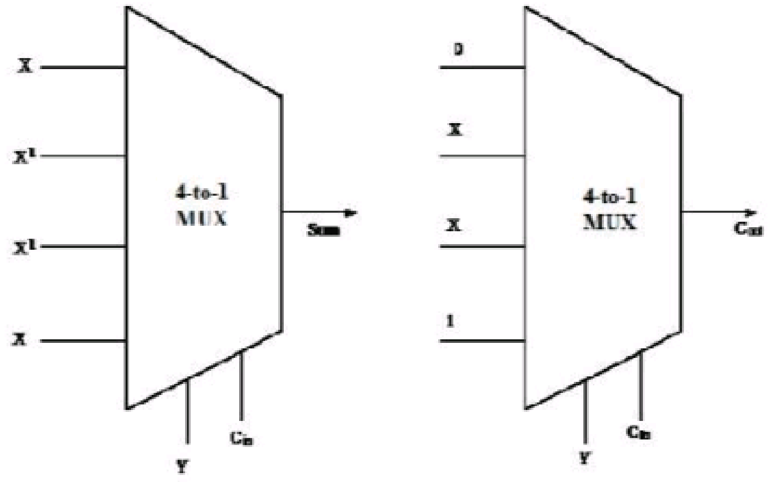
When $X = 0, Y = 1$, then C_{out} depends on C_{in}.

When $X = 1, Y = 0$, then C_{out} depends on C_{in}.

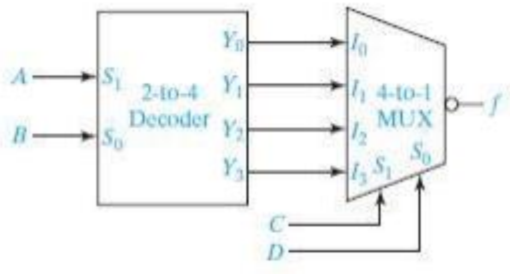
When $X = 1, Y = 1$, then C_{out} is 1 independent of C_{in} is 0 or 1.

(c) again using two 4-to-1 MUXes, but this time connect Cin and Y to the control inputs of the MUXes, and connect 1's, 0's, X, or X′ to each data input. Note that in this fashion, any N-variable logic function may be implemented using a 2(N−1)-to-1 MUX. (5 points)

Implementing the case where C_{in} and Y are control inputs using 4-to-1 MUX:



8. The circuit below has a 2-to-4 decoder with active high outputs connected to a 4-to-1 MUX with an active low output. Derive a minimum SOP or a minimum POS expression for the output, f(A, B, C, D). (10 points)



Consider the following truth table for the 2-to-4 decoder with active high outputs:
Table 1: Truth table for 2-to-4 decoder

Input		Output			
A	*B*	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Express each output values Y_0, Y_1 , Y_2 and Y_3 from the truth table illustrated in Table 1 into their respective sum-of-products (SOP) logical expression.

Y_0 = A'B'
Y_1 = A'B
Y_2 = AB'
Y_3 = AB

Consider the truth table for the 4-to-1 MUX

Input		Output
C	*D*	*f*
0	0	$I_0' = Y_0'$
0	1	$I_1' = Y_1'$
1	0	$I_2' = Y_2'$
1	1	$I_3' = Y_3'$

Output of the multiplexer in the Sum of products representation:
f = C'D'I_0' + C'DI_1' + CD'I_2' + CDI_3'
f = C'D'$Y_{0'}$ + C'DY_1' + CD'Y_2' + CDY_3'

Substituting the expression for Y’s in the above equation gives:

$$\begin{aligned} f &= C'D'(A'B')' + C'D(A'B)' + CD'(AB')' + CD(AB)' \\ &= C'D'(A+B) + C'D(A+B') + CD'(A'+B) + CD(A'+B') \\ &= AC'D' + BC'D' + AC'D + B'C'D + A'CD' + BCD' + A'CD + B'CD \\ &= AC'D' + AC'D + B'C'D + B'CD + A'CD' + A'CD + BC'D' + BCD' \end{aligned}$$

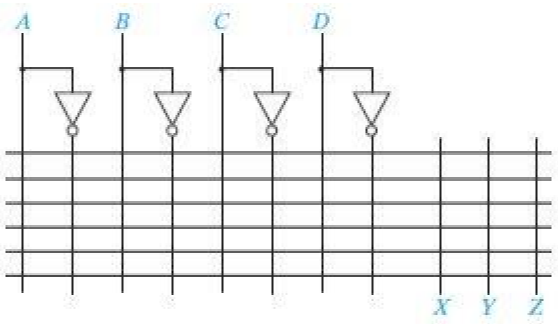
Simplify further.

$$\begin{aligned} f &= AC'(D'+D) + B'D(C'+C) + A'C(D'+D) + BD'(C'+C) \\ &= AC' + A'C + BD' + B'D \end{aligned}$$

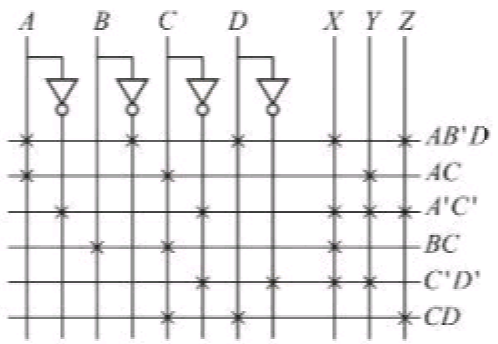
Therefore the minimum SOP expression for the output $f(A,B,C,D)$ is

$$\boxed{f(A,B,C,D) = AC' + A'C + BD' + B'D}.$$

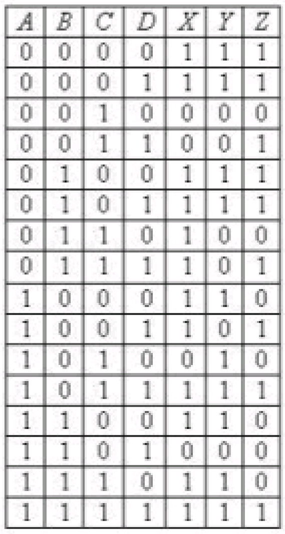
9. The following PLA will be used to implement the following equations:
X = AB′D + A′C′ + BC + C′D′
Y = A′C′ + AC + C′D′
Z = CD + A′C′ + AB′D
(a) Indicate the connections that will be made to program the PLA to implement these equations. (5 points)
(b) Specify the truth table for a ROM which realizes these same equations. (5 points)



(a)
Implementing PLA by the equations



(b)
Truth table for a ROM by using the equations.



A	B	C	D	X	Y	Z
0	0	0	0	1	1	1
0	0	0	1	1	1	1
0	0	1	0	0	0	0
0	0	1	1	0	0	1
0	1	0	0	1	1	1
0	1	0	1	1	1	1
0	1	1	0	1	0	0
0	1	1	1	1	0	1
1	0	0	0	1	1	0
1	0	0	1	1	0	1
1	0	1	0	0	1	0
1	0	1	1	1	1	1
1	1	0	0	1	1	0
1	1	0	1	0	0	0
1	1	1	0	1	1	0
1	1	1	1	1	1	1