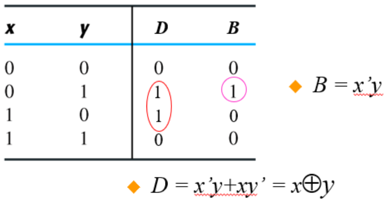
Final Examination Prep 2023

Q1) Create the following simple combinational logic circuits:

(a) Half-Subtractor (Hint: input 2 bits (X,Y)/output 2 bits (D, B) of difference between two inputs and borrow. Design step - First, design the truth table, then find the Boolean function from the truth table. Finally, draw the logic circuit.)

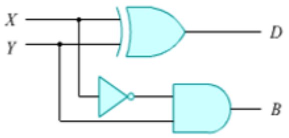
ANS)



x	y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

◆ $B = x'y$

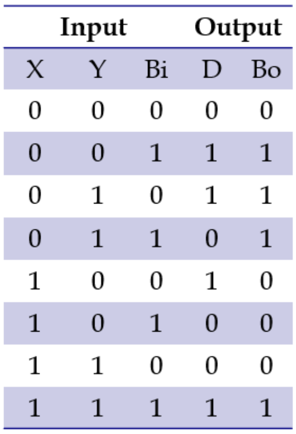
◆ $D = x'y + xy' = x \oplus y$



$D = \overline{X}Y + X\overline{Y} = X \oplus Y$

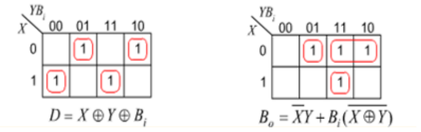
$B = \overline{X}Y$

(b) Full-Subtractor (Hint: input 3 bits (X,Y, and Bi); Bi is for borrow input/output 2 bits (D, Bo) of difference between two inputs and borrow output. Design step - First, design the truth table. Second simplify the Boolean function using the Karnaugh map. Third, find the simplified Boolean function from the Karnaugh map. Finally, draw the logic circuit.)



Input			Output	
X	Y	Bi	D	Bo
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

◆ *Simplified it through k-map*

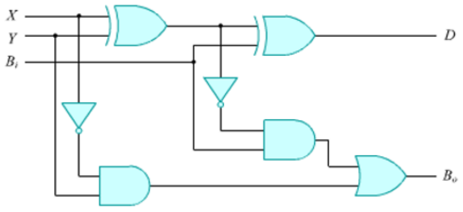


$X \backslash YB_i$	00	01	11	10
0		1		1
1	1		1	

$D = X \oplus Y \oplus B_i$

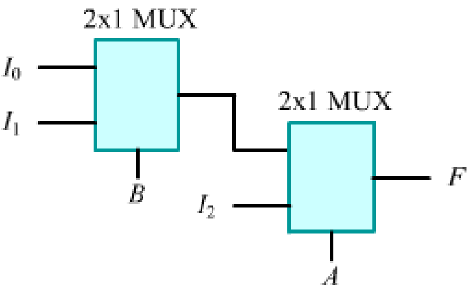
$X \backslash YB_i$	00	01	11	10
0		1	1	1
1			1	

$B_o = \overline{X}Y + B_i(\overline{X \oplus Y})$

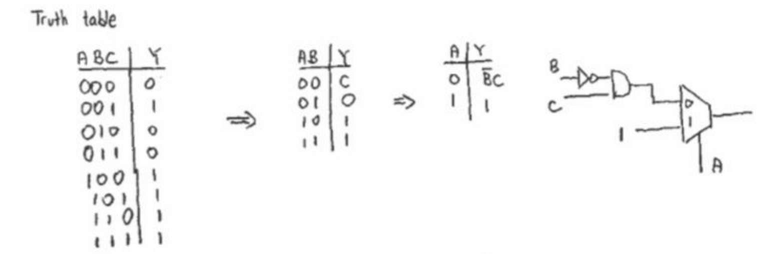


(c) Design a 3-1 multiplexer using two of 2 to 1 multiplexers. The input selection is observed the below rules.

- Rule 1) If AB = 00, select I_0
- Rule 2) If AB = 01, select I_1
- Rule 3) If AB=1x, select I_2



Q2) Draw a schematic to show how the function $Y=AB+B\overline{}C+AC\overline{}$ can be implemented with a 2:1 multiplexer, one inverter, and one two-input AND gate.


Truth table

ABC	Y
000	0
001	1
010	0
011	0
100	1
101	1
110	1
111	1

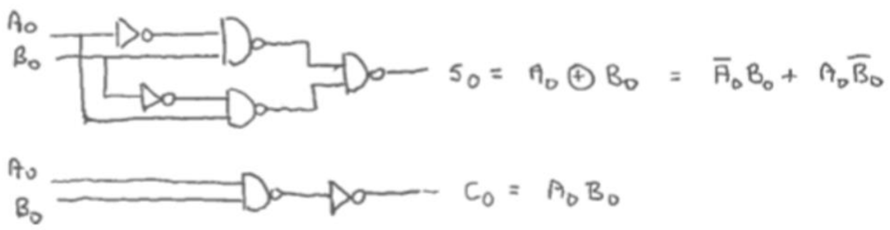
⇒

AB	Y
00	C
01	0
10	1
11	1

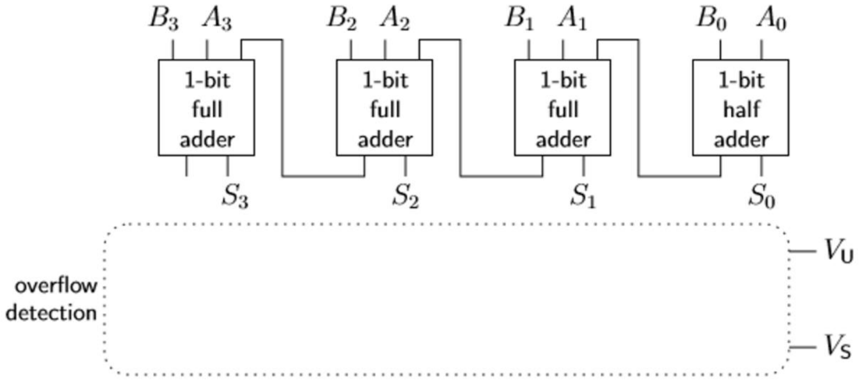
⇒

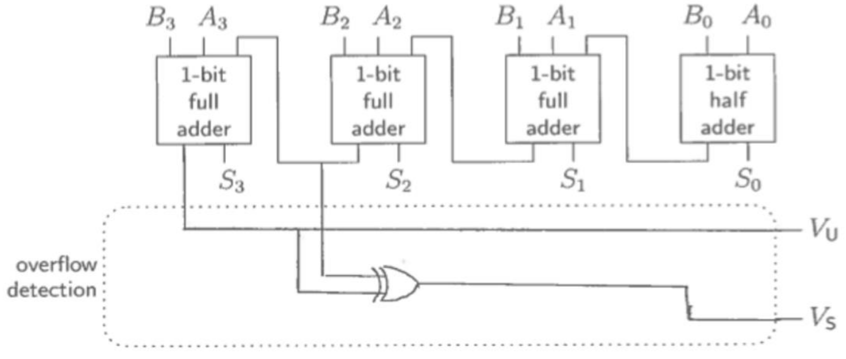
A	Y
0	$\overline{B}C$
1	1

(a) Unlike a 1-bit full-adder, which has three inputs, a 1-bit half-adder circuit computes the sum and carry-out of only two input bits. Use of a half-adder is shown in the four-bit adder of part (e) below. Draw a schematic for a half-adder circuit, using only inverters and two-input NAND gates. Use A and B as names for inputs, and S and Co as names for output.

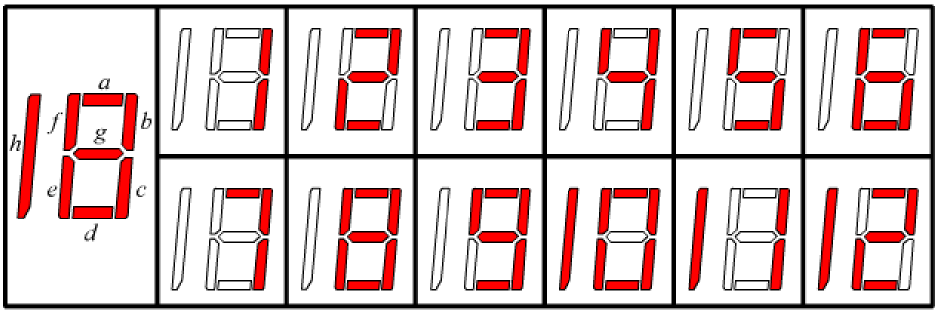

A_0
B_0
$S_0 = A_0 \oplus B_0 = \overline{A}_0 B_0 + A_0 \overline{B}_0$
A_0
B_0
$C_0 = A_0 B_0$

(b) Add wires and logic gates within the area labeled "overflow detection" so that VU = 1 indicates unsigned overflow and VS = 1 indicates signed overflow. You can connect your gates to whichever wires of the four-bit adder you want to use.





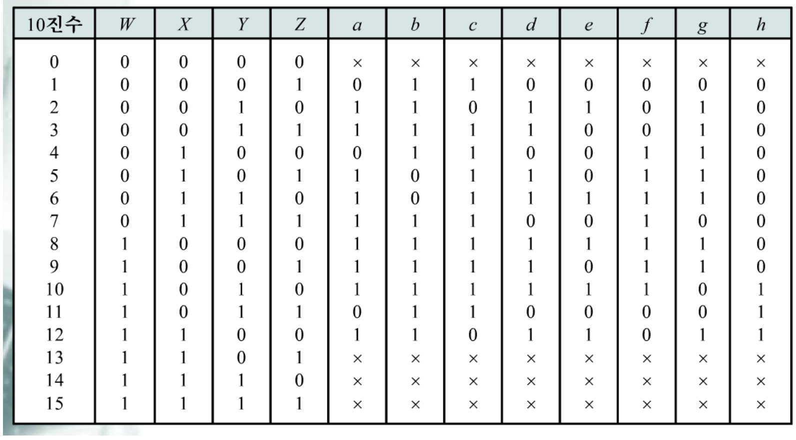
Q3) We want to design a decoder for an 8-segment LED to indicate numbers from 1 to 12 on a table watch. However, the numbers 0, 13, 14, and 15 should not be displayed. Follow these design steps:



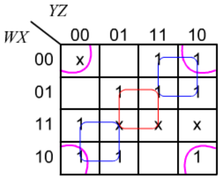
Step 1: Create a truth table using input variables W, X, Y, Z, and output variables a~h for the 8-segment LED.

Step 2: Simplify the output functions using the Karnaugh map to minimize the logic expressions.

Step 3: Draw a logic circuit based on the simplified output functions, incorporating the necessary gates and connections to drive the 8-segment LED.

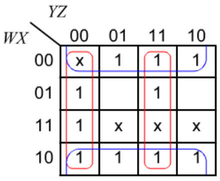


10진수	W	X	Y	Z	a	b	c	d	e	f	g	h
0	0	0	0	0	×	×	×	×	×	×	×	×
1	0	0	0	1	0	1	1	0	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1	0
3	0	0	1	1	1	1	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0	0	1	1	0
5	0	1	0	1	1	0	1	1	0	1	1	0
6	0	1	1	0	1	0	1	1	1	1	1	0
7	0	1	1	1	1	1	1	0	0	1	0	0
8	1	0	0	0	1	1	1	1	1	1	1	0
9	1	0	0	1	1	1	1	1	0	1	1	0
10	1	0	1	0	1	1	1	1	1	1	0	1
11	1	0	1	1	0	1	1	0	0	0	0	1
12	1	1	0	0	1	1	0	1	1	0	1	1
13	1	1	0	1	×	×	×	×	×	×	×	×
14	1	1	1	0	×	×	×	×	×	×	×	×
15	1	1	1	1	×	×	×	×	×	×	×	×



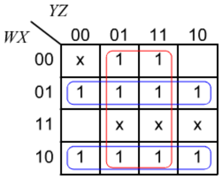
WX \ YZ	00	01	11	10
00	x		1	1
01		1	1	1
11	1	x	x	x
10	1	1		1

$a = (W \oplus Y) + (X \oplus Z)$



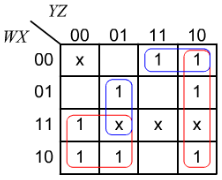
WX \ YZ	00	01	11	10
00	x	1	1	1
01	1		1	
11	1	x	x	x
10	1	1	1	1

$b = \overline{X} + (Y \oplus Z)$



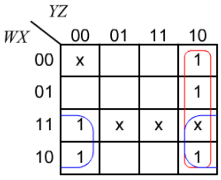
WX \ YZ	00	01	11	10
00	x	1	1	
01	1	1	1	1
11		x	x	x
10	1	1	1	1

$c = Z + (W \oplus X)$



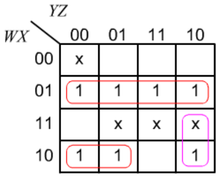
WX \ YZ	00	01	11	10
00	x		1	1
01		1		1
11	1	x	x	x
10	1	1		1

$d = W\overline{Y} + Y\overline{Z} + \overline{W}\,\overline{X}Y + X\overline{Y}Z$



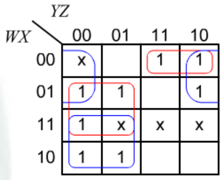
WX \ YZ	00	01	11	10
00	x			1
01				1
11	1	x	x	x
10	1			1

$e = W\overline{Z} + Y\overline{Z}$



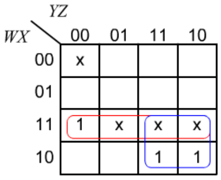
WX \ YZ	00	01	11	10
00	x			
01	1	1	1	1
11		x	x	x
10	1	1		1

$f = \overline{W}X + W\overline{X}\,\overline{Y} + WY\overline{Z}$



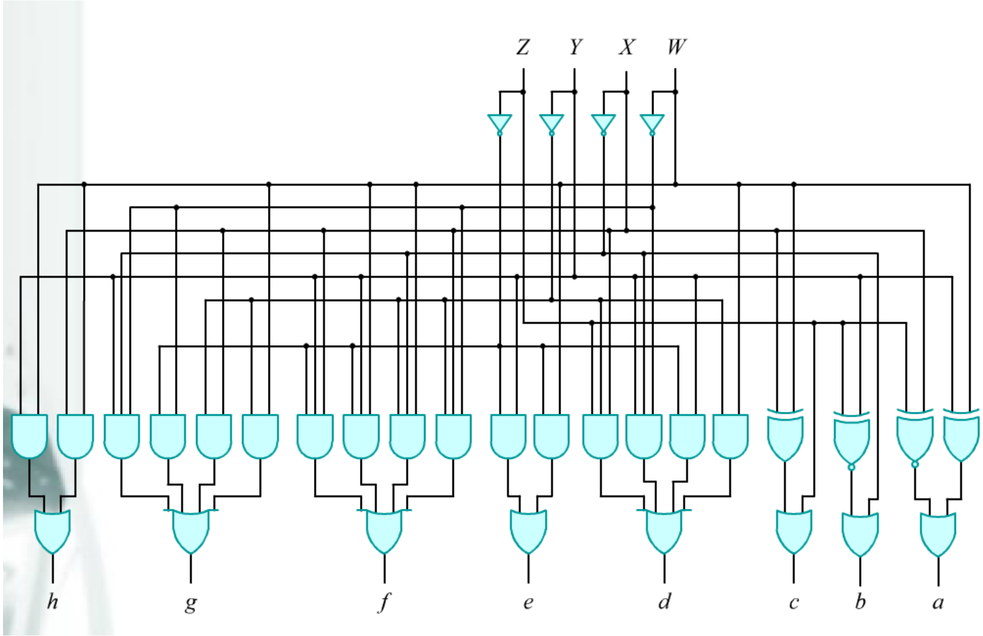
WX \ YZ	00	01	11	10
00	x		1	1
01	1	1		1
11	1	x	x	x
10	1	1		

$g = W\overline{Y} + X\overline{Y} + \overline{W}\,\overline{Z} + \overline{W}\,\overline{X}Y$



WX \ YZ	00	01	11	10
00	x			
01				
11	1	x	x	x
10			1	1

$h = WX + WY$


Z
Y
X
W
h
g
f
e
d
c
b
a

Q4) Design a 3-excess to BCD code converter using the following design steps:

Step 1: Construct a truth table for the input 3-excess code (A, B, C, D) and the output BCD code (W, X, Y, Z).
Step 2: Simplify the output functions using a 4-variable Karnaugh map to minimize the logic expressions.
Step 3: Draw a logic circuit based on the simplified functions, incorporating the necessary gates and connections to implement the 3-excess to BCD code conversion.

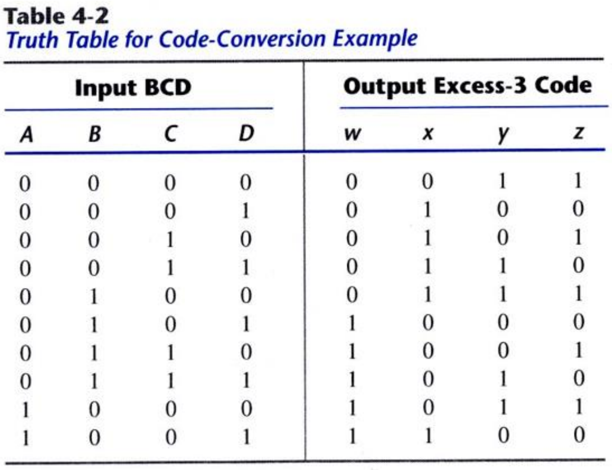
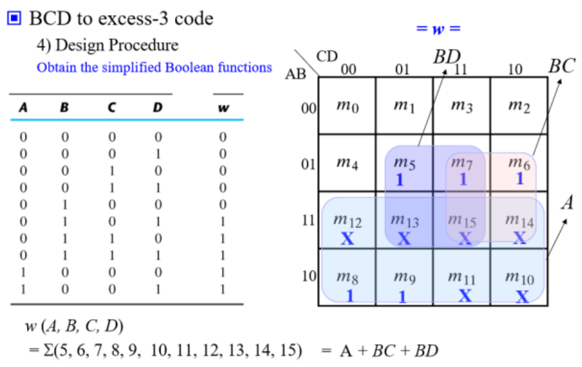


Table 4-2
Truth Table for Code-Conversion Example

Input BCD				Output Excess-3 Code			
A	*B*	*C*	*D*	*w*	*x*	*y*	*z*
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0



▣ BCD to excess-3 code

4) Design Procedure

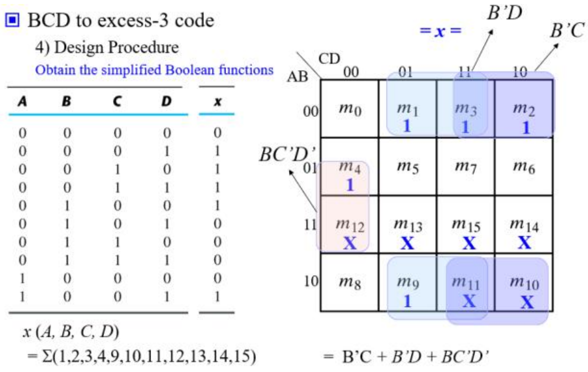
Obtain the simplified Boolean functions

A	*B*	*C*	*D*	*w*
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1

= *w* =

AB \ CD	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5 1	m_7 1	m_6 1
11	m_{12} X	m_{13} X	m_{15} X	m_{14} X
10	m_8 1	m_9 1	m_{11} X	m_{10} X

w (*A, B, C, D*)
= Σ(5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15) = A + *BC* + *BD*



▣ BCD to excess-3 code

4) Design Procedure

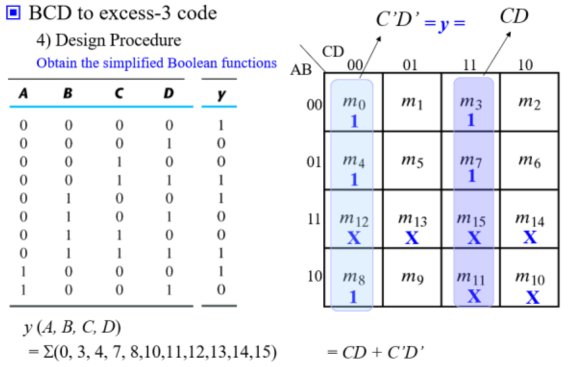
Obtain the simplified Boolean functions

A	*B*	*C*	*D*	*x*
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1

= *x* =

AB \ CD	00	01	11	10
00	m_0	m_1 1	m_3 1	m_2 1
01	m_4 1	m_5	m_7	m_6
11	m_{12} X	m_{13} X	m_{15} X	m_{14} X
10	m_8	m_9 1	m_{11} X	m_{10} X

x (*A, B, C, D*)
= Σ(1,2,3,4,9,10,11,12,13,14,15) = B'C + *B'D* + *BC'D'*



▣ BCD to excess-3 code

4) Design Procedure

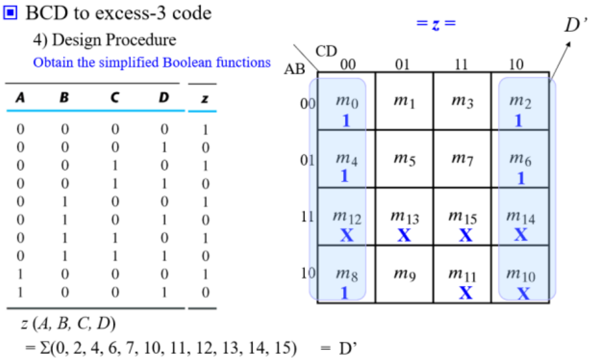
Obtain the simplified Boolean functions

A	*B*	*C*	*D*	*y*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0

C'D' = *y* = *CD*

AB \ CD	00	01	11	10
00	m_0 1	m_1	m_3 1	m_2
01	m_4 1	m_5	m_7 1	m_6
11	m_{12} X	m_{13} X	m_{15} X	m_{14} X
10	m_8 1	m_9	m_{11} X	m_{10} X

y (*A, B, C, D*)
= Σ(0, 3, 4, 7, 8,10,11,12,13,14,15) = *CD* + *C'D'*



▣ BCD to excess-3 code

4) Design Procedure

Obtain the simplified Boolean functions

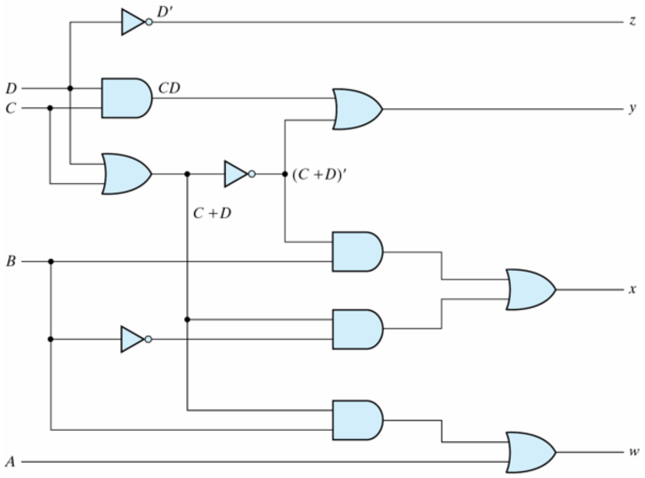
A	*B*	*C*	*D*	*z*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0

= *z* =

AB \ CD	00	01	11	10
00	m_0 1	m_1	m_3	m_2 1
01	m_4 1	m_5	m_7	m_6 1
11	m_{12} X	m_{13} X	m_{15} X	m_{14} X
10	m_8 1	m_9	m_{11} X	m_{10} X

z (*A, B, C, D*)
= Σ(0, 2, 4, 6, 7, 10, 11, 12, 13, 14, 15) = D'

4) Design Procedure: Draw the logic diagram



Logic Diagram for BCD to Excess-3 Code Converter

$z = D'$

$y = CD + C'D' = CD + (C+D)'$

$x = B'C + B'D + BC'D'$

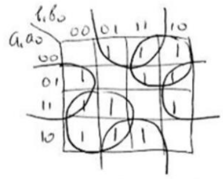
$\quad = B'(C+D) + B(C+D)'$

$w = A + BC + BD$

Q5) Consider a two-bit magnitude comparator circuit. It has two-bit inputs A = a_1a_0 and B= b_1b_0 (where, a_1 and b_1 are the most-significant bits, which mean left-most-bits). The output f should be f =1 when A ≠ B; otherwise f =0.

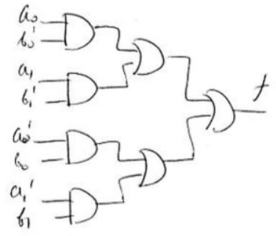
(a) Find a minimal SOP expression for f. Draw the logic diagram of an AND-OR circuit implementation of f using only 2-input gates. Using whatever answer, you obtain here to answer parts (b)-(d) below. (Design Steps: Truth Table – simplified using Karnaugh map – draw the logic circuit)

a_1 a_0 b_1 b_0	f
0 0 0 0	0
0 0 0 1	1
0 0 1 0	1
0 0 1 1	1
0 1 0 0	1
0 1 0 1	0
0 1 1 0	1
0 1 1 1	1
1 0 0 0	1
1 0 0 1	1
1 0 1 0	0
1 0 1 1	1
1 1 0 0	1
1 1 0 1	1
1 1 1 0	1
1 1 1 1	0

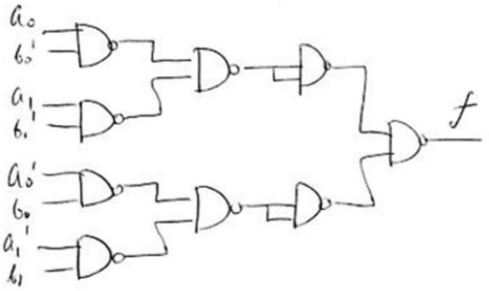


a_1a_0 \ b_1b_0	00	01	11	10
00		1	1	1
01	1		1	1
11	1	1		1
10	1	1	1	

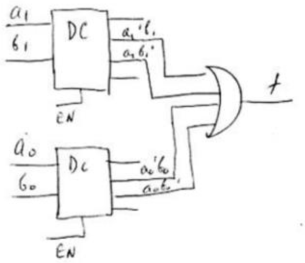
$f = a_0 b_0' + a_1 b_1' + a_0' b_0 + a_1' b_1$



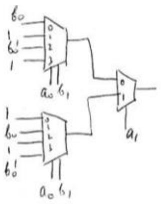
(b) By applying algebraic transformations and circuit manipulation, design a circuit using only 2-input NAND gates to implement the logic diagram from part (a). You cannot use inverters in this implementation.



(c) Implement the logic function f using 2-to-4 decoders. You can utilize additional AND, OR, and NOT gates as needed. Each decoder should include an ENABLE input.

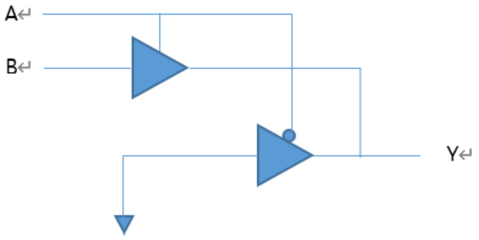


(d) Implement the logic function f using two 4-to-1 multiplexers and one 2-to-1 multiplexer. You can incorporate additional AND, OR, and NOT gates if necessary.



$a_1 a_0 b_1 b_0$	f	
0000	0	$f = b_0$ || $a_1 = 0$, $a_0 b_1 = 00$
0001	1	
0010	1	$f = 1$ || $a_1 = 0$, $a_0 b_1 = 01$ etc.
0011	1	
0100	1	$f = b_0'$
0101	0	
0110	1	$f = 1$
0111	1	
1000	1	$f = 1$
1001	1	
1010	0	$f = b_0$
1011	1	
1100	1	$f = 1$
1101	1	
1110	1	$f = b_0'$
1111	0	

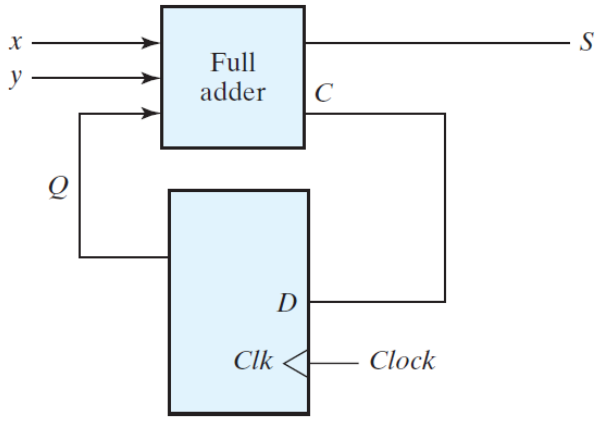
(e) The circuit below uses two tristate buffers (or, tristate gates), one with active-high enable, and one with active-low enable. What simple logic function of A and B does the circuit implement? Give a reason for your answer using truth table.



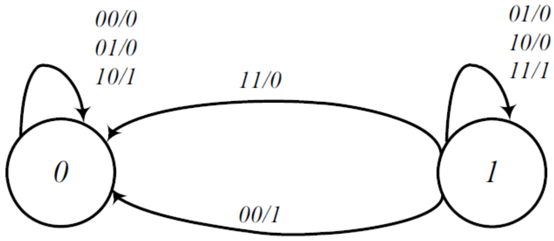
A	B	Y
0	0	0 (from GND)
0	1	0 (from GND)
1	0	0 (from B)
1	1	1 (from B)

The table shows that the circuit acts as an AND gate.

Q6) A sequential circuit has one flip-flop Q, two inputs x and y, and one output S . It consists of a full-adder circuit connected to a D flip-flop, as shown below. Derive the state table and state diagram of the sequential circuit.

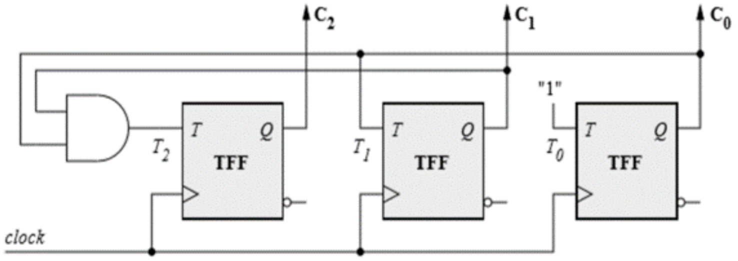


Present state	Inputs		Next state	Output
Q	x	y	Q	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



$$S = x \oplus y \oplus Q$$
$$Q(t + 1) = xy + xQ + yQ$$

Q7) Consider the circuit depicted below.



(a) Derive the state table and the state diagram for this circuit and determine the type of the mealy machine. The outputs of the systems are C_2, C_1, C_0 (treat C_2 as the most-significant bit; that is, left-most-bit).

(b) Based on your answer to part (a), what does this circuit implement?

(c) Use a method of your choosing to redesign this system using JK flip-flops, and sketch the resulting circuit.

(a)

Here, Q_2=C_2, $Q_1 = C_1$, Q_0=C_0

Since we have T FF, the TFF characteristic equation is for C_2^+, C_1^+, C_0^+

$C_2^+ = C_2 \oplus T_2$, $C_1^+ = C_1 \oplus T_1$, $C_0^+ = C_0 \oplus T_0$

Also, from the circuit, $T_0 = 1$, $T_1 = C_0$, $T_2 = C_0 \cdot C_1$

$C_2^+ = C_2 \oplus C_0 \cdot C_1$, $C_1^+ = C_1 \oplus C_0$, $C_0^+ = C_0 \oplus 1 = \overline{C_0}$

ANS:

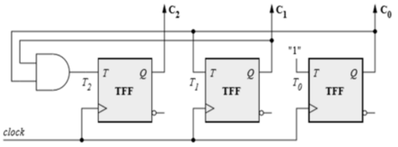
Use

$C_2^+ = C_2 \oplus C_0 \cdot C_1$,

$C_1^+ = C_1 \oplus C_0$,

$C_0^+ = C_0 \oplus 1 = \overline{C_0}$ for Table

C_2	C_1	C_0	C_2^+	C_1^+	C_0^+
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

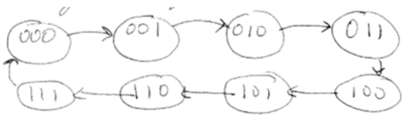


C_2	C_1	C_0	C_2^+	C_1^+	C_0^+
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

ANS:

Since there are e T FF, we have $2^3 = 8$ states encoded using values of C_2 C_1 C_0: 000, 001, …,111

The state diagram is given below:



(b) What does this circuit implements?
The circuit is simply binary up-counter.

(c) Redesign the circuit using JK FFs.

Let us use the table from (a); and add JK colums using the JK FF characteristic table:

CURRSTATE			NEXT STATE								
C_2	C_1	C_0	C_2^+	C_1^+	C_0^+	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

J_2:

$C_0 \backslash C_2C_1$	00	01	11	10
0	0	0	X	X
1	0	1	X	X

J_1:

$C_0 \backslash C_2C_1$	00	01	11	10
0	0	X	X	0
1	1	X	X	1

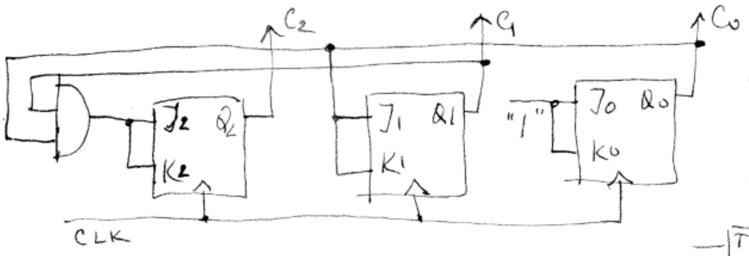
K_2:

$C_0 \backslash C_2C_1$	00	01	11	10
0	X	X	0	0
1	X	X	1	0

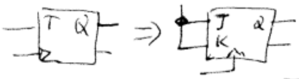
K_1:

$C_0 \backslash C_1C_2$				
	X	0	0	X
	X	1	1	X

$J_2 = K_2 = C_1 C_0$ $J_2 = K_2 = C_0$ $J_0 = K_0 = 1$



NOTE: YOU could use the initial circuit and substitute T FF with JK FF:

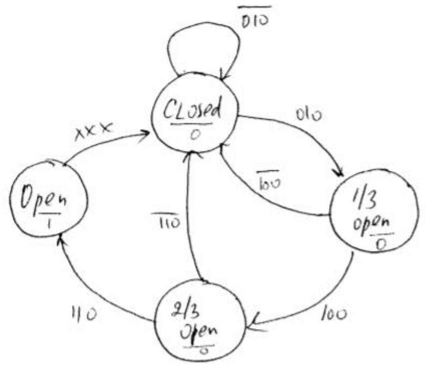


Q8) Design a sequential circuit for a three-number combination lock with inputs w, x, y, and output z. The lock can be opened by entering a specific three-bit binary code: 010, 100, 110 (the combination).

The lock operates as follows: Starting from the "closed" state, entering the code 010 transitions the system to the "1/3-open" state. Then, entering the code 100 moves it to the "2/3-open" state, but any other code entry returns it to the closed state. From the 2/3-open state, entering the code 110 opens the lock and transitions it to the "open" state. Any other code entry returns it to the closed state. When in the open state, the system generates an output z = 1, controlling the mechanical system that opens the lock. Finally, the system automatically transitions back to the closed state regardless of the input.

(a) Derive the state diagram for a Moor-type system, which will be used for parts (b)-(d).

(a)



(b) Assign the necessary number of bits (AB) to represent the states of the system: CLOSED = 00, 1/3-OPEN = 01, 2/3-OPEN = 10, OPEN = 11. Create the state table defining the three input variables w, x, y, current states (A, B), next states (A*, B*), and output z.

(b) STATES:
CLOSED = 00
1/3 OPEN = 01
2/3 OPEN = 10
OPEN = 11

STATE TABLE

CURR. STATE		INPUTS			NEXT STATE		OUT
A	B	W	X	Y	A*	B*	Z
0	0	0 X	1 X	0 X	0 0	1 0	0
0	1	1 X	0 X	0 X	1 0	0 0	0
1	0	1 X	1 X	0 X	1 0	1 0	0
1	1	X	X	X	0	0	1

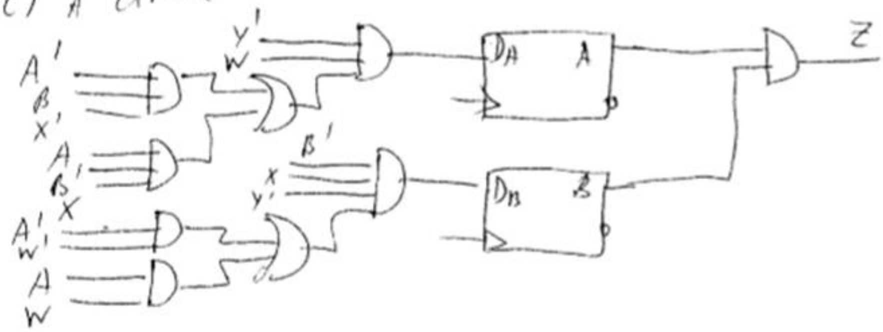
$A^* = A'B\,wx'y' + AB'wxy' = y'w(A'Bx' + AB'x)$

$B^* = A'B'w'xy' + AB'wxy' = B'xy'(A'w' + Aw)$

$Z = AB$

(c) Implement the system using D flip-flops and any required combinational gates, and sketch the circuit. Note that simplification of the functions may not yield significant results.

(c) A circuit with 2 FLIP-FLOPS is needed ; $D_A = A^*$, $D_B = B^*$


A'
B'
X'
A
B'
X
A'
W'
A
W
Y'
W
B'
X
Y'
D_A
A
D_B
B
Z