

**ECE2030A Introduction to Computer Engineering
Fall 2008**

Homework Assignment #2 Solution

Prepared by Thanh Tran and Prof. Hsien-Hsin S. Lee

1. (20%) Use Quine-McClusey Method to simplify the following canonical SOP forms. Find all the Essential Prime Implicants based the Q-M Method, and identify the non-Essential Prime Implicants. List all possible solutions of your results.

$$1.1. F(A,B,C,D,E) = \sum m(0,1,4,5,8,9,11,13,17,20,21,28,30,31) + d(10,12,16)$$

Convert to Binary

	A	B	C	D	E	
(0)	0	0	0	0	0	x
(1)	0	0	0	0	1	x
(4)	0	0	1	0	0	x
(5)	0	0	1	0	1	x
(8)	0	1	0	0	0	x
(9)	0	1	0	0	1	x
(10)	0	1	0	1	0	x
(11)	0	1	0	1	1	x
(12)	0	1	1	0	0	x
(13)	0	1	1	0	1	x
(16)	1	0	0	0	0	x
(17)	1	0	0	0	1	x
(20)	1	0	1	0	0	x
(21)	1	0	1	0	1	x
(28)	1	1	1	0	0	x
(30)	1	1	1	1	0	x
(31)	1	1	1	1	1	x

Sorted:

	A	B	C	D	E	
(0)	0	0	0	0	0	x
(1)	0	0	0	0	1	x
(4)	0	0	1	0	0	x
(8)	0	1	0	0	0	x
(16)	1	0	0	0	0	x
(5)	0	0	1	0	1	x
(9)	0	1	0	0	1	x
(10)	0	1	0	1	0	x
(12)	0	1	1	0	0	x
(17)	1	0	0	0	1	x
(20)	1	0	1	0	0	x
(11)	0	1	0	1	1	x
(13)	0	1	1	0	1	x
(21)	1	0	1	0	1	x
(28)	1	1	1	0	0	x
(30)	1	1	1	1	0	x
(31)	1	1	1	1	1	x

Loop 1

	A	B	C	D	E	
(0,1)	0	0	0	0	--	x
(0,4)	0	0	--	0	0	x
(0,8)	0	--	0	0	0	x
(0,16)	--	0	0	0	0	x
(1,5)	0	0	--	0	1	x
(1,9)	0	--	0	0	1	x
(1,17)	--	0	0	0	1	x
(4,5)	0	0	1	0	--	x
(4,12)	0	--	1	0	0	x
(4,20)	--	0	1	0	0	x
(8,9)	0	1	0	0	--	x
(8,10)	0	1	0	--	0	x
(8,12)	0	1	--	0	0	x
(16,17)	1	0	0	0	--	x
(16,20)	1	0	--	0	0	x
(5,13)	0	--	1	0	1	x
(5,21)	--	0	1	0	1	x
(9,11)	0	1	0	--	1	x
(9,13)	0	1	--	0	1	x
(10,11)	0	1	0	1	--	x
(12,13)	0	1	1	0	--	x
(12,28)	--	1	1	0	0	x
(17,21)	1	0	--	0	1	x
(20,21)	1	0	1	0	--	x
(20,28)	1	--	1	0	0	x
(28,30)	1	1	1	--	0	
(30,31)	1	1	1	1	--	

Loop 2

	A	B	C	D	E	
(0,1,4,5)	0	0	--	0	--	x
(0,1,8,9)	0	--	0	0	--	x
(0,1,16,17)	--	0	0	0	--	x
(0,4,8,12)	0	--	--	0	0	x
(0,4,16,20)	--	0	--	0	0	x
(1,5,9,13)	0	--	--	0	1	x
(1,5,17,21)	--	0	--	0	1	x
(4,5,12,13)	0	--	1	0	--	x
(4,5,20,21)	--	0	1	0	--	x
(4,12,20,28)	--	--	1	0	0	
(8,9,10,11)	0	1	0	--	--	
(8,9,12,13)	0	1	--	0	--	x
(16,17,20,21)	1	0	--	0	--	x

Loop 3:

	A	B	C	D	E
(0,1,4,5,8,9,12,13)	0	--	--	0	--
(0,1,4,5,16,17,20,21)	--	0	--	0	--

Prime Implicant Table:

	0	1	4	5	8	9	11	13	17	20	21	28	30	31	10	12	16
(28,30)												X	X				
(30,31)													X	X			
(4,12,20,28)			X							X		X				X	
(8,9,10,11)					X	X	X								X		
(0,1,4,5,8,9,12,13)	X	X	X	X	X	X		X								X	
(0,1,4,5,16,17,20,21)	X	X	X	X					X	X	X						X

Select rows that contain essential prime implicants.

	0	1	4	5	8	9	11	13	17	20	21	28	30	31	10	12	16
(28,30)												X	X				
(30,31)													X	X			
(4,12,20,28)			X							X		X				X	
(8,9,10,11)					X	X	X								X		
(0,1,4,5,8,9,12,13)	X	X	X	X	X	X		X								X	
(0,1,4,5,16,17,20,21)	X	X	X	X					X	X	X						X

	A	B	C	D	E
(28,30)	1	1	1	--	0
(30,31)	1	1	1	1	--
(4,12,20,28)	--	--	1	0	0
(8,9,10,11)	0	1	0	--	--
(0,1,4,5,8,9,12,13)	0	--	--	0	--
(0,1,4,5,16,17,20,21)	--	0	--	0	--

m(28) is left to be covered by one of the two non-essential prime implicants:

$$A \cdot B \cdot C \cdot \bar{E} \text{ and } C \cdot \bar{D} \cdot \bar{E}.$$

Since we want a minimal solution, for non essential PI, we pick $C \cdot \bar{D} \cdot \bar{E}$:

$$F(A,B,C,D,E) = A \cdot B \cdot C \cdot D + C \cdot \bar{D} \cdot \bar{E} + \bar{A} \cdot B \cdot \bar{C} + \bar{A} \cdot \bar{D} + \bar{B} \cdot \bar{D}$$

$$1.2. F(A,B,C,D,E) = \sum m(0, 4, 12, 15, 27, 29, 30) + d(1, 5, 9, 10, 14, 16, 20, 28, 31)$$

Convert To Binary:

	A	B	C	D	E	
(0)	0	0	0	0	0	x
(1)	0	0	0	0	1	x
(4)	0	0	1	0	0	x
(5)	0	0	1	0	1	x
(9)	0	1	0	0	1	x
(10)	0	1	0	1	0	x
(12)	0	1	1	0	0	x
(14)	0	1	1	1	0	x
(15)	0	1	1	1	1	x
(16)	1	0	0	0	0	x
(20)	1	0	1	0	0	x
(27)	1	1	0	1	1	x
(28)	1	1	1	0	0	x
(29)	1	1	1	0	1	x
(30)	1	1	1	1	0	x
(31)	1	1	1	1	1	x

Sorted:

	A	B	C	D	E	
(0)	0	0	0	0	0	x
(1)	0	0	0	0	1	x
(4)	0	0	1	0	0	x
(16)	1	0	0	0	0	x
(5)	0	0	1	0	1	x
(9)	0	1	0	0	1	x
(10)	0	1	0	1	0	x
(12)	0	1	1	0	0	x
(20)	1	0	1	0	0	x
(14)	0	1	1	1	0	x
(28)	1	1	1	0	0	x
(15)	0	1	1	1	1	x
(27)	1	1	0	1	1	x
(29)	1	1	1	0	1	x
(30)	1	1	1	1	0	x
(31)	1	1	1	1	1	x

Loop 1:

	A	B	C	D	E	
(0,1)	0	0	0	0	--	x
(0,4)	0	0	--	0	0	x
(0,16)	--	0	0	0	0	x
(1,5)	0	0	--	0	1	x
(1,9)	0	--	0	0	1	
(4,5)	0	0	1	0	--	x
(4,12)	0	--	1	0	0	x
(4,20)	--	0	1	0	0	x
(16,20)	1	0	--	0	0	x
(10,14)	0	1	--	1	0	
(12,14)	0	1	1	--	0	x
(12,28)	--	1	1	0	0	x
(20,28)	1	--	1	0	0	x
(14,15)	0	1	1	1	--	x
(14,30)	--	1	1	1	0	x
(28,29)	1	1	1	0	--	x
(28,30)	1	1	1	--	0	x
(15,31)	--	1	1	1	1	x
(27,31)	1	1	--	1	1	
(29,31)	1	1	1	--	1	x
(30,31)	1	1	1	1	--	x

Loop 2:

	A	B	C	D	E	
(0,1,4,5)	0	0	--	0	--	
(0,4,16,20)	--	0	--	0	0	
(4,12,20,28)	--	--	1	0	0	
(12,14,28,30)	--	1	1	--	0	
(14,15,30,31)	--	1	1	1	--	
(28,29,30,31)	1	1	1	--	--	

Prime Implicant Table:

	0	4	12	15	27	29	30	1	5	9	10	14	16	20	28	31
(1,9)								X		X						
(10,14)											X	X				
(27,31)					X											X
(0,1,4,5)	X	X						X	X							
(0,4,16,20)	X	X											X	X		
(4,12,20,28)		X	X											X	X	
(12,14,28,30)			X				X					X			X	
(14,15,30,31)				X			X					X				X
(28,29,30,31)						X	X								X	X

Select rows that contain essential prime implicants.

	0	4	12	15	27	29	30	1	5	9	10	14	16	20	28	31
(1,9)								X		X						
(10,14)											X	X				
(27,31)					X											X
(0,1,4,5)	X	X						X	X							
(0,4,16,20)	X	X											X	X		
(4,12,20,28)		X	X											X	X	
(12,14,28,30)			X				X					X			X	
(14,15,30,31)				X			X					X				X
(28,29,30,31)						X	X								X	X

	A	B	C	D	E
(1,9)	0	--	0	0	1
(10,14)	0	1	--	1	0
(27,31)	1	1	--	1	1
(0,1,4,5)	0	0	--	0	--
(0,4,16,20)	--	0	--	0	0
(4,12,20,28)	--	--	1	0	0
(12,14,28,30)	--	1	1	--	0
(14,15,30,31)	--	1	1	1	--
(28,29,30,31)	1	1	1	--	--

m(0,4,12) are left to be covered by the remaining non-essential implicants:
 $\overline{A} \cdot \overline{B} \cdot \overline{D}$, $\overline{B} \cdot \overline{D} \cdot \overline{E}$, $C \cdot \overline{D} \cdot \overline{E}$, and $B \cdot C \cdot \overline{E}$

There are four possible solutions:

$$F(A,B,C,D,E) = A \cdot B \cdot D \cdot E + \overline{A} \cdot \overline{B} \cdot \overline{D} + C \cdot \overline{D} \cdot \overline{E} + B \cdot C \cdot D + A \cdot B \cdot C$$

$$F(A,B,C,D,E) = A \cdot B \cdot D \cdot E + \overline{A} \cdot \overline{B} \cdot \overline{D} + B \cdot C \cdot \overline{E} + B \cdot C \cdot D + A \cdot B \cdot C$$

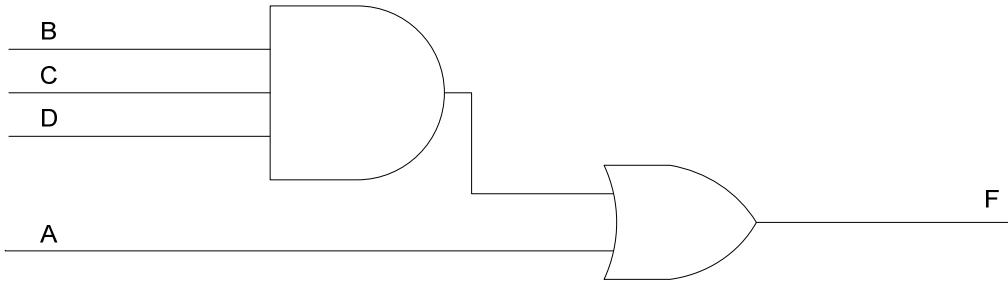
$$F(A,B,C,D,E) = A \cdot B \cdot D \cdot E + \overline{B} \cdot \overline{D} \cdot \overline{E} + C \cdot \overline{D} \cdot \overline{E} + B \cdot C \cdot D + A \cdot B \cdot C$$

$$F(A,B,C,D,E) = A \cdot B \cdot D \cdot E + \overline{B} \cdot \overline{D} \cdot \overline{E} + B \cdot C \cdot \overline{E} + B \cdot C \cdot D + A \cdot B \cdot C$$

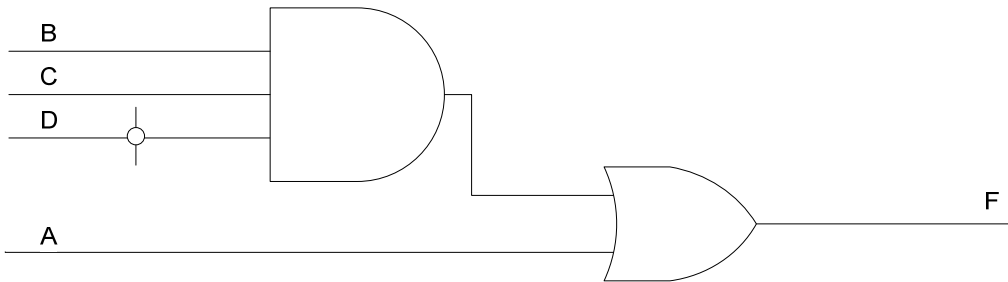
2. (10%) Implement the following Boolean Expressions using Mixed Logic. For each equation, show two implementations of the logic circuits either completely in NAND gates or completely in NOR gates. Please discuss how many NAND (or NOR) gates are needed for each logic circuit. For each gate, you are allowed to use any arbitrary number of inputs. Do not simplify the equations, leave them intact.

2.1. $F = A + B \cdot C \cdot \bar{D}$

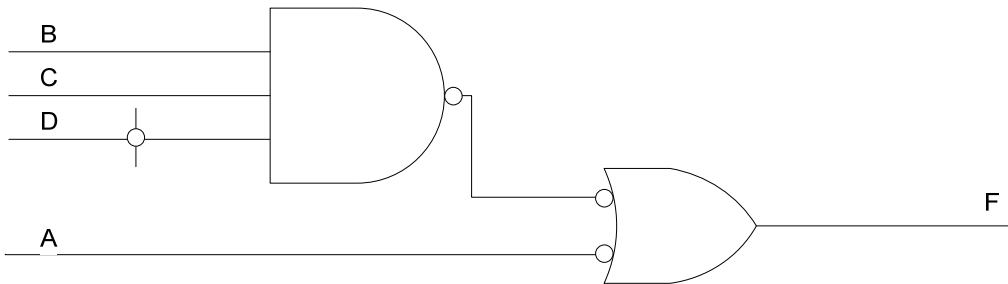
Step 1: The general form



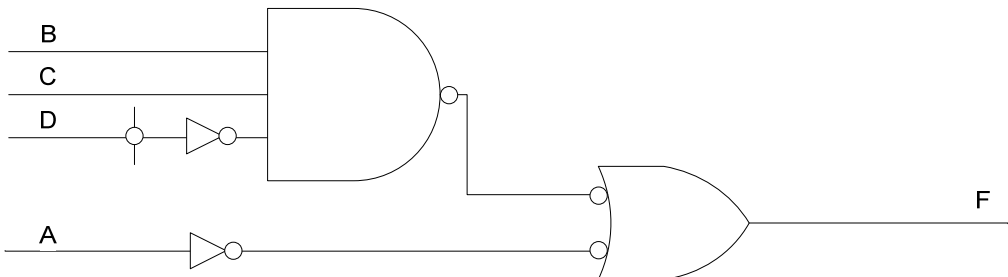
Step 2: Add vertical bars for each inversion



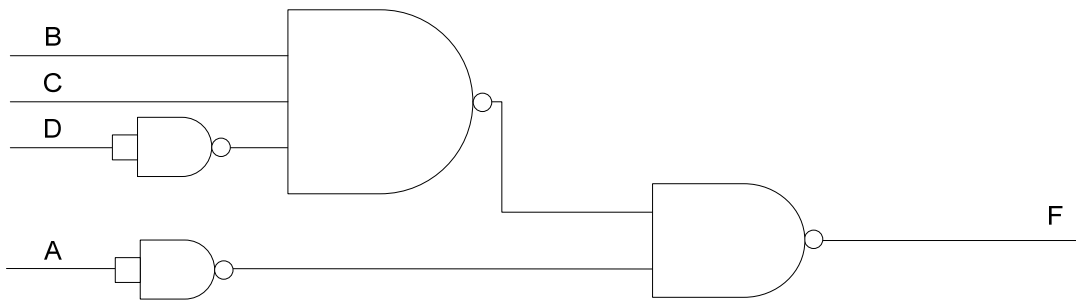
Step 3: Convert Gates to NAND gates



Step 4: Balance number of bubbles with inverters



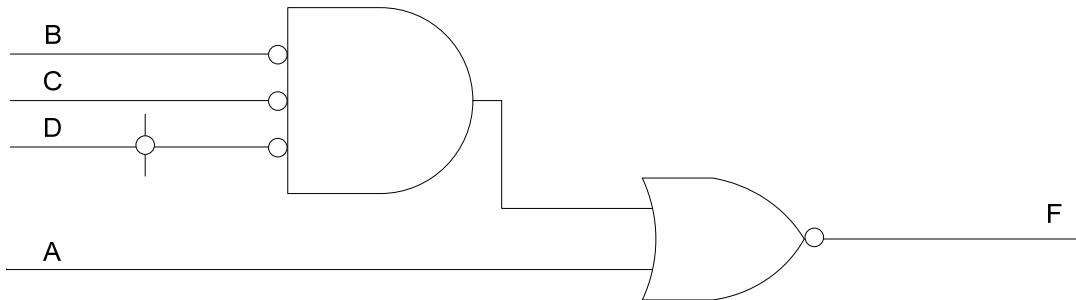
Step 5: Change to NAND representation



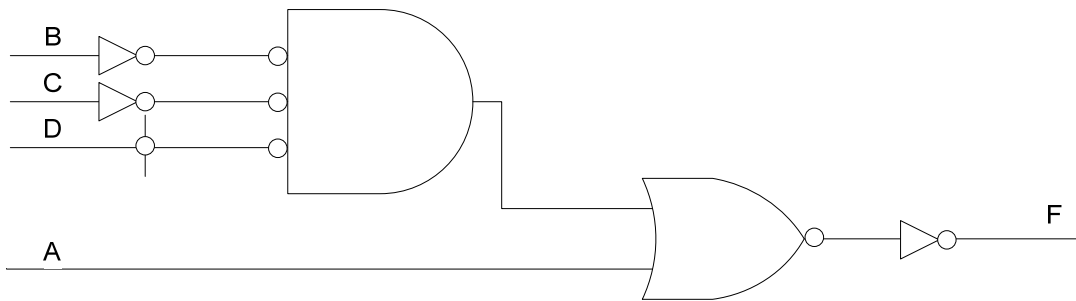
4 NAND gates used. (one has 3 inputs)

For the NOR gates restart at step 3:

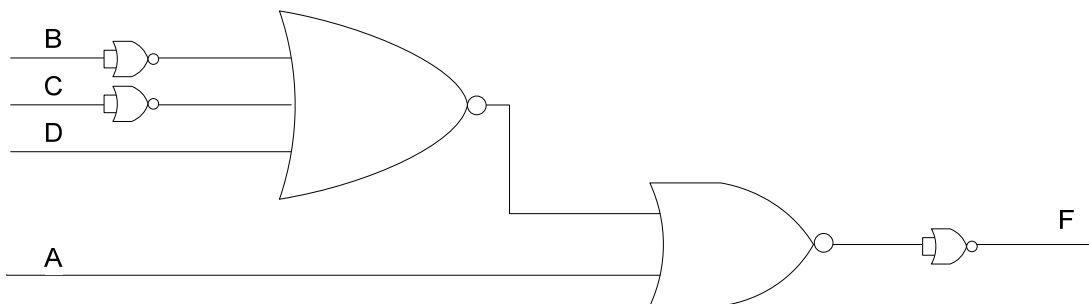
Step 3: Convert Gates to NOR gates



Step 4: Balance number of bubbles with inverters



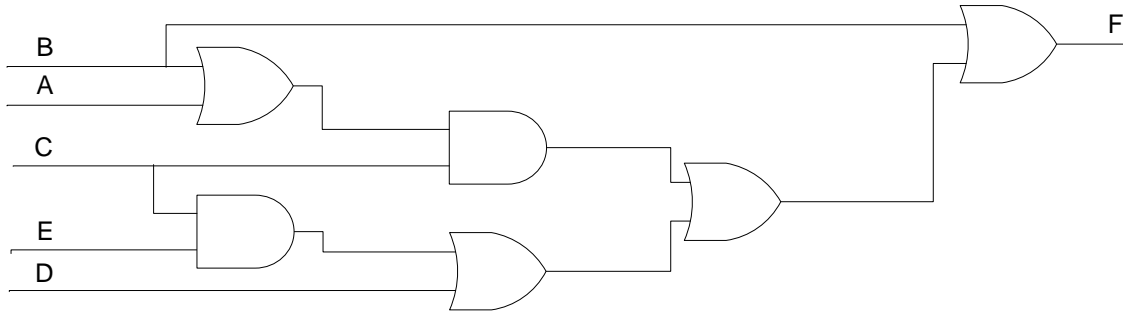
Step 5: Change to NOR representation



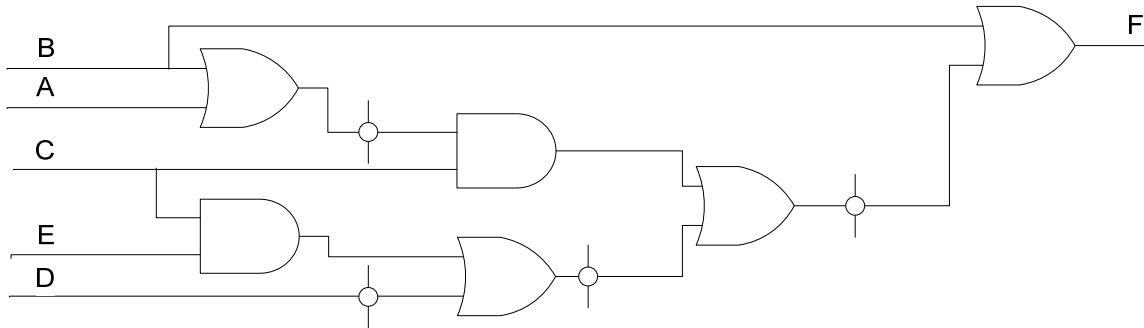
5 NOR gates used. (one has 3 inputs)

$$2.2. F = \overline{\overline{A + B \cdot C + \overline{D} + C \cdot E + B}}$$

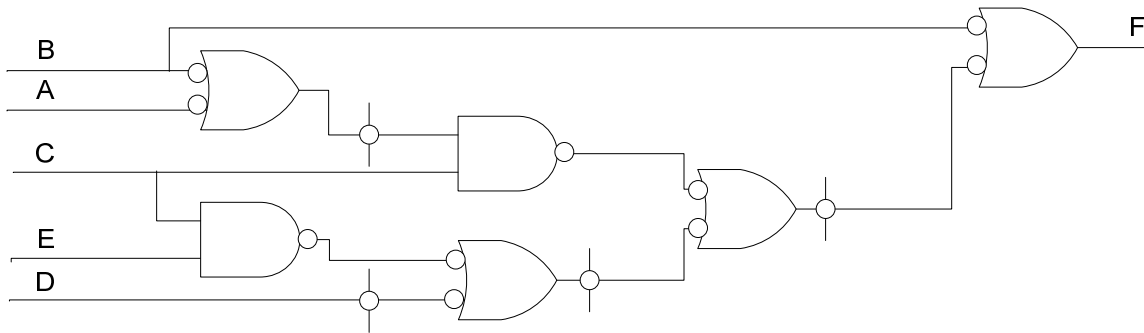
Step1: The general form



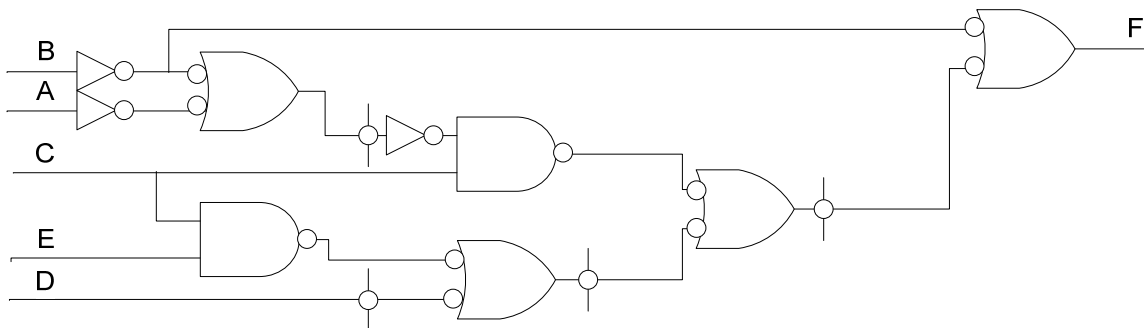
Step2: Add vertical bars for each inversion



Step 3: Convert Gates to NAND gates



Step 4: Balance number of bubbles with inverters:



3. (10%) For the given canonical POS representation below, please finish the sub-problems. You can draw your Multiplexor and Decoder using block diagram with appropriate input and output labels. In other words, you do not need to provide the design details within a MUX block or Decoder block.

$$F(A,B,C,D) = \prod M(2, 3, 6, 7, 9, 12, 14, 15)$$

Represent the function as minterms. $F(A,B,C,D) = \sum m(0,1,4,5,8,10,11,13)$

3.1. Design it using an **8-to-1 Multiplexor**

There are multiple variations depending on which variable you decided not to be part of the select lines.

For example, if D is the fourth variable, the corresponding minterms for each input lines are shown below.

Notice for input line A0, D' corresponds to m(0) and D corresponds to m(1). The desired minterms are highlighted.

If both minterms are desired, the state of the fourth term doesn't matter and the output is 1.

If neither minterm is desired, F = 0.

If only the minterm corresponding to D' is needed, F = D'.

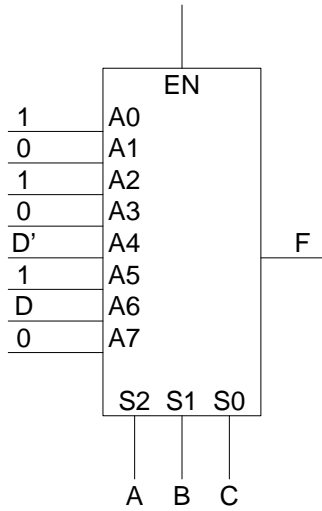
If only the minterm corresponding to D is needed, F = D.

Version 1:

(corresponding minterms)

D'	D	A	B	C	F
0	1	0	0	0	1
2	3	0	0	1	0
4	5	0	1	0	1
6	7	0	1	1	0
8	9	1	0	0	D'
10	11	1	0	1	1
12	13	1	1	0	D
14	15	1	1	1	0

Therefore, the 8-to-1 Multiplexor is:



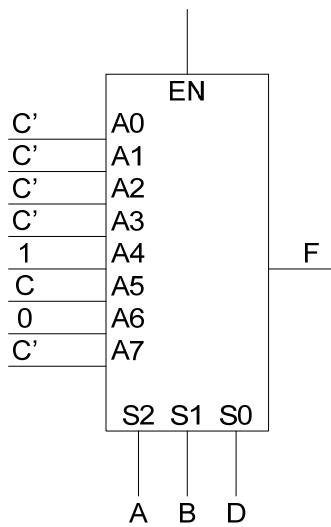
Similarly, other versions of the multiplexor are calculated.

Version 2:

(corresponding minterms)

C'	C
0	2
1	3
4	6
5	7
8	10
9	11
12	14
13	15

A	B	D	F
0	0	0	C'
0	0	1	C'
0	1	0	C'
0	1	1	C'
1	0	0	1
1	0	1	C
1	1	0	0
1	1	1	C'

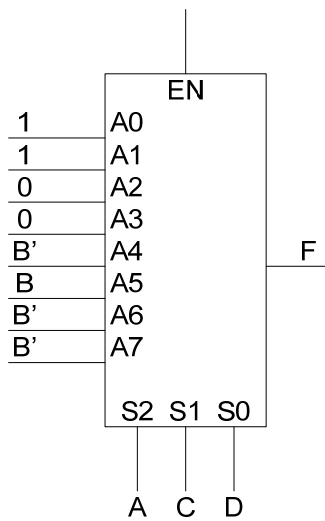


Version 3:

(corresponding minterms)

B'	B
0	4
1	5
2	6
3	7
8	12
9	13
10	14
11	15

A	C	D	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	B'
1	0	1	B
1	1	0	B'
1	1	1	B'

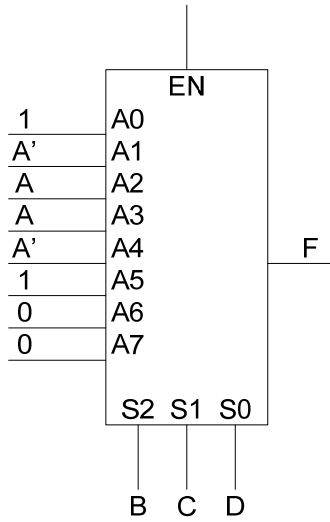


Version 4:

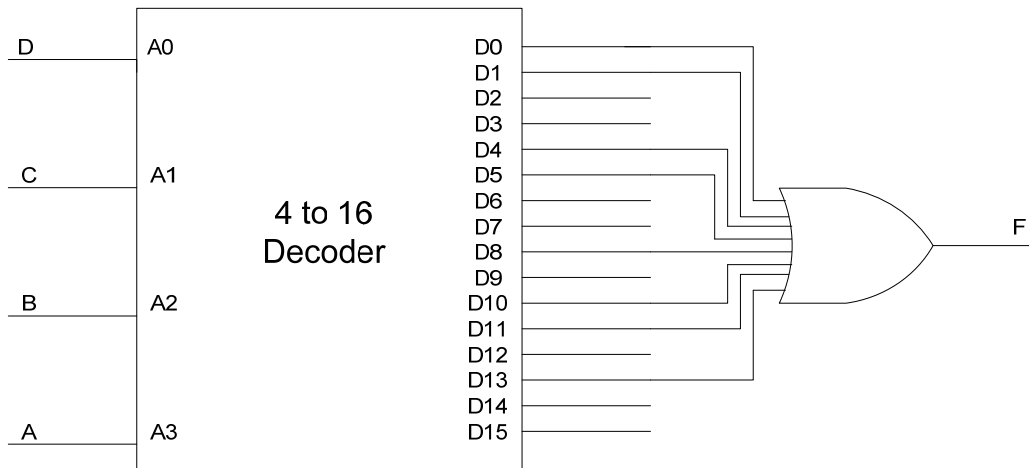
(corresponding minterms)

A'	A
0	8
1	9
2	10
3	11
4	12
5	13
6	14
7	15

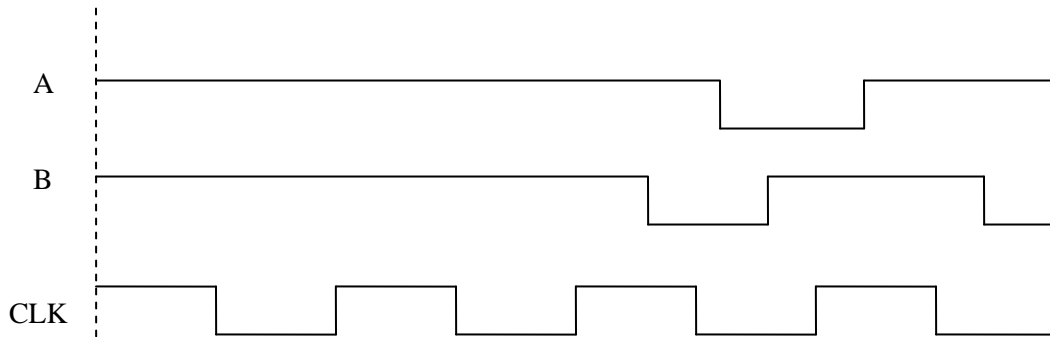
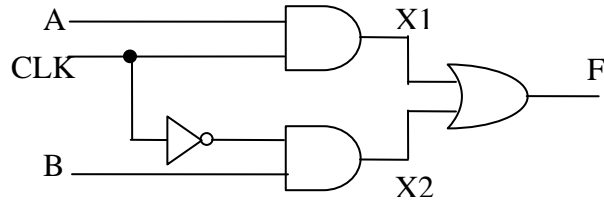
B	C	D	F
0	0	0	1
0	0	1	A'
0	1	0	A
0	1	1	A
1	0	0	A'
1	0	1	1
1	1	0	0
1	1	1	0



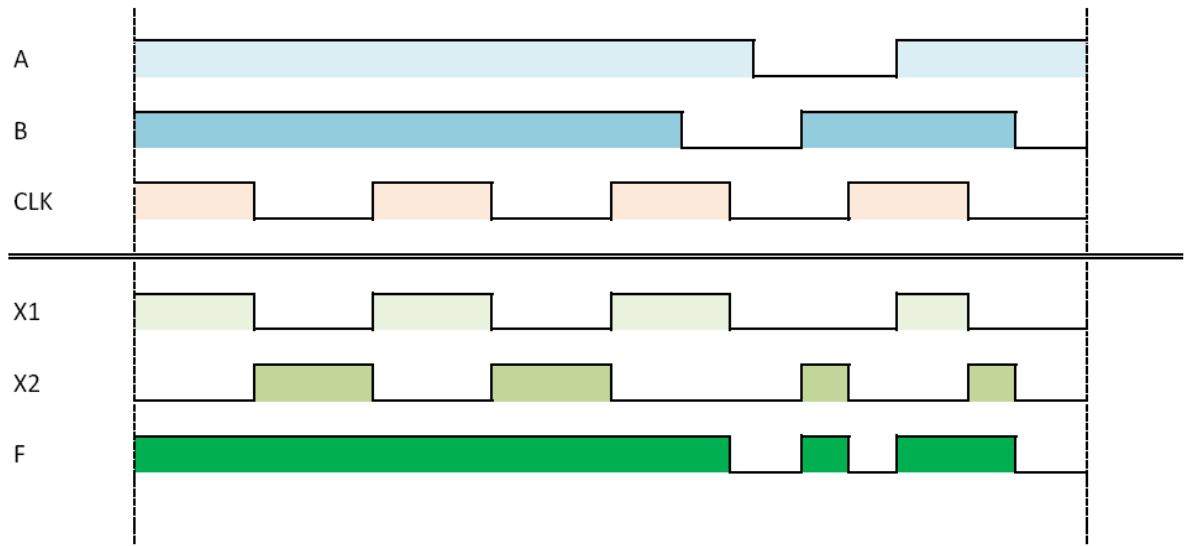
3.2. Design it using a **4-to-16 Decoder and OR gates**



4. (10%) Given the following combinational logic and input waveforms A, B, and CLK, draw the corresponding timing diagram for signal X1, X2 and F.

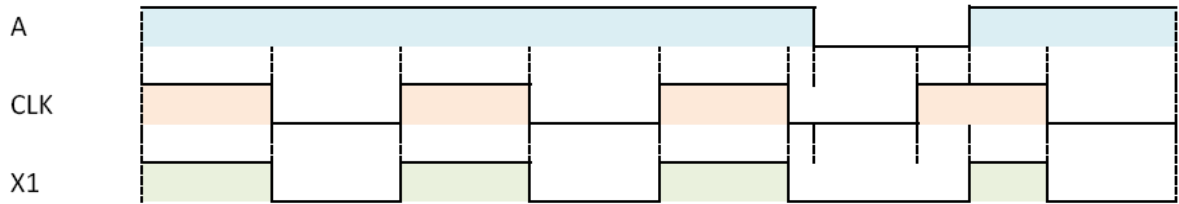


The timing diagram will be as follows:

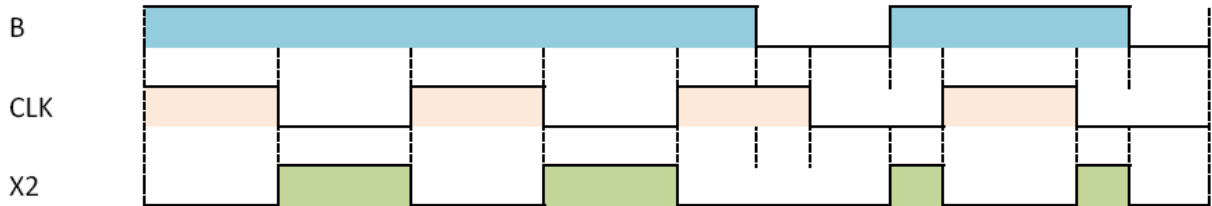


The computation of each element of the output is shown below.

$$X1 = A \cdot CLK.$$



$$X2 = B \cdot \overline{CLK}.$$



$$F = X1 + X2.$$



5. (20%) Binary Coded Decimal (or BCD) is a coding method which uses 4-bit binary to represent a decimal from 0 up to 9 as shown below. Note that, 4-bit binaries beyond 1001 are not used.

Decimal Symbol	BCD Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Figure 3(a)

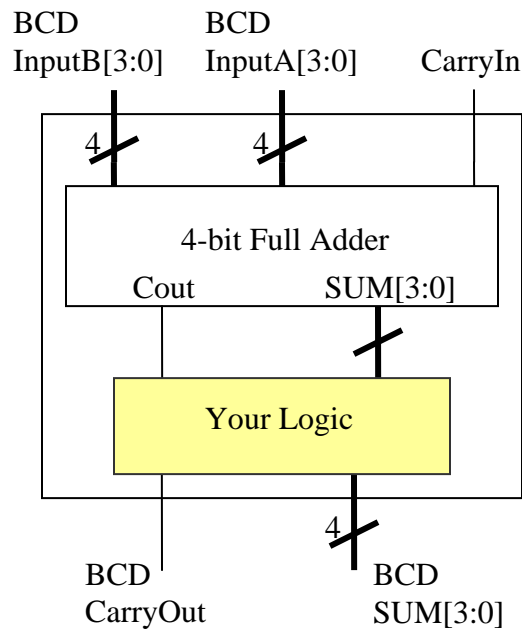


Figure 3(b)

The first intuitive step to design a functional BCD adder is to have a “carry generation logic” which generate a ‘1’ and the corresponding correct BCD SUM output when the output of the sum is equal to or greater than 10. For example, if the BCD inputs A and B are 8 (1000) and 9 (1001), then the BCD output should have a BCD CarryOut=“1” and an encoded BCD SUM[3:0] = 0111 (“7”). In contrast, the same inputs to a 4-bit Full Adder will generate a Cout=1 and SUM[3:0]=0001. Design a combinational logic, the shaded area in Figure 3(b), which can convert a Cout and SUM[3:0] from a Full Adder into the BCD style Carryout and sum. (You can use any method you like to design this, but try to minimize the logic using either K-map or Q-M method.) **HINT:** This problem has little to do with an Adder design, it is more a conventional combinational logic design problem. You have to consider the truth table of those cases when the output of a Full Adder is greater than 9.

Let F[3:0] refer to the 4-bit full adder SUM[3:0] and B[3:0] refer to the BCD SUM[3:0].
The Full-Adder and BCD representation of the values from 0 to 19 are shown below:
(Note that, there is a carry-in bit into the BCD adder, so the maximum sum will $9+9+1 = 19$).

Decimal	Cout	F[3]	F[2]	F[1]	F[0]
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1
10	0	1	0	1	0
11	0	1	0	1	1
12	0	1	1	0	0
13	0	1	1	0	1
14	0	1	1	1	0
15	0	1	1	1	1
16	1	0	0	0	0
17	1	0	0	0	1
18	1	0	0	1	0
19	1	0	0	1	1

Decimal	Carry	B[3]	B[2]	B[1]	B[0]
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1
10	1	0	0	0	0
11	1	0	0	0	1
12	1	0	0	1	0
13	1	0	0	1	1
14	1	0	1	0	0
15	1	0	1	0	1
16	1	0	1	1	0
17	1	0	1	1	1
18	1	1	0	0	0
19	1	1	0	0	1

To find the logic for each bit of the output, generate two 4x4 K-maps (one for decimal values from 0 to 15, and one from 16 to 31). (Note values greater than 19 are not valid for BCD with a carry bit (i.e. overflow error). For the sake of the K-map, treat values greater than 19 as don't care).

To find Carry Bit.

For Cout = 0;

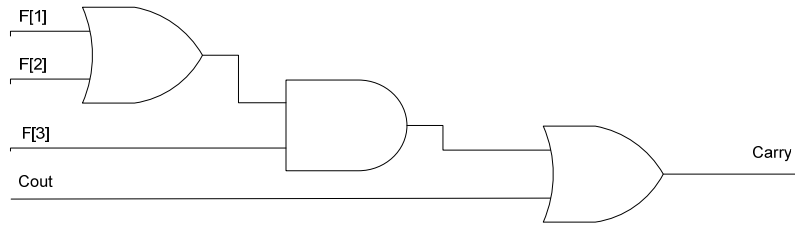
F[3]F[2]	F[1]F[0]			
	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

For Cout = 1;

F[3]F[2]	F[1]F[0]			
	00	01	11	10
00	1	1	1	1
01	X	X	X	X
11	X	X	X	X
10	X	X	X	X

$$\text{Carry} = \overline{\text{Cout}} \cdot (F[3] \cdot F[2] + F[3] \cdot F[1]) + \text{Cout}$$

$$\text{Carry} = F[3] \cdot (F[1] + F[2]) + \text{Cout}$$



To find B[0]:

From inspection, one can see that $B[0] = F[0]$.
One can also derive this relation from the K-maps.

For Cout = 0;

F[3]F[2] \ F[1]F[0]	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	1	1	0
10	0	1	1	0

For Cout = 1;

F[3]F[2] \ F[1]F[0]	00	01	11	10
00	0	1	1	0
01	X	X	X	X
11	X	X	X	X
10	X	X	X	X

$$B[0] = \overline{\text{Cout}} \cdot F[0] + \text{Cout} \cdot F[0]$$

$$B[0] = F[0]$$

F[0]

B[0]

To find B[1]:

For Cout = 0;

F[3]F[2] \ F[1]F[0]	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	1	1	0	0
10	0	0	0	0

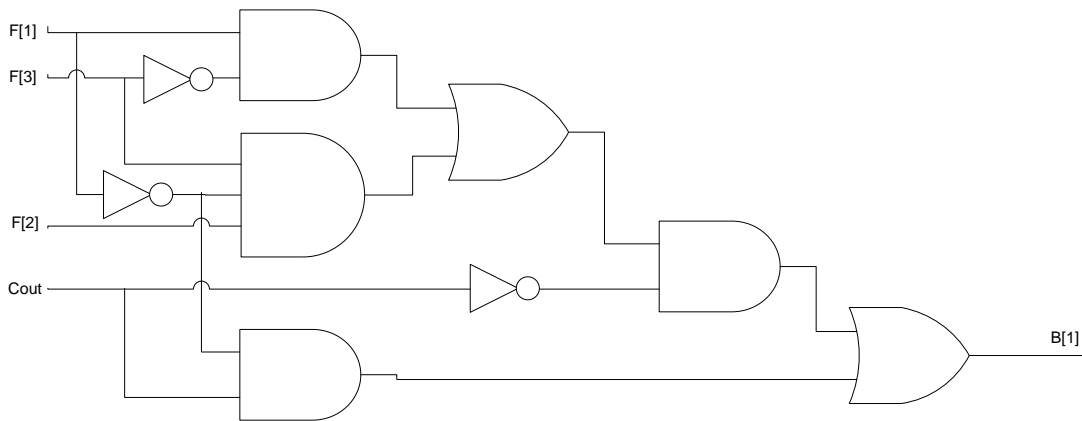
For Cout = 1;

F[3]F[2] \ F[1]F[0]	00	01	11	10
00	1	1	0	0
01	X	X	X	X
11	X	X	X	X
10	X	X	X	X

$$B[1] = \overline{\text{Cout}} \cdot (\overline{F[3]} \cdot F[1] + F[3] \cdot F[2] \cdot \overline{F[1]}) + \text{Cout} \cdot \overline{F[1]}$$

Another solution uses the carry bit. This adds a delay to the system because the carry bit must be computed first; but this can sometimes reduced the number of gates used.

$$B[1] = \text{Carry} \oplus F[1]$$



To find B[2]:

For Cout = 0;

F[3]F[2] \ F[1]F[0]	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	1	1
10	0	0	0	0

For Cout = 1;

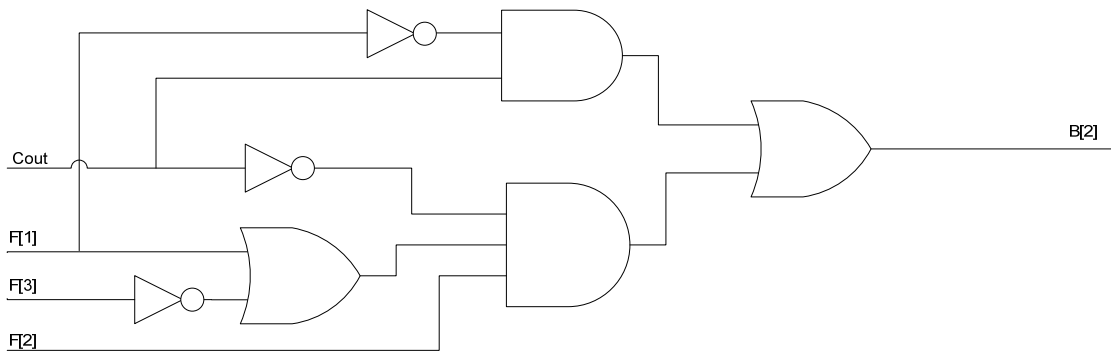
F[3]F[2] \ F[1]F[0]	00	01	11	10
00	1	1	0	0
01	X	X	X	X
11	X	X	X	X
10	X	X	X	X

$$B[2] = \overline{\text{Cout}} \cdot (\overline{F[3]} \cdot F[2] + F[2] \cdot F[1]) + \text{Cout} \cdot \overline{F[1]}$$

$$B[2] = \overline{\text{Cout}} \cdot F[2] \cdot (\overline{F[3]} + F[1]) + \text{Cout} \cdot \overline{F[1]}$$

A solution using the carry bit is:

$$B[2] = \overline{\text{Carry}} \cdot F[2] + \text{Carry} \cdot \overline{F[1] \oplus F[2]}$$



To find B[3]:

For Cout = 0;

F[3]F[2] \ F[1]F[0]	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	1	1	0	0

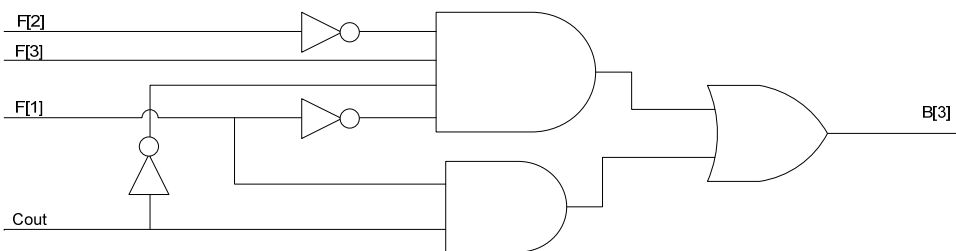
For Cout = 1;

F[3]F[2] \ F[1]F[0]	00	01	11	10
00	0	0	1	1
01	X	X	X	X
11	X	X	X	X
10	X	X	X	X

$$B[3] = \overline{\text{Cout}} \cdot F[3] \cdot \overline{F[2]} \cdot \overline{F[1]} + \text{Cout} \cdot F[1]$$

Using the carry bit:

$$B[3] = \overline{\text{Carry}} \cdot F[3] + \text{Carry} \cdot F[1]$$



6. (30%) Seven-Segment Display. A DVD player display contains 5 different modes: On (ON), Off (OF), Stop (SP), Fast-forward (FF), and Play (PL). Using 2 seven-segment displays, each mode is encoded and illustrated in Figure 6(b). There are 5 push buttons that control the entering of the 5 states. Assume that when a button is pushed, the state will stay until another button is pushed. At any given time, only one button can stay pushed (or enabled). If more than one button is pushed simultaneously, the priority is determined as specified in Figure 6(b) and only the one with the highest priority will be enabled. To display correctly, you have to design a Decoder to convert a given push to the 14 control signals (A to G, 2 bits each) that control the 2 seven-segment displays. Please show and minimize your logic design of such a decoder. You can use any basic gates.

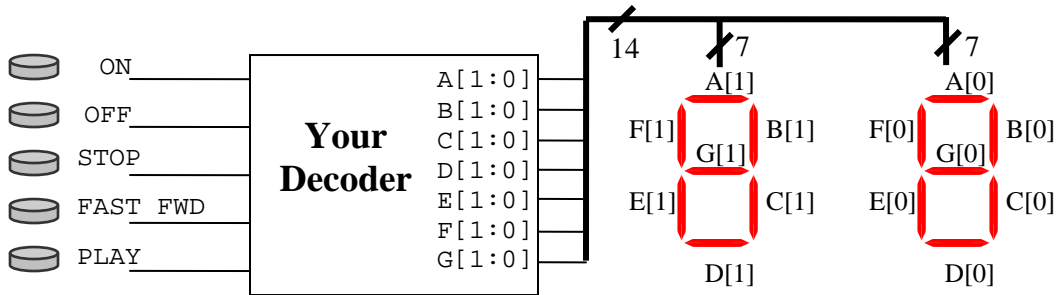


Figure 6(a)

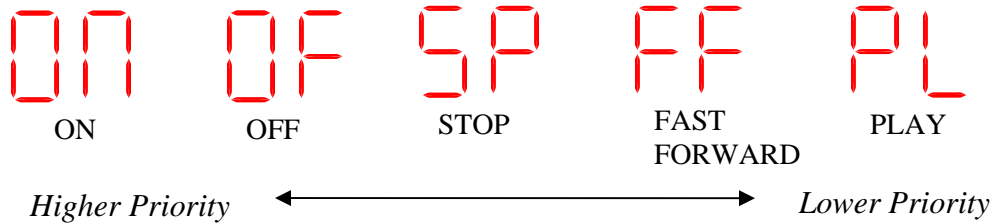


Figure 6(b)

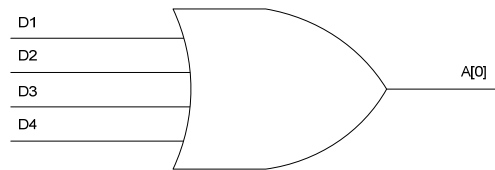
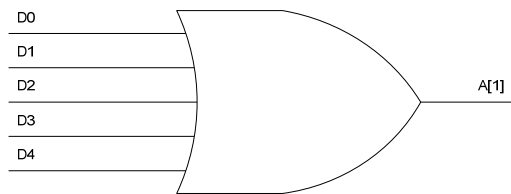
	O	N	O	F	S	P	F	F	P	L
	LD	RD	LD	RD	LD	RD	LD	RD	LD	RD
A	1	1	1	1	1	1	1	1	1	0
B	1	1	1	0	0	1	0	0	1	0
C	1	1	1	0	1	0	0	0	0	0
D	1	0	1	0	1	0	0	0	0	1
E	1	1	1	1	0	1	1	1	1	1
F	1	1	1	1	1	1	1	1	1	1
G	0	0	0	1	1	1	1	1	1	0

ON	OF	SP	FF	PL	LD (Left Display)							RD (Right Display)							
D4	D3	D2	D1	D0	A[1]	B[1]	C[1]	D[1]	E[1]	F[1]	G[1]	A[0]	B[0]	C[0]	D[0]	E[0]	F[0]	G[0]	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0	1	1	1	0	0	0	1	1	1	0	0
0	0	0	1	X	1	0	0	0	1	1	1	1	0	0	0	1	1	1	1
0	0	1	X	X	1	0	1	1	0	1	1	1	1	0	0	1	1	1	1
0	1	X	X	X	1	1	1	1	1	1	0	1	0	0	0	1	1	1	1
1	X	X	X	X	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0

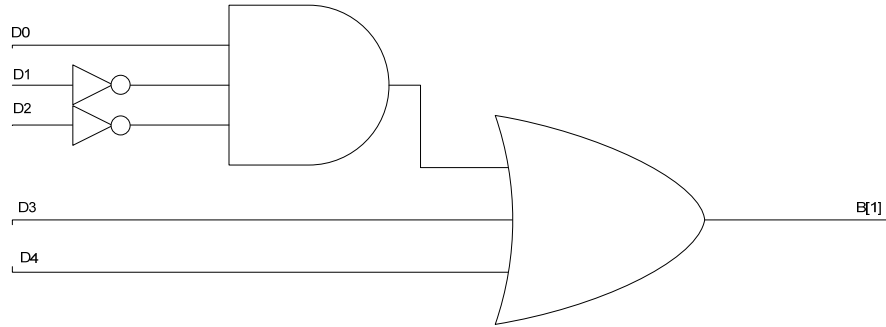
$$A[1] = D0 + D1 + D2 + D3 + D4$$

I am applying Boolean simplification $\overline{AB} + A = A + B$

$$\begin{aligned} A[0] &= \overline{D4D3D2D1} + \overline{D4D3D2} + \overline{D4D3} + D4 = \overline{D4D3}(D2D1 + D2) + \overline{D4D3} + D4 \\ &= \overline{D4D3}(D1 + D2) + \overline{D4D3} + D4 = \overline{D4}(D3(D1 + D2) + D3) + D4 = \overline{D4}(D1 + D2 + D3) + D4 \\ &= D1 + D2 + D3 + D4 \end{aligned}$$



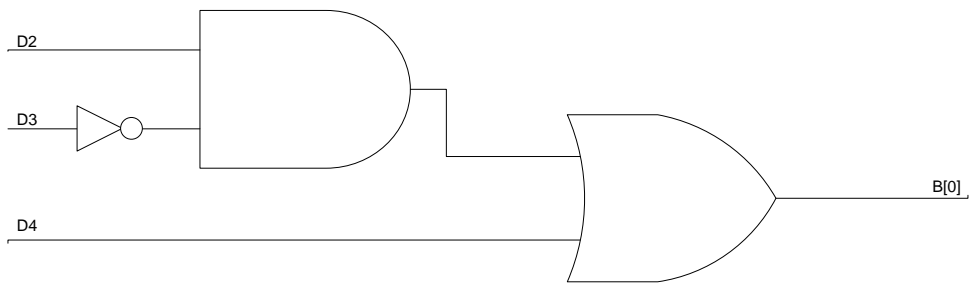
$$\begin{aligned} B[1] &= D0 \cdot \overline{D1} \cdot \overline{D2} \cdot \overline{D3} \cdot \overline{D4} + D3 \cdot \overline{D4} + D4 \\ &= D0 \cdot \overline{D1} \cdot \overline{D2} + D3 + D4 \end{aligned}$$



Use Simplification property again

$$B[0] = D2 \cdot \overline{D3} \cdot \overline{D4} + D4$$

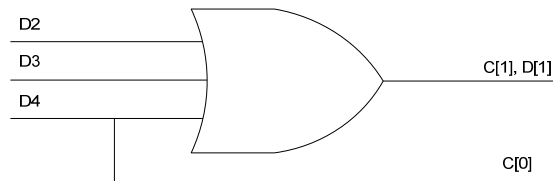
$$= D2 \cdot \overline{D3} + D4$$



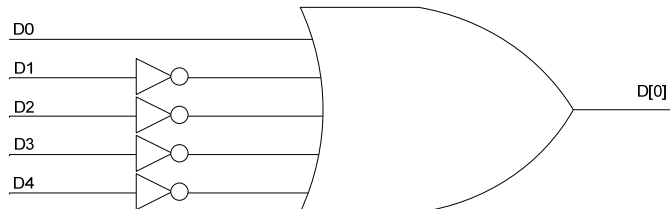
$$C[1] = D2 + D3 + D4$$

$$C[0] = D4$$

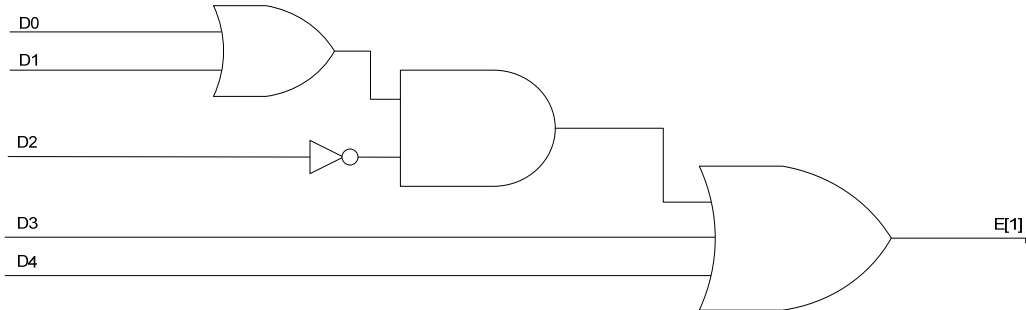
$$D[1] = C[1] = D2 + D3 + D4$$



$$D[0] = D0 \cdot \overline{D1} \cdot \overline{D2} \cdot \overline{D3} \cdot \overline{D4}$$

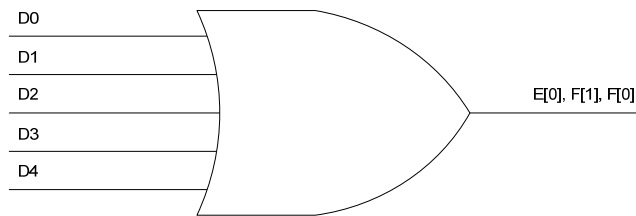


$$\begin{aligned}
 E[1] &= D0 \cdot \overline{D1} \cdot \overline{D2} \cdot \overline{D3} \cdot \overline{D4} + D1 \cdot \overline{D2} \cdot \overline{D3} \cdot \overline{D4} + D3 \cdot \overline{D4} + D4 \\
 &= D0 \cdot \overline{D1} \cdot \overline{D2} + D1 \cdot \overline{D2} + D3 + D4 \\
 &= (D0 \cdot \overline{D1} + D1) \cdot \overline{D2} + D3 + D4 = (D0 + D1) \cdot \overline{D2} + D3 + D4
 \end{aligned}$$

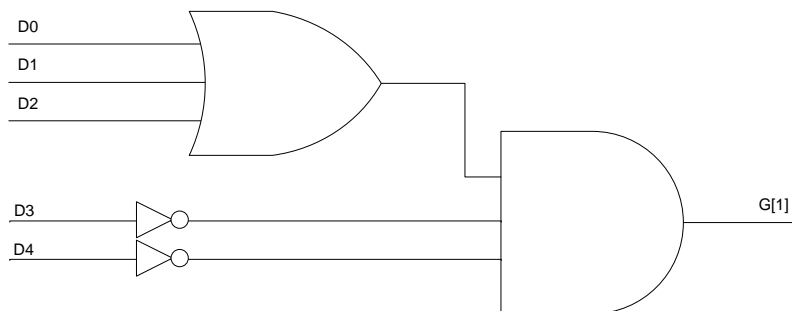


$$E[0] = D0 + D1 + D2 + D3 + D4$$

$$F[1] = F[0] = E[0] = D0 + D1 + D2 + D3 + D4$$



$$\begin{aligned}
 G[1] &= D0 \cdot \overline{D1} \cdot \overline{D2} \cdot \overline{D3} \cdot \overline{D4} + D1 \cdot \overline{D2} \cdot \overline{D3} \cdot \overline{D4} + D2 \cdot \overline{D3} \cdot \overline{D4} \\
 &= (D0 \cdot \overline{D1} \cdot \overline{D2} + D1 \cdot \overline{D2} + D2) \cdot \overline{D3} \cdot \overline{D4} \\
 &= ((D0 \cdot \overline{D1} + D1) \cdot \overline{D2} + D2) \cdot \overline{D3} \cdot \overline{D4} \\
 &= (D0 + D1 + D2) \cdot \overline{D3} \cdot \overline{D4}
 \end{aligned}$$



$$\begin{aligned}G[0] &= D1 \cdot \overline{D2} \cdot \overline{D3} \cdot \overline{D4} + D2 \cdot \overline{D3} \cdot \overline{D4} + D3 \cdot \overline{D4} \\&= (D1 \cdot \overline{D2} \cdot \overline{D3} + D2 \cdot \overline{D3} + D3) \cdot \overline{D4} \\&= ((D1 \cdot \overline{D2} + D2) \cdot \overline{D3} + D3) \cdot \overline{D4} \\&= (D1 + D2 + D3) \cdot \overline{D4}\end{aligned}$$

