B.Tech 4th Semester (AEIE) F Scheme, May 2014

Digital Electronics Paper-EE-204-F

Note : Attempt five questions. Question 1 is compulsory and one question from each of the four sections.

1. What is a logic gate? Explain each logic gate. 5x4
    a. With truth table and logic symbol.
    b. Implement the following using MUX
        i. AND gate
        ii. OR gate
    c. Explain ring counter
    d. Design the circuit of HA using ROM

SECTION-A

2. Minimise the following function using the GM method. 5x4
    a. $Y=\sum m(0,1,3,7,10,14)$
    b. Convert the followings:
        i. $(268.75)_{10}$ to binary
        ii. $(1100101011.1110)_2$ to octal
        iii. $(360)_8$ to hexadecimal
    c. Minimise the following using K-Map
        i. $Y =\sum m(4,5,10,15)$
        ii. $Y=\pi M(3,6,9,14,15)$
    d. Explain the procedure form generation matrix in binary cycle codes
3. (a) Write in detail about various error detecting and correcting codes. 15
    (b) Find the 9's and 10's complement of the following numbers 5
    (i). 25 (ii) 155 (iii) 333 (iv) 982

SECTION-B

4. (a) Design the circuit of full adder using 8:1 MUX 10
    (b) Design a full subtractor using half subtractors. 10
5. Give the truth table and logic diagram of:
    a. 3:8 decoder
    b. Implement the function $F(A,B,C) = \sum m(1,3,5,6)$ using decoder 5
    c. Design a binary to gray code converter. 10

SECTION-C

6. (a) Give the excitation table of the followings flip-flop
    (i) D FF
    (ii) JK FF
    (iii) T FF

(iv) SR FF

b. Design 0,1,2,3,0 counter using D FF
c. Explain the working of Master-Slave JK FF
d. Explain the working of Serial-in-Serial-out register

7. Design synchronour decade counter using: 10,10
   a. JK Flip-Flop
   b. Give the difference between the following
      i. Decoder and De-Multiplexer
      ii. Ripple counter and Synchronous counter
      iii. Latch and Flip-Flop

SECTION-D

8. (i) Design the circuit of Half Adder using PLA 10
   (ii) Design BCD to XS-3 code converter using PLA 10

9. (a) Realise the following function using ROM 10
      i. $F=\sum m(0,1,2,3)$
      ii. $F=\sum m(0,2,5)$
   b. With the help of state table and state diagram design a Mod-4 up/down counter. 10

SOLUTION

B.Tech 4th Semester (AEIE) F Scheme, May 2014

Digital Electronics Paper-EE-204-F

Note : Attempt five questions. Question 1 is compulsory and one question from each of the four sections.

1. What is a logic gate? Explain each logic gate. 5x4

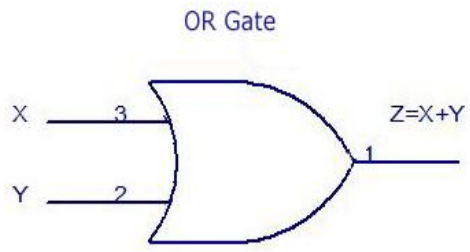
a. With truth table and logic symbol.

Solution:

Logic gates are the basic building blocks of any digital system. It is an electronic circuit having one or more than one input and only one output. The relationship between the input and the output is based on a certain **logic.** Based on this, logic gates are named as AND gate, OR gate, NOT gate etc.

THE OR GATE

The OR gate produces a HIGH output when any or all of the inputs is HIGH. Figure shows the symbol for an OR and its truth table. The operation function sign for the OR gate is +

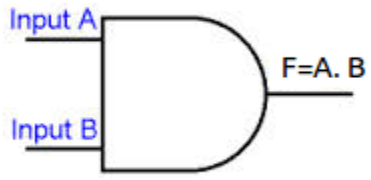


TRUTH TABLE

INPUTS		OUTPUT
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

THE AND GATE

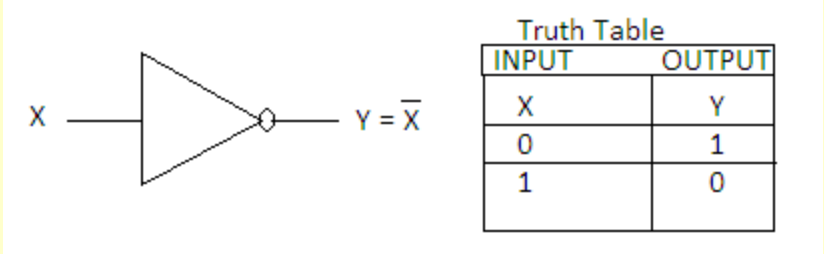
The AND gate produces a HIGH output when all of the inputs are HIGH. The AND operation is denoted by a dor (.)..AND gate is shown in figure below along with the associated Truth Table.



INPUTS		OUTPUT
A	B	F = A . B
0	0	0
0	1	0
1	0	0
1	1	1

NOT GATE

An inverter or NOT gate produces the complement of the input i.e. if the input if '0' the output produces is '1' and vice versa. The inverter symbol and truth table is shown below:



Truth Table

INPUT	OUTPUT
X	Y
0	1
1	0

b. Implement the following using MUX

i. AND gate
ii. OR gate

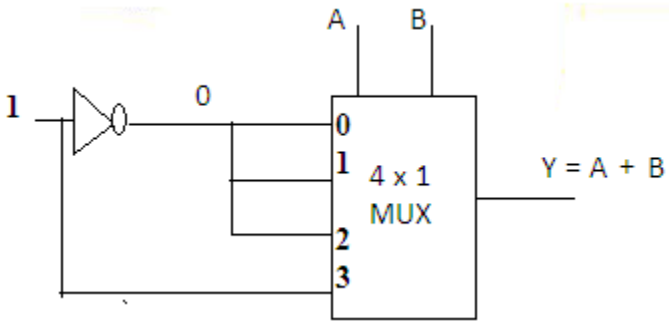
AND gate

The truth table of AND gate is shown below:

A	B	Y = A . B
0	0	0
0	1	0
1	0	0
1	1	1

SOP eqn for AND gate function Y = Y = A'B' + A'B + AB' + AB

The function Y=A.B can be implemented by using a 4 x 1 MUX with the select lines as AB and multiplexer output Y giving the output Y=A.B



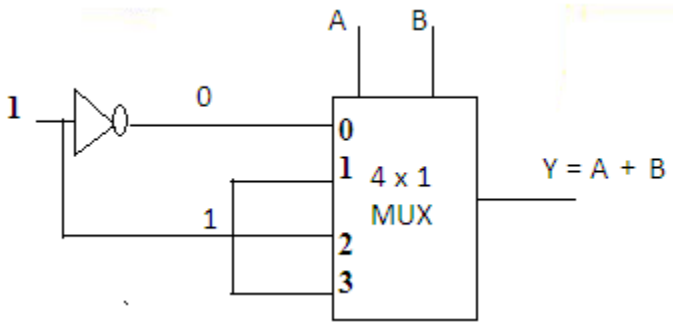
OR gate implementation

The truth table of OR gate is shown below:

A	B	Y = A . B
0	0	0
0	1	1
1	0	1
1	1	1

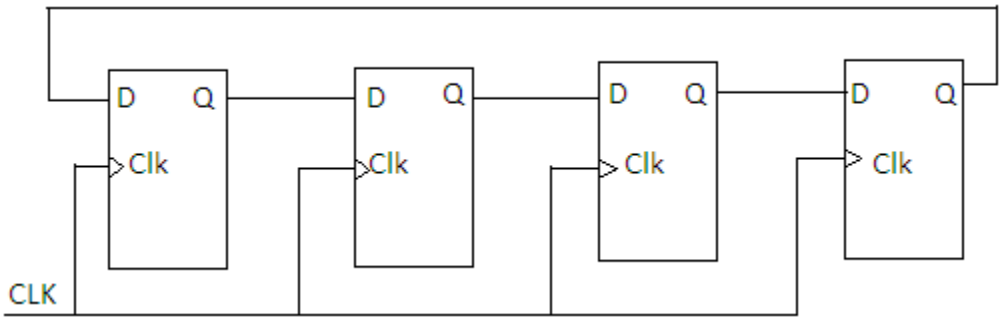
SOP eqn for Y = A'B' + A'B + AB' + AB

The function Y=A + B can be implemented by using a 4 x 1 MUX with the select lines as AB and multiplexer output Y giving the output Y=A + B



c. Explain ring counter

A ring counter is constructed using the serial-in serial-out register whose output is fed back as input. Thus the bits from one stage to another is shifted with the arrival of every clock pulse. A 4-bit ring counter is shown in the following figure.



The ring counter output is shown in the following table

CLK	Q1	Q2	Q3	Q4
	1	0	0	0
↑	0	1	0	0
↑	0	0	1	0
↑	0	0	0	1
↑	1	0	0	0

d. Design the circuit of HA using ROM

A	B	Sum	Cy

0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The SP equations for the Sum and Carry (Cy) are given below:

Sum = A'.B + A.B'

Cy = A.B

The output are stored at the corresponding input combinations which for a ROM act as the address lines. The complete implementation is shown below:

SECTION-A

2. Minimise the following function using the GM method. 5x4
   a. Y=∑m(0,1,3,7,10,14)

Minterm	M0	M1	M3	M7	M10	M14
Binary	0000	0001	0011	0111	1010	1110
No. of 1's	0	1	2	3	2	3

Rewriting the minterms by grouping them as per no. of 1's contained in their binary value. Then comparing and cheking every mineterm from one group with adjacent group for one bit change.

No of '1's	Minterm	Binary	checked	Pair	
0	m0	0000	√	m0,m1	000-
1	m1	0001	√	m1,m3	00-1
2	m3	0011	√	m3,m7	0-11
	m10	1010	√	m10,m14	1-10
3	m7	0111	√		
	m14	1110	√		

We see that after first level of pairing no further grouping are possible. So the final equation uses all the four minterms on the right column.

Y= A'B'C' + A'B'D + A'CD + ACD'

Final we draw the PI chart as shown below:

	m0	m1	m3	m7	m10	m14
A'B'C' **(EPI)**	x	x				
A'B'D		x	x			
A'CD **(EPI)**			x	x		
ACD' **(EPI)**					x	x
	✓			✓	✓	✓

The final solution contain only the three essential prime implecants.

Y = A'B'C' + A'CD + ACD'

b. Convert the followings:

i. $(268.75)_{10}$ to binary

ii. $(1100101011.1110)_2$ to octal

iii. $(360)_8$ to hexadecimal

Solution:

i. 268.75 =

	268	
2	134	0
2	67	0
2	33	1
2	16	1
2	8	0
2	4	0
2	2	0
2	1	0
2	0	1

0.75 =

0.75 x 2 = 1 .5
0.5 x 2 = 1 .0

Therefor $(268.75)_{10} = (100001100.11)_2$

ii. $(1100101011.1110)_2$ to octal

Binary to Octal is quite easy, we can take group of three bits from right (LSB) and write the equivalent octal of binary combination.

$(1100101011.1110)_2 = (1453.70)_8$

iii. $(360)_8$ to hexadecimal

Octal to hex is also easy. First convert Octal to binary by writing 3-bit binay code of each octal digit.

$(360)_8 = (011110000)_2$

Now group the bits in 4-bit each group and write the equivalent hex digit.

So the answer is: $(011110000)_2$ = $(0F0)_{16}$

c. Minimise the following using K-Map

i. Y =∑m(4,5,10,15)

ii. Y=πM(3,6,9,14,15)

i. Y= ∑m(4,5,10,15)

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	0	0	1	0
10	0	0	0	1

Solution (i)

Y= A'BC' + ABCD +AB'CD'

ii. Y= ∏M(3,6,9,14,15)

AB \ CD	C+D	C+D'	C'+D'	C'+D
A+B	1	1	0	1
A+B'	1	1	1	0
A'+B'	1	1	0	0
A'+B	1	0	1	1

Solution (ii)

Y=(A+B+C'+D')(A'+B'+C')(B'+C'+D)(A'+B''C+D')

d. Explain the procedure form generation matrix in binary cycle codes

A code *C* is cyclic if

(i) *C* is a linear code;

(ii) any cyclic shift of a codeword is also a codeword, i.e. whenever $a_0,\ldots a_{n-1} \in C$, then also $a_{n-1}a_0 \ldots a_{n-2} \in C$.

Theorem Suppose *C* is a cyclic code of codewords of length *n* with the generator polynomial$g(x) = g_0 + g_1x + \ldots + g_rx^r$.

Then *dim* (*C*) = n - r and a generator matrix G_1 for *C* is

$$G_1 = \begin{pmatrix} g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & 0 & \ldots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \ldots & g_r & 0 & \ldots & 0 \\ .. & .. & & & & & & & & .. \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & g_0 & \ldots & g_r \end{pmatrix}$$

Proof

(i) All rows of G_1 are linearly independent.

(ii) The $n - r$ rows of G represent codewords

$$g(x),\ xg(x),\ x^2g(x),\ldots,\ x^{n-r-1}g(x) \qquad (*)$$

(iii) It remains to show that every codeword in C can be expressed as a linear combination of vectors from (*).

Indeed, if a(x) Î C, then

$a(x) = q(x)g(x)$.

Since *deg* $a(x) < n$ we have *deg* $q(x) < n - r$.

Hence

$$q(x)g(x) = (q_0 + q_1x + \ldots + q_{n-r-1}x^{n-r-1})g(x)$$

$$= q_0g(x) + q_1xg(x) + \ldots + q_{n-r-1}x^{n-r-1}g(x).$$

Example:

The task is to determine all ternary codes of length 4 and generators for them.

Factorization of $x^4 - 1$ over $GF(3)$ has the form

$x^4 - 1 = (x - 1)(x^3 + x^2 + x + 1) = (x - 1)(x + 1)(x^2 + 1)$

Therefore there are $2^3 = 8$ divisors of $x^4 - 1$ and each generates a cyclic code.

Generator polynomial	Generator matrix
1	I_4
x	$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$
$x + 1$	$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$
$x^2 + 1$	$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$
$(x - 1)(x + 1) = x^2 - 1$	$\begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$
$(x - 1)(x^2 + 1) = x^3 - x^2 + x - 1$	[-1 1 -1 1]
$(x + 1)(x^2 + 1)$	[1 1 1 1]
$x^4 - 1 = 0$	[0 0 0 0]

3. (a) Write in detail about various error detecting and correcting codes. 15

Different Error detecting and correcting codes are described below:

i. **Parity Method**

ii. **Longitudinal Redundancy Check (LRC)**

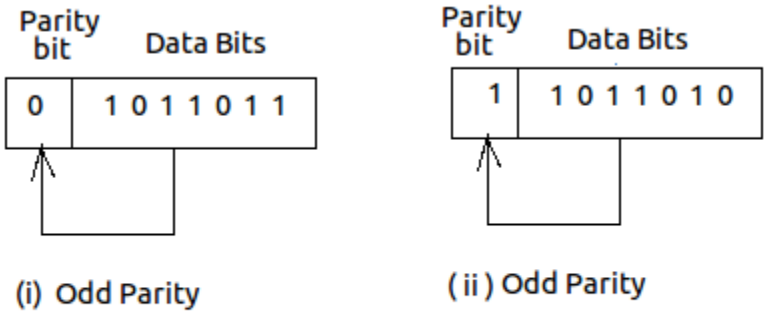
iii. **Hamming Code**

iv. **Cyclic Redundancy Check (CRC)**

Parity Check:

Parity system is a good method of error detecting codes. It is a simple method of finding 1-bit error in transmitted code. In this an additional bit is appended with the code at the transmitting end such that the number of 1's become odd or even and according parity named as odd parity or even parity. At the receiving end the same parity as sent from the transmitting end is again checked. Only problem with this code is that if more than 1-bit change occur during transmission then this is not suitable method.

Example of odd parity



(i) Odd Parity (ii) Odd Parity

Longitudinal Redundancy Check (LRC):

In this system a two dimensional parity is generated for a block of codes. HRC horizontal parity is appended with every code word and also vertical parity is appended with every column for a group of codes. Then this whole block is transmitted. At the receiving end the block is checked for any change in horizontal and vertical parity. Coordinates of the bit in error gives the exact code in which error occurred. As an example, if the characters RAVI is to be sent whose ASCII codes are

Code	ASCII	Binary	With Odd Parity
R	52h	1010010	01010010
A	41h	1000001	11000001
V	56h	1010110	11010110
I	49h	1001001	01001001
		Vertical Parity ->	11110011

Now suppose this code is transmitted and the received code with 1-bit error is given below.

The error can be easily found by checking the horizontal parity of each code and also vertical parity. The location of error is wrong parity in row and column.

With Odd Parity	

0 1 0 1 0 0 1 0	√
1 1 0 0 0 **1** 0 1	**X**
1 1 0 1 0 1 1 0	√
0 1 0 0 1 0 0 1	√
1 1 1 1 0 0 1 1	
√ √ √ √ √ **X** √ √	

So we find that the error is in row 2 and column 3 from right(bit shown in bold). It can be corrected by simply complementing this bit.

Hamming Code:

Hamming code is used to detect error in RAMs. In this code k parity bits are added to form the n-bit data word, forming a new code word of n+k bit. Those positions as power of 2 are reserved for parity bits and the remaining bits are the data word. For example, consider a code word 11001101 to be transmitted. It is first appended with parity bits at positions at power of 2 i.e. at positions 1,2,4,8,…. As shown below:

12	11	10	9	8	7	6	5	4	3	2	1
				P8				P4		P2	P1
1	1	0	0		1	1	0		1		

The parity bits are then calculated as:

P1: bits 11+9+7+5+3+1 = 1+0+1+0+1 =1 for even parity

P2:bits 11+10+7+6+3+2= 1+0+1+1+1= 0 for even parity

P4: bits 12+7+6+5+4 = 1+1+1+0 = 1 for even parity

P8: bits 12+11+10+9+8 = 1+1+0+0 = 0 for even parity

So the code becomes as :

12	11	10	9	8	7	6	5	4	3	2	1
				P8				P4		P2	P1
1	1	0	0	0	1	1	0	1	1	0	1

The code to be transmitted is:

110001101101

Cyclic Redundancy Check (CRC):

In this code the polynomial (M(x)) is appended with the as many zeros as the degree of the divisor polynomial(P(x)) (step-1). The algorithm at the sending and the receiving end is given below:

- Sending
    1. Multiply M(x) by x^n
    2. Divide x^nM(x) by P(x)
    3. Ignore the quotient and keep the reminder C(x)
    4. Form and send F(x) = x^nM(x)+C(x)
- Receiving

1. Receive F'(x)
2. Divide F'(x) by P(x)
3. Accept if remainder is 0, reject otherwise

(b) Find the 9's and 10's complement of the following numbers 5
(i). 25 (ii) 155 (iii) 333 (iv) 982

9's and 10's complement are the signed representation is decimal system (BCD). We append 0 to represent +ve number; and 9 to represent –ve number.
The 9's complement of a number is found by $((10^n - 1) - N)$ where n is the number of digits in the number N

	Decimal Number	9's Complement	10's Complement = 9's Complment +1
I	25 = 025	$((10^n - 1) - N)$ = $((10^3 - 1) - 025)$ =999-025 = 975 Here MSD('9') indicate –ve number	$((10^n - 1) - N) + 1$ = = 975 +1 = 976 Here MSD('9') indicate –ve number
Ii	155= 0155	$((10^n - 1) - N)$ = $((10^4 - 1) - 0155)$ =9999-0155 = 9844 Here MSD('9') indicate –ve number	$((10^n - 1) - N) + 1$ = =9844+1 = 9845 Here MSD('9') indicate –ve number
Iii	333 = 0333	$((10^n - 1) - N)$ = $((10^4 - 1) - 0333)$ =9999-0333 = 9666 Here MSD('9') indicate –ve number	$((10^n - 1) - N) + 1$ = = 9666 +1 = 9667 Here MSD('9') indicate –ve number
iv	982 = 0982	$((10^n - 1) - N)$ = $((10^4 - 1) - 0982)$ =999-0982 = 9017 Here MSD('9') indicate –ve number	$((10^n - 1) - N) + 1$ = = 9017 +1 = 9018 Here MSD('9') indicate –ve number

SECTION-B

4. (a) Design the circuit of full adder using 8:1 MUX 10

Solution:

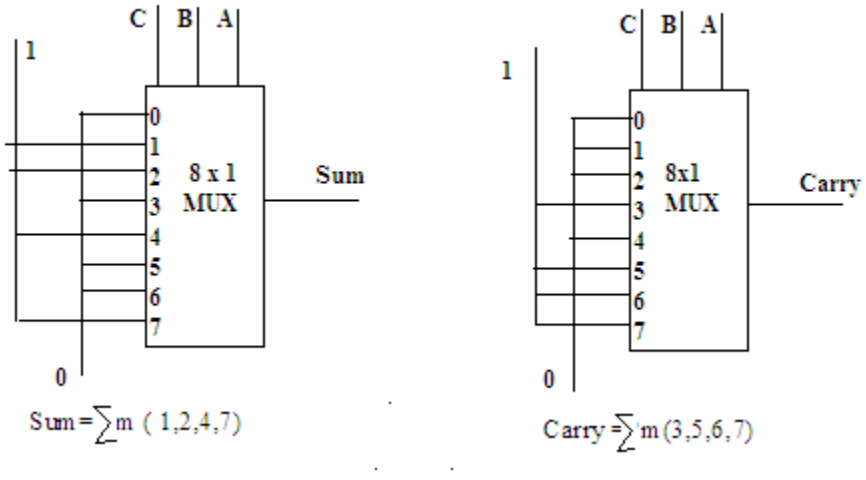
The truth table for the full adder is shown below:

A	B	C	Sum	Carry
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

Write the SOP equations

Sum= ∑m (1,2,4,7)

Carry =∑m(3,5,6,7)



(b) Design a full subtractor using half subtractors. 10

The truth table for full subtractor is shown below:

A	B	C	Diff	Borrow
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

The SOP equations for the Diff and Borrow is given below:

Diff = $\sum m(1,2,4,7) = A'B'C + A'BC' + AB'C' + ABC$

Borrow = $\sum m(1,2,3,7) = A'B'C + A'BC' + A'BC + ABC$

Solving the above equations algebraically we get:

Diff = $A'B'C + A'BC' + AB'C' + ABC$

$= A'(B'C + BC') + A(B'C' + BC)$

$= A \text{ xor } B \text{ xor } C$

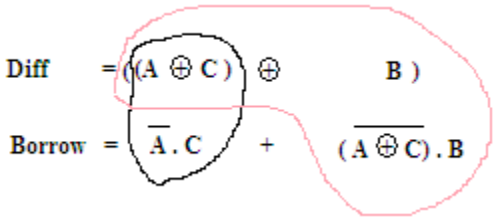
Borrow = $A'B'C + A'BC' + A'BC + ABC$

$= (A'B'C + A'BC) + (A'BC' + ABC)$

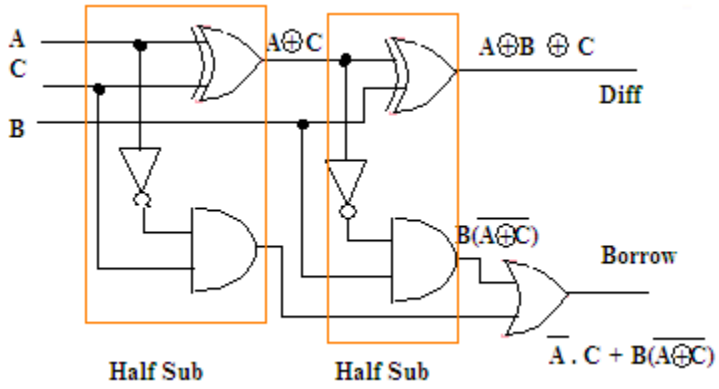
$= A'C(B' + B) \cdot C + B(A'C' + AC)$

$= A'C + (A \text{ xnor } C).B$

Looking at the simplified equation again:



We find that there are two half subtractors hidden in these two equations and have been circled clearly. We can now construct the **full subtractor** using two Half Subtractor.



5. Give the truth table and logic diagram of:

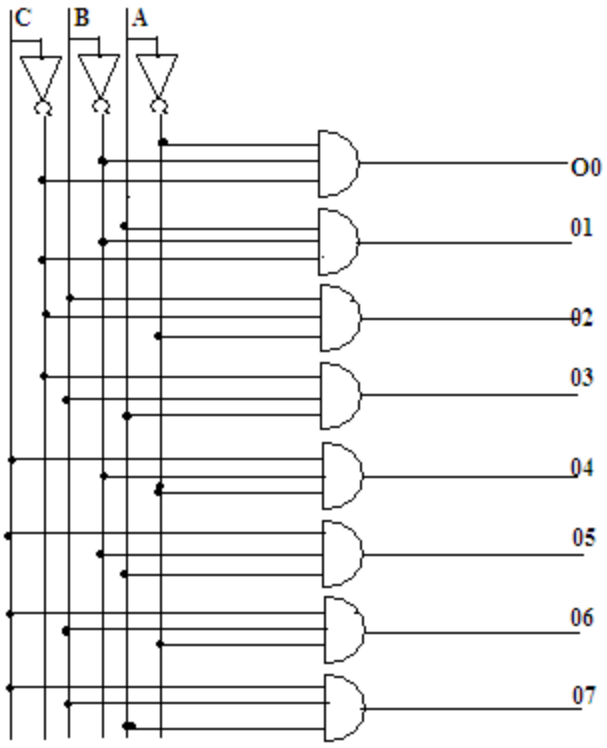
a. 3:8 decoder **5**

A decoder is a circuit that changes a code into a set of signals. It is called a decoder because it does the reverse of encoding.A common type of decoder is the line decoder which takes an n-digit binary number and decodes it into 2^N output lines. The truth table of a 3:8 decoder is shown below:

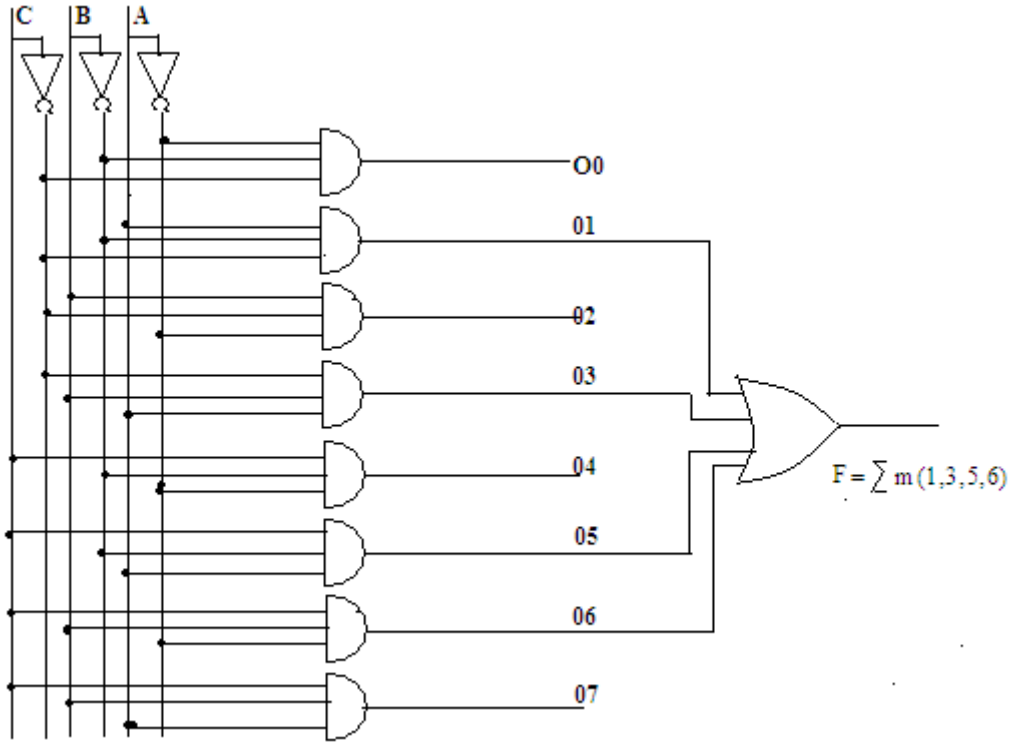
C	B	A	O0	O1	O2	O3	O4	O5	O6	O7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

There are 8 output and hence 8 independent minterms.

These can be implemented by a simple AND gate array connected to various input combinations as shown in the figure.



b. Implement the function F(A,B,C) = ∑m(1,3,5,6) using decoder **5**



c. Design a binary to gray code converter. **10**

Truth Table for Binary to Gray code converter

Decimal	Binary Input			Gray Output		
Decimal	B2	B1	B0	G2	G1	G0
0	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	1	0
5	1	0	1	1	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	0

Logical Equations:

G2 = B2B1'B0'+B2B1'B1 + B2B1B0' + B2B1B0

G1 = B2'B1B0' + B2'B1B0 + B2B1'B0' + B2B1'B0

G0 = B2'B1'B0 + B2'B1B0' + B2B1'B0 +B2B1B0'

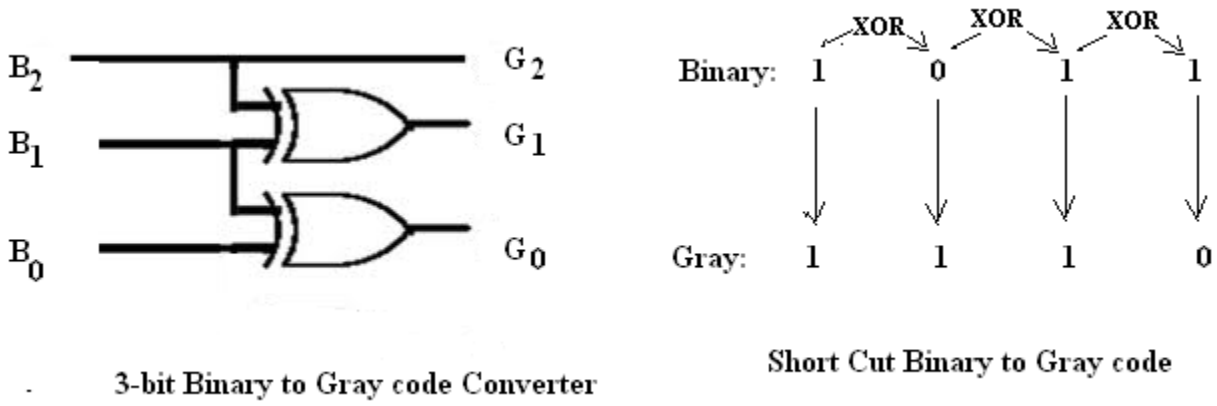
Simplification:

G2 = B2B1'(B0' + B0) + B2B1(B0' + B0)
= B2B1' + B2B1
= B2(B1' +B1)
= B2

G1 = B2'B1B0' + B2'B1B0 + B2B1'B0' + B2B1'B0
= B2'B1(B0' + B0) + B2B1'(B0' + B0)
= B2'B1 + B2B1'
= B2 $\oplus$ B1

G0 = B2'B1'B0 + B2'B1B0' + B2B1'B0 + B2B1B0'
= B2'(B1'B0 + B1B0') + B2(B1'B0 + B1B0')
=(B1'B0 + B1B0') (B2' + B2)
= B1 $\oplus$ B0

Logic Design of a 3-bit binary to gray code converter



3-bit Binary to Gray code Converter

Short Cut Binary to Gray code

Figure-: A 3-bit Binary to Gray Code Converter

SECTION-C

6. (a) Give the excitation table of the followings flip-flop

I. D FF
II. JK FF
III. T FF
IV. SR FF

Excitation Tables: an excitation table shows the minimum inputs that are necessary to generate a particular next state (in other words, to "excite" it to the next state) when the current state is known. They are similar to truth tables and state tables, but rearrange the data so that the current state and next state are next to each other on the left-hand side of the table, and the inputs needed to make that state change happen are shown on the right side of the table. The various excitation tables are shown below:

(i) Excitation Table D FF

PS	NS	D
0	0	0
0	1	1
1	0	0
1	1	1

(ii) Excitation Table JK FF

PS	NS	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

(iii) Excitation Table T FF

PS	NS	T
0	0	0
0	1	1
1	0	1
1	1	0

(iv) Excitation Table SR FF

PS	NS	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

b. Design 0,1,2,3,0 counter using D FF

Answer is being prepared

c. Explain the working of Master-Slave JK FF

Answer is being prepared

d. Explain the working of Serial-in-Serial-out register

Answer is being prepared

7. Design synchronour decade counter using:

a. JK Flip-Flop 10

Answer is being prepared

b. Give the difference between the following 10
   i. Decoder and De-Multiplexer
   ii. Ripple counter and Synchronous counter
   iii. Latch and Flip-Flop

Answer is being prepared

SECTION-D

8. (i) Design the circuit of Half Adder using PLA 10

Answer is being prepared

(ii) Design BCD to XS-3 code converter using PLA 10

Answer is being prepared

9. (a) Realise the following function using ROM 10
   i. $F=\sum m(0,1,2,3)$
   ii. $F=\sum m(0,2,5)$

Answer is being prepared

b. With the help of state table and state diagram design a Mod-4 up/down counter. 10

Answer is being prepared