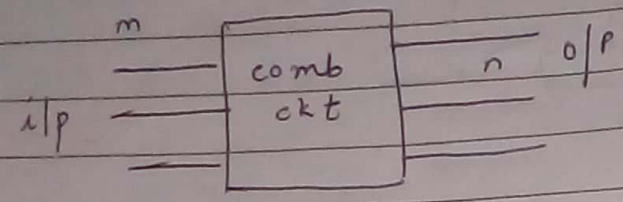
Unit-2: COMBINATIONAL CIRCUIT

In combinational circuit, output depends upon only present state not on past output and no feedback path is connected from output to input



DESIGN procedure to analyse any COMBINATIONAL CIRCUIT

Step1: Determine no. of i/p and no. of o/p.

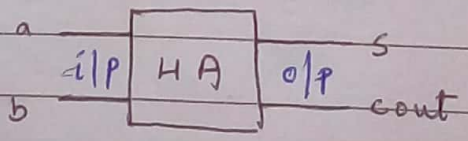
Step2: Draw the truth table

Step3: Now with the help of Boolean Algebra or K-Map, determine minimized value of every output which is f^n of i/p

Step4: Now with the help of Logic Gates draw every o/p w.r.t i/p

HALF ADDER

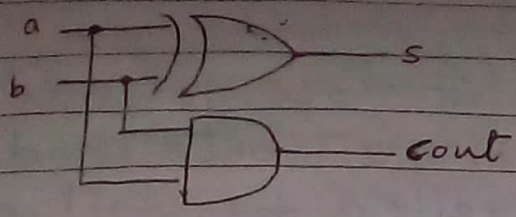
i/p		o/p	
a	b	s	cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



$S = a \oplus b$ $\quad$ $cout = a.b$

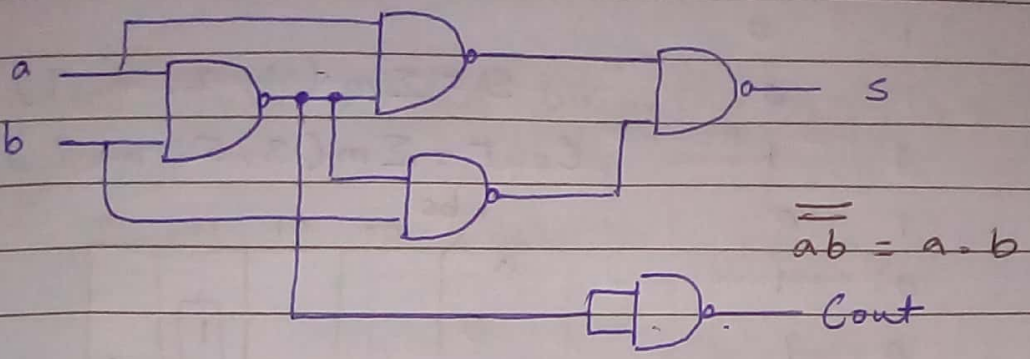
Page No. Date: / /201

Scanned by CamScanner



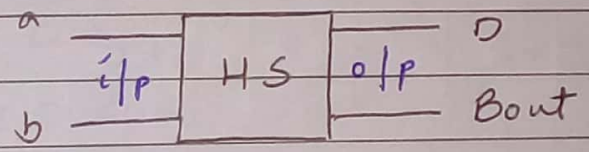
Q. Implement half adder by using min. no. of NAND gates.

5 Gates



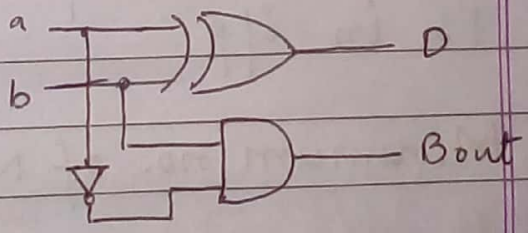
HALF SUBTRACTER

i/p		o/p	
a	b	D	Bout
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



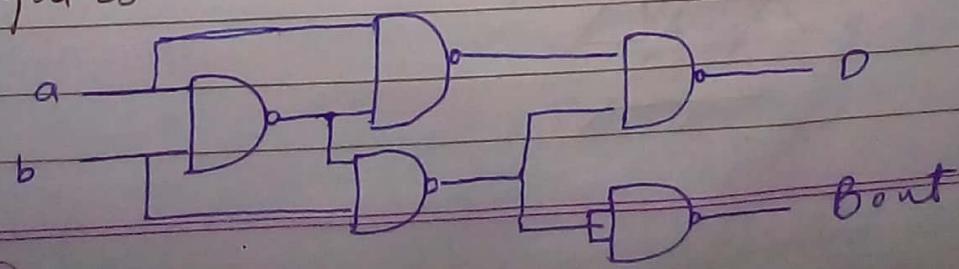
$D = a \oplus b$

$Bout = \bar{a} \cdot b$



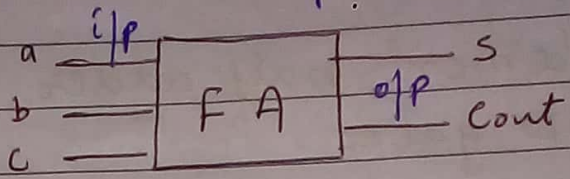
Q. Implement half subtracted by using min. no. of NAND gates

5 Gates



FULL ADDER

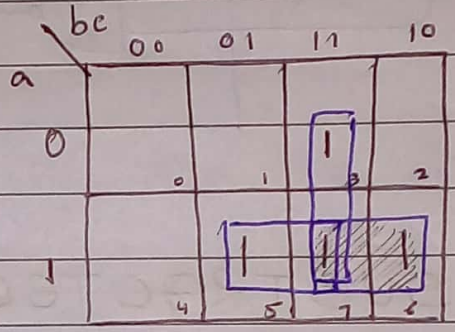
In half adder, it does not add carry, it simply adds 2 single bits and this problem is eliminated by full adder



i/p			o/p	
a	b	c	S	cout
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

$S = \Sigma m(1, 2, 4, 7) = a \oplus b \oplus c$

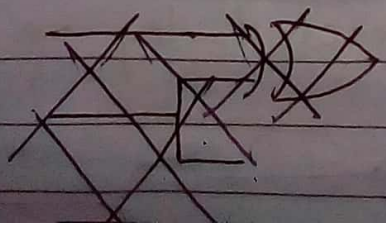
$Cout = \Sigma m(3, 5, 6, 7) = ab + bc + ca$

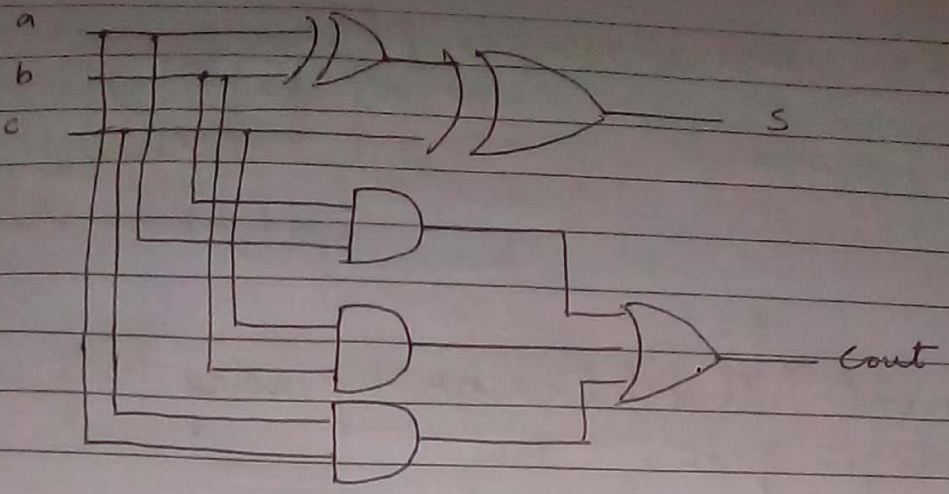


$bc + ac + ab$

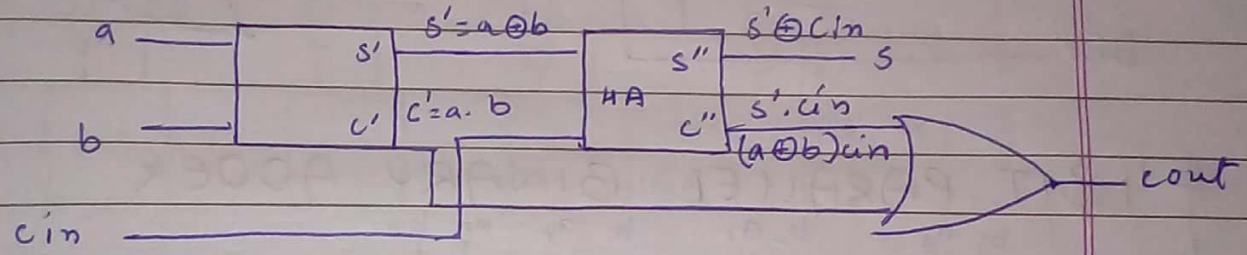
* sum of full adder will be high, when total no. of ~~one's~~ 1's at i/p are odd. Hence XOR exp^n.
* Cout will be high, when there are two or more 1's in i/p

Minimum no. of NAND gates = 9





Q. Implement full added by using half adder and logic gates.



$S = a \oplus b \oplus c_{in}$

$C_{out} = \Sigma m(3,5,6,7)$

011 101 110 111

$C_{out} = \bar{a}bc_{in} + a\bar{b}c_{in} + ab\bar{c_{in}}$

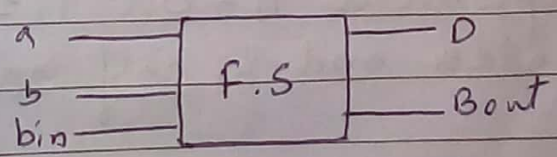
$= (\bar{a}b + \bar{b}a)c_{in} + ab(c_{in} + \bar{c}_{in})$

$= (a \oplus b)c_{in} + ab$

FULL SUBTRACTER

a-(b+bin)

a	b	bin	D	Bout
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



$D = \Sigma m(1,2,4,7)$

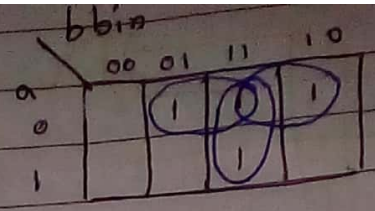
$= a \oplus b \oplus bin$

$B_{out} = \Sigma m(1,2,3,7)$

$= \bar{a}b + bbin + bin\bar{a}$

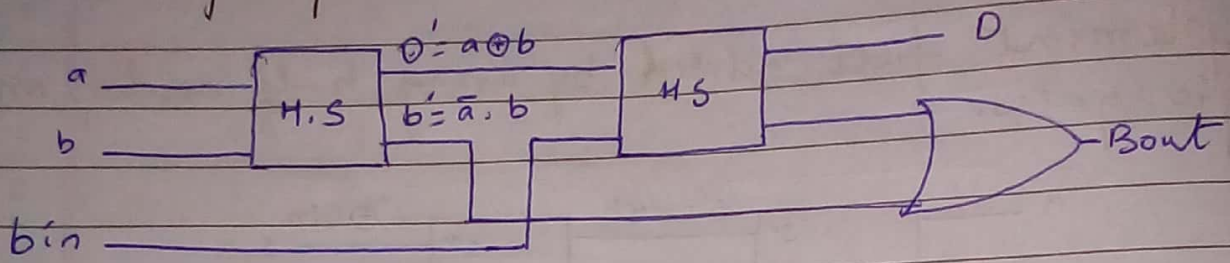
Date: / /201

Page No.

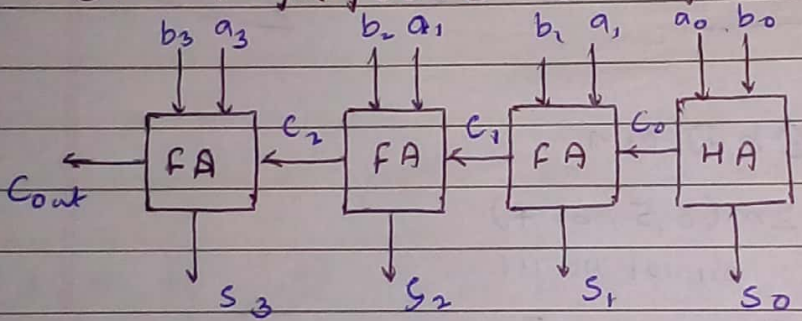


$\bar{a}b + bb_{in} + b_{in}\bar{a}$

Q. Implement full subtracter by using half subtracter and logic gate.



4 BIT PARALLEL BINARY ADDER



$C_2\ C_1\ C_0$

$a_3\ a_2\ a_1\ a_0$

$b_3\ b_2\ b_1\ b_0$

$S_3\ S_2\ S_1\ S_0$

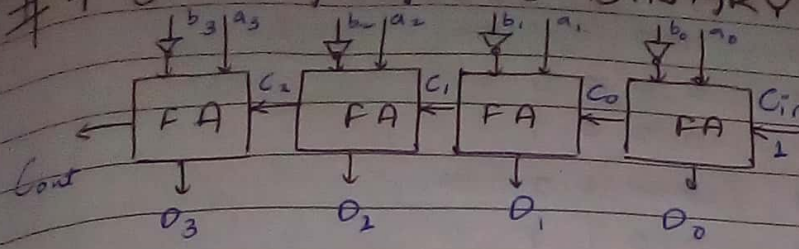
C_{out}

gmp.

* To implement n-bit ||el adder we require (n-1) full adder and 1 half adder or n full adder

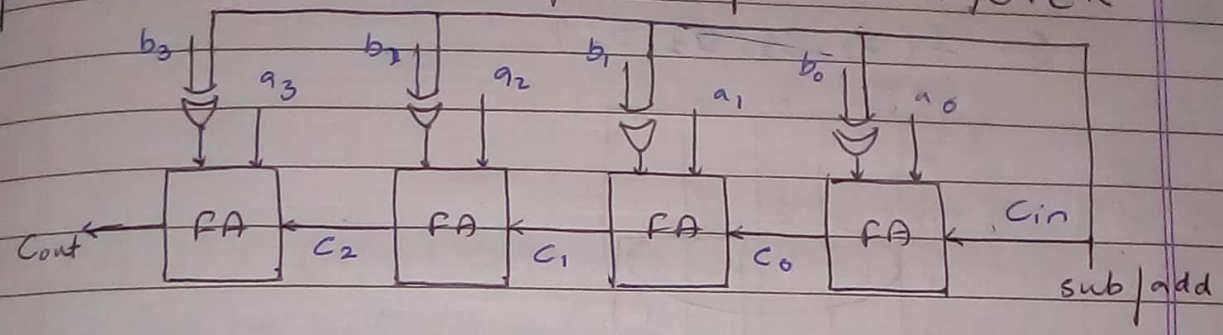
Date : / /201

4 BIT PARALLEL BINARY SUBTRACTER



0 1 1 1
− 0 1 0 1
1 0 1 0 → 1's
+ 1
1 0 1 1
+ 1 0 1 1
DISCARD ← (1) 0 1 1 0

4 BIT PARALLEL ADDER / SUBTRACTER



If sub/add = 0 , it works as 4 bit Ilel adder
= 1 , it works as 4 bit Ilel sub

Q. Minimum no. of half adder and or gates to implement 8 bit Ilel subtracter.

8 FA ⇒ 16 HA + 8 OR Gate

BCD ADDER

In this if sum > 9 add 6

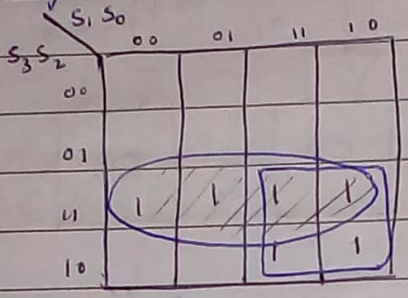
S/(cout)	S_3	S_2	S_1	S_0
0 0	0	0	0	0
1 0	0	0	0	1
2 0	0	0	1	0
3 0	0	0	1	1
4 0	0	1	0	0
5 0	0	1	0	1
6 0	0	1	1	0
7 0	0	1	1	1

Page No. Date : / /201

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

$S = \Sigma m (10, 11, 12, 13, 14, 15)$

$S = f(S_3, S_2, S_1, S_0)$



$S = S_3 S_2 + S_3 S_1$

1st 11cl adder

$a_3\ a_2\ a_1\ a_0$

$b_3\ b_2\ a_1\ b_0$

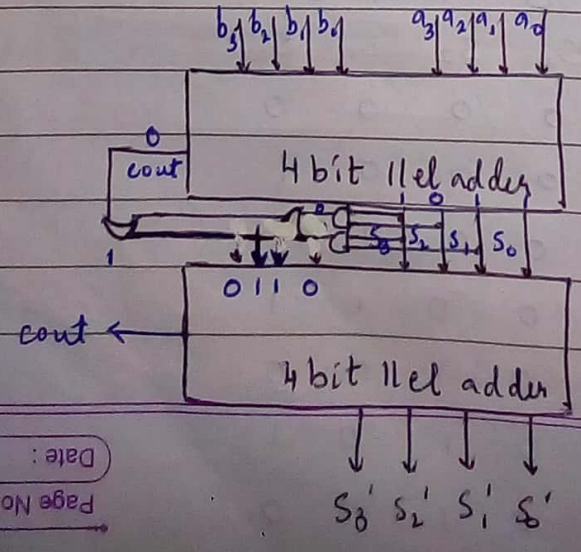
C_0 $S_3\ S_2\ S_1\ S_0$

2nd 11cl adder

0 0 0 0 if ≤ 9 or $C_0 = 0$

or 0 1 1 0

cout $S_3'\ S_2'\ S_1'\ S_0'$ if > 9 or $C_0 1$



Date : / /201

Page No.

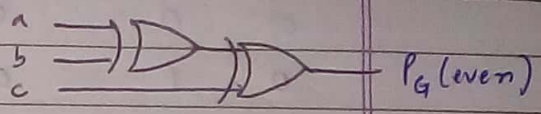
3 BIT PARITY GENERATOR

We check at output either there is even parity or odd parity

a	b	c	P_G(even)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$P_G = \Sigma m(1, 2, 4, 7)$

$= a \oplus b \oplus c$



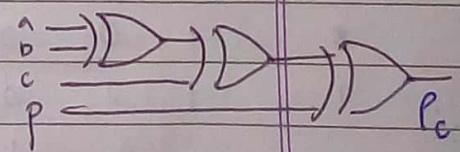
PARITY CHECKER

Parity bit is added with message signal and we check whether there is any error or not at output for even parity or odd parity.

a	b	c	p	P_c(even)
0	0	0	0	0
0	0	0	1	1 ✓
0	0	1	0	1 ✓
0	0	1	1	0
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
1	0	1	0	0
1	0	1	1	1 ✓
1	1	0	0	0
1	1	0	1	1 ✓
1	1	1	0	1 ✓
1	1	1	1	0

$P_c = \Sigma m(1, 2, 4, 7, 8, 11, 13, 14)$

$= a \oplus b \oplus c \oplus p$



Date : / /201

Page No.

MAGNITUDE COMPARATOR

It compares 2 no. and gives 3 output $a>b$, $a=b$ and $a<b$.

1 Bit Magnitude Comparator

i/p		output		
a	b	a>b	a=b	a<b
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$f_{a>b} = a.\bar{b}$

$f_{a=0} = a \odot b$

$f_{a<b} = \bar{a}.b$

2 Bit Magnitude Comparator

It compares 2 bits of a & b

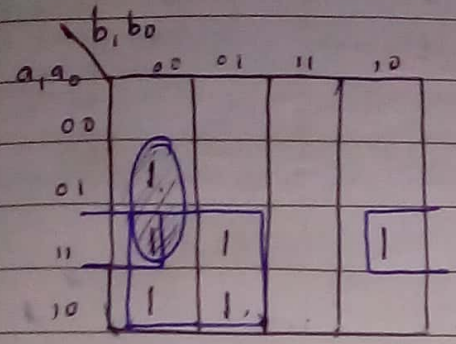
	a_1	a_0	b_1	b_0	$f_{a>b}$	$f_{a=b}$	$f_{a<b}$
0	0	0	0	0	0	1	0
1	0	0	0	1	0	0	1
2	0	0	1	0	0	0	1
3	0	0	1	1	0	0	1
4	0	1	0	0	1	0	0
5	0	1	0	1	0	1	0
6	0	1	1	0	0	0	1
7	0	1	1	1	0	0	1
8	1	0	0	0	1	0	0
9	1	0	0	1	1	0	0
10	1	0	1	0	0	1	0
11	1	0	1	1	0	0	1
12	1	1	0	0	1	0	0
13	1	1	0	1	1	0	0
14	1	1	1	0	1	0	0
15	1	1	1	1	0	1	0

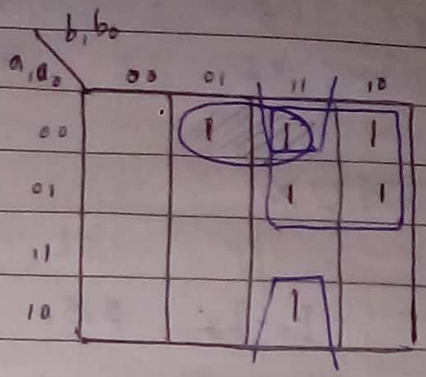
Page No.
Date : / /201

$f_{a>b} = \Sigma m(4, 8, 9, 12, 13, 14) = a_1\bar{b}_1 + a_0 b_1 \bar{b}_0 + a_1 a_0 \bar{b}_0$

$f_{a=b} = \Sigma m(0, 5, 10, 15)$

$f_{a<b} = \Sigma m(1, 2, 3, 6, 7, 11) = \bar{a}_1 b_1 + \bar{a}_1 \bar{a}_0 b_0 + \bar{a}_0 b_1 b_0$





$f_{a=b} = \Sigma m = (0, 5, 10, 15)$

$= \bar{a}_1 \bar{a}_0 \bar{b}_1 \bar{b}_0 + \bar{a}_1 a_0 \bar{b}_1 b_0 + a_1 \bar{a}_0 b_1 \bar{b}_0 + a_1 a_0 b_1 b_0$

$= \bar{a}_1 \bar{b}_1 (\bar{a}_0 \bar{b}_0 + a_0 b_0) + a_1 b_1 (\bar{a}_0 \bar{b}_0 + a_0 b_0)$

$= \bar{a}_1 \bar{b}_1 (a_0 \odot b_0) + a_1 b_1 (a_0 \odot b_0)$

$= (\bar{a}_1 \bar{b}_1 + a_1 b_1)(a_0 \odot b_0)$

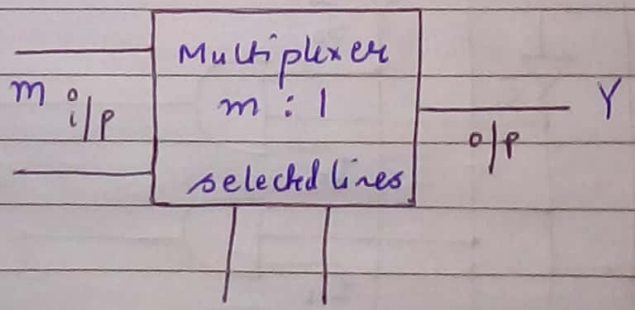
$= (a_1 \odot b_1)(a_0 \odot b_0)$

MULTIPLEXER

$m = 2^s$

m = no. of i/p

s = selected lines



It has many i/p and only one o/p and it connects only one i/p with o/p depending upon the value of selected lines. That's why it is also called data selector or many to one logic circuit or Ilel to serial converter.

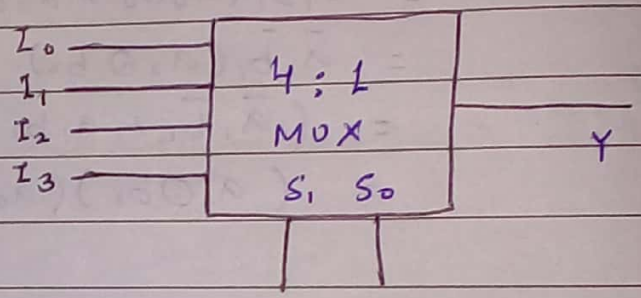
MUX	i/ps	slines	o/p
2:1	2	1	1
4:1	4	2	1
8:1	8	3	1
16:1	16	4	1

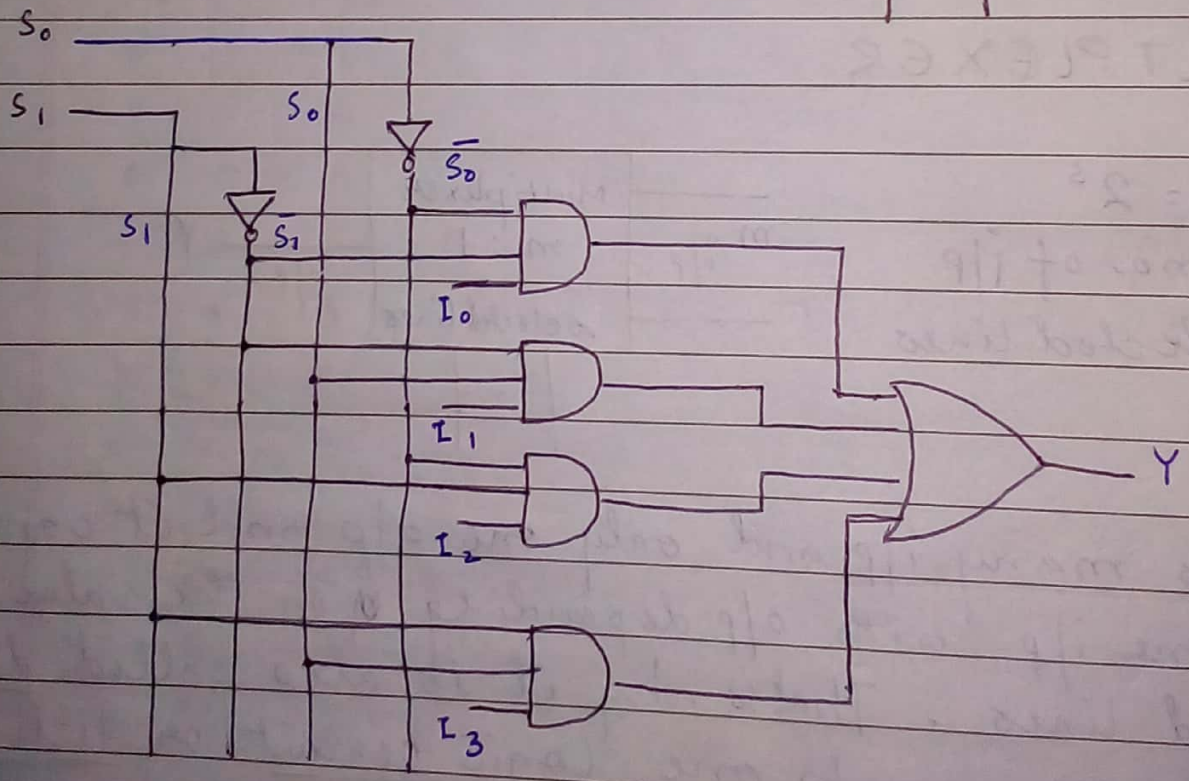
4:1 MUX

	i/p	sl	o/p
4:1	I_0-I_3	S_1S_0	Y

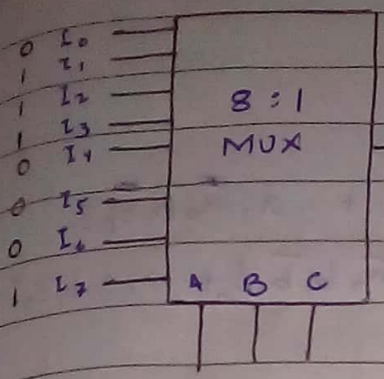
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$$Y = \bar{S_1}\bar{S_0} I_0 + \bar{S_1} S_0 I_1 + S_1 \bar{S_0} I_2 + S_1 S_0 I_3$$





Q. Implement $f = \Sigma m(1, 2, 3, 7)$ by using 8 to 1 MUX

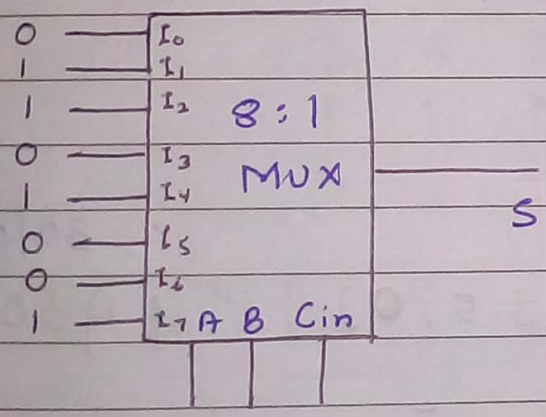


$$f = \bar{A}\bar{B}CI_1 + \bar{A}B\bar{C}I_2 + \bar{A}BCI_3 + ABCI_7$$

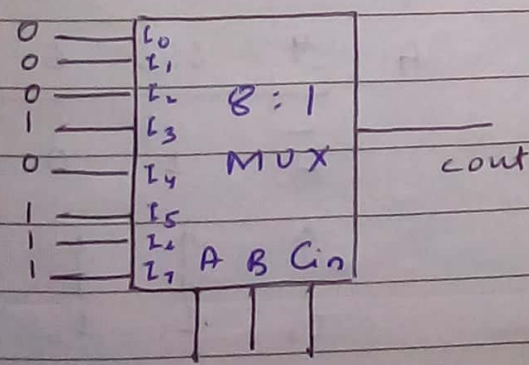
In Multiplexer, output is in S.O.P. form and all the minterms given in f" are connected to +5V and remaining terms are connected to GND and only one output is possible in one line

Q. Implement full adder by using 8:1 MUX

$f(A, B, C_{in}) = \Sigma m(1, 2, 4, 7)$



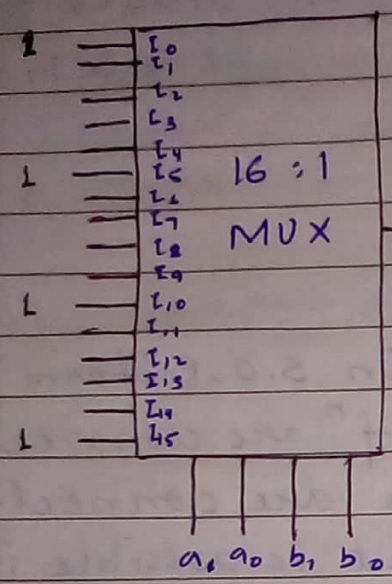
$$S = AB\bar{C}_{in}I_1 + A\bar{B}C_{in}I_2 + \bar{A}BC_{in}I_4 + ABC_{in}I_7$$



$$Cout = A\bar{B}\bar{C}_{in}I_3 + \bar{A}B\bar{C}_{in}I_5 + AB\bar{C}_{in}I_6 + ABC_{in}I_7$$

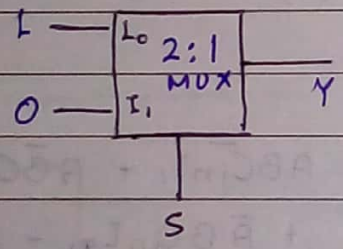
Scanned by CamScanner

Q. Implement $f_{a=b}$ of 2 bit Magnitude comparator by using 16:1 MUX.



$f_{a=b} = \bar{a}_1 \bar{a}_0 \bar{b}_1 \bar{b}_0 I_0 + \bar{a}_1 a_0 b_1 b_0 I_5 + a_1 \bar{a}_0 b_1 \bar{b}_0 I_{10} + a_1 a_0 b_1 b_0 I_{15}$

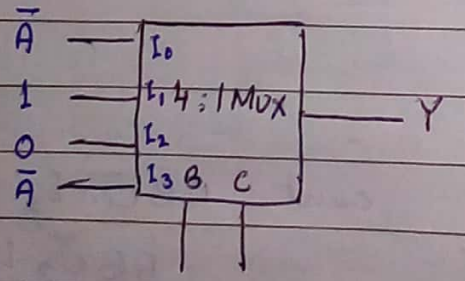
Q. Implement NOT gate using 2:1 MUX



$Y = \bar{S} I_0 + S I_1$

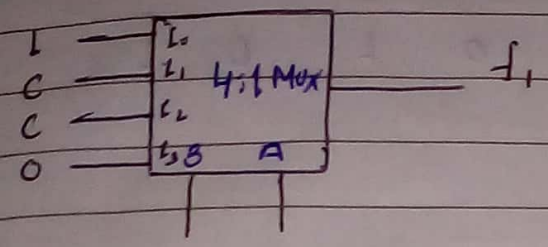
30/08/17

Q. Implement $f_1 = \Sigma m(1, 3, 5, 6)$, f^n of ABC



$\bar{A}$	(0)	(1)	2	(3)
A	4	(5)	6	7
	$\bar{A}$	1	0	$\bar{A}$
	I_0	I_1	I_2	I_3

$\bar{C}$	(0)	2	4	6
C	(1)	(3)	(5)	7
	↑L	↑C	↑C	↑O
	I_0	I_1	I_2	I_3

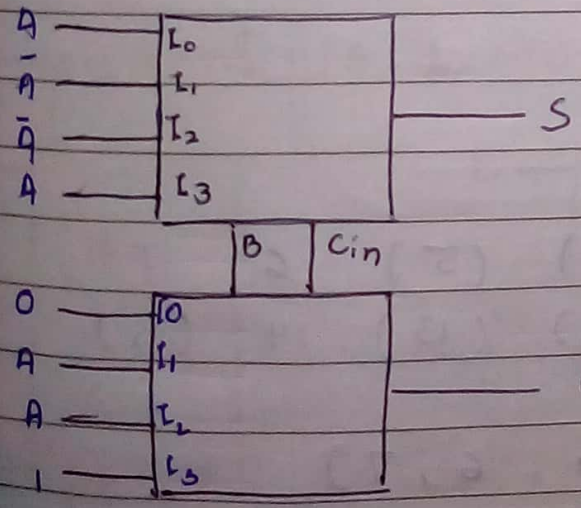


Q. Implement full adder by using 4:1 mux. (if B & C are selected lines)

$$S = \Sigma m(1, 2, 4, 7)$$

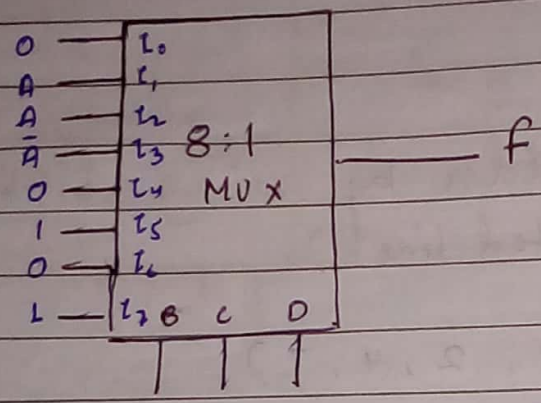
$$Cout = \Sigma m(3, 5, 6, 7)$$

$\bar{A}$	0	(1)	(2)	3
A	(4)	5	6	(7)
	A	$\bar{A}$	$\bar{A}$	A
	↑	↑	↑	↑
	I_0	I_1	I_2	I_3

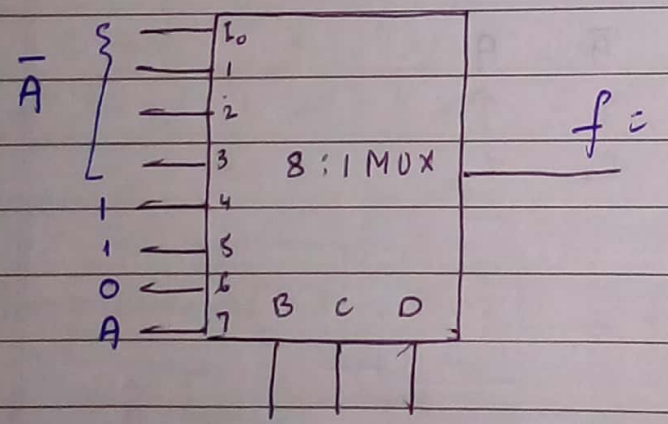


Q. Implement $f = \Sigma m(3,5,7,9,10,13,15)$ by using 8:1 MUX

$\bar{A}$	0	1	2	(3)	4	(5)	6	(7)
A	8	(9)	(10)	11	12	(13)	14	(15)
	0	A	A	$\bar{A}$	0	1	0	1



Q. In the following MUX circuit, what will be value of f^m in maxterm



$\bar{A}$	(0)	(1)	(2)	(3)	(4)	(5)	6	7
A	8	9	10	11	(12)	(13)	14	(15)

$f = \Pi M(8,9,10,11,14,6,7)$

Q. Minimum no. of 2:1 MUX req. to implement Half Adder.

A	B
0	0
0	1
1	0
1	1

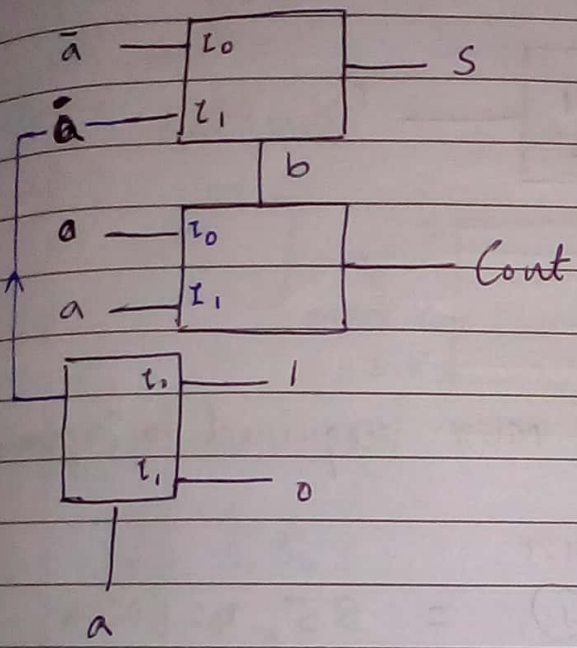
$S = \Sigma m(1, 2)$

$Cout = \Sigma m(3)$

0	(1)
(2)	3

a ā

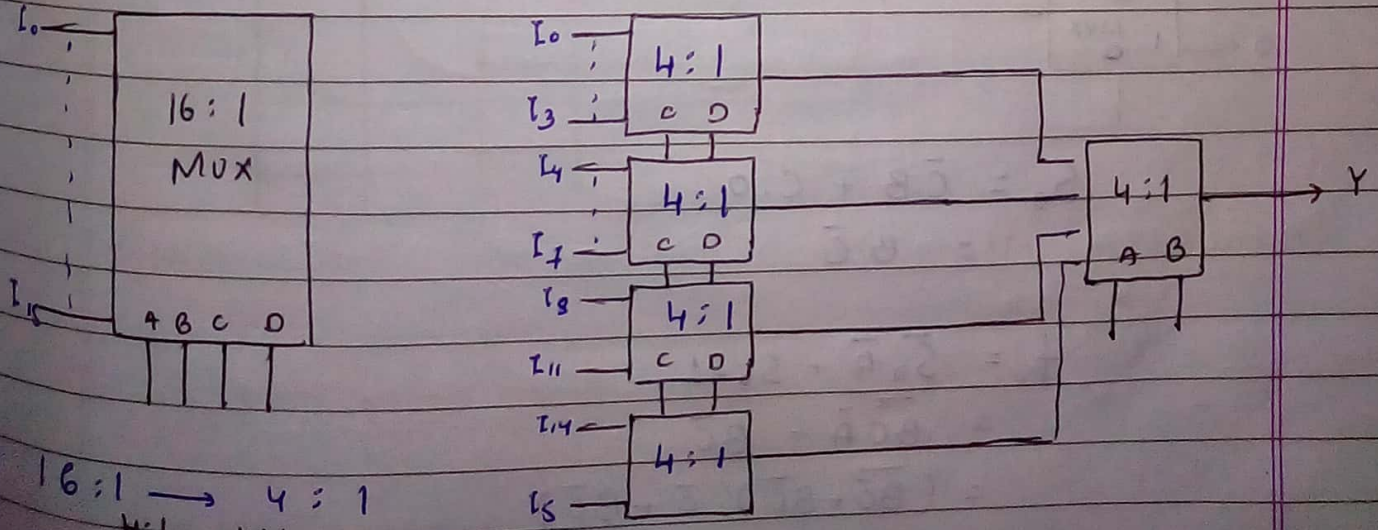
0	1
2	(3)
0	^



3, 2:1 MUX required

MULTIPLEXER TREE

Q. Implement 16:1 MUX by using 4:1 MUX



16:1 → 4:1

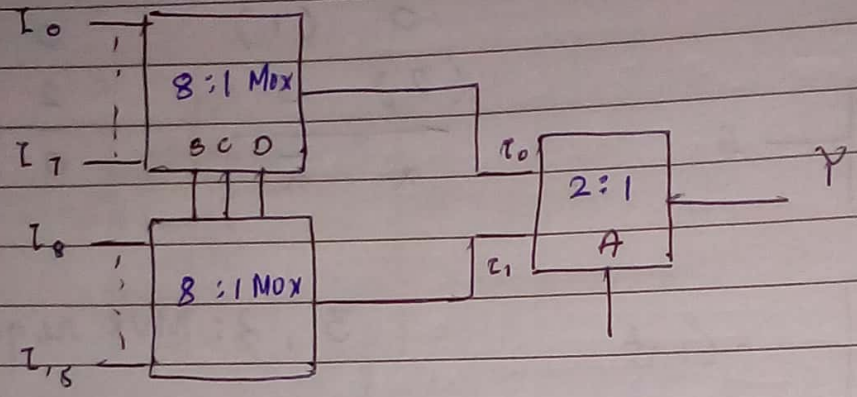
$\frac{16}{4} = 4 + 1 = 5$

Q. Implement 16:1 MUX by using 8:1 MUX and 2:1 MUX

16:1 → 8:1

$\frac{16}{8} = \underset{8:1}{(2)} = \underset{2:1}{(1)}$

$\frac{2}{2}$

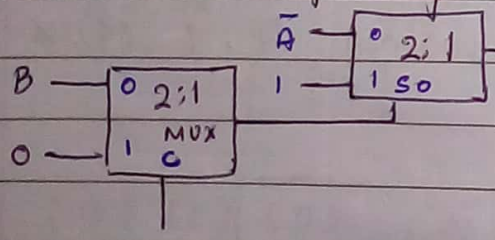


Q. Determine total no. of 4:1 MUX required to implement 256:1 MUX

4:1 4:1 4:1 4:1

$\frac{256}{4} = \underset{4}{(64)} + \underset{4}{(16)} + \underset{4}{(4)} + \underset{}{(1)} = 85$, 4:1 MUX

Q Determine output f^m in the following circuit



$S_0 = \bar{C}B + C.0$

$= B\bar{C}$

$f = \bar{S}_0\bar{A} + S_0.1$

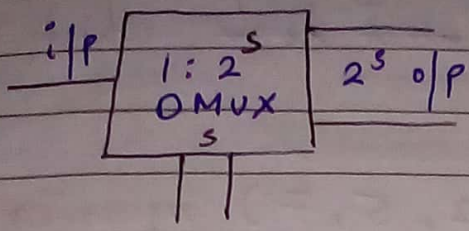
$= \overline{B\bar{C}}\,\bar{A} + B\bar{C}$

$= (\overline{B\bar{C}} + B\bar{C})(\bar{A} + B\bar{C})$

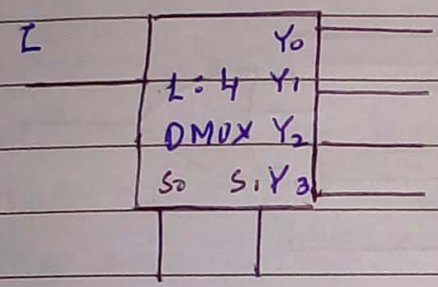
$= \bar{A} + B\bar{C}$

DEMULTIPLEXER (DEMUX):

Reverse process of MUX. It has only one i/p and many o/p



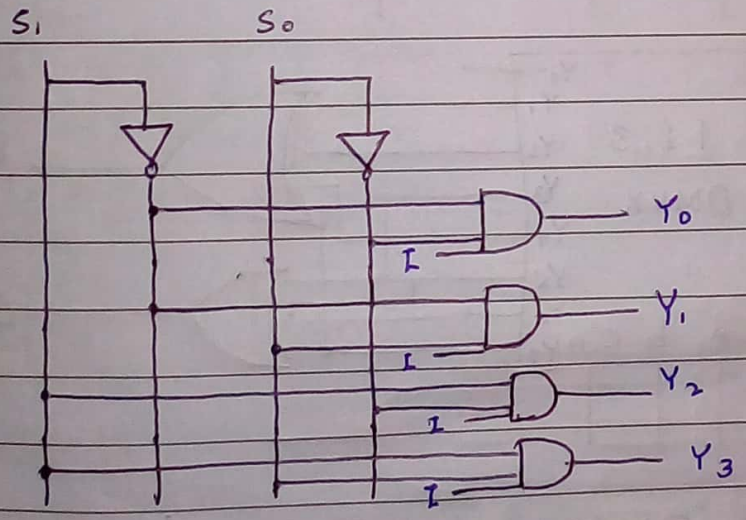
1:4 DMUX



S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

$Y_0 = \bar{S_1}\bar{S_0} I$ $Y_1 = \bar{S_1} S_0 I$ $Y_2 = S_1 \bar{S_0} I$

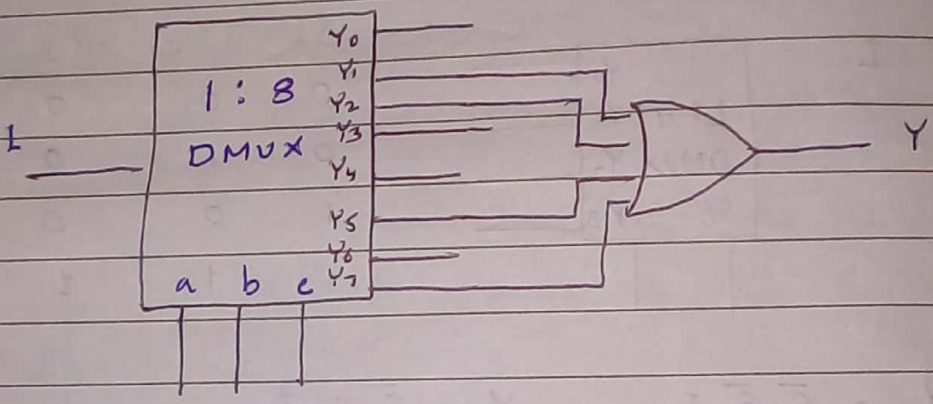
$Y_3 = S_1 S_0 I$



IMPLEMENTATION of any f^n using DMUX

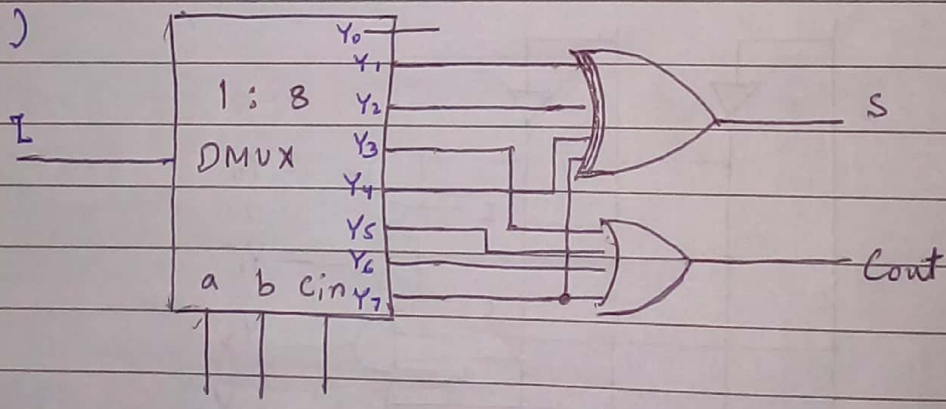
Connect all the minterms (outputs) to input of OR gate which are given in f^n and leave remaining minterms ~~and leave~~ which are not given in f^n.

Q Implement $Y = \Sigma m(1, 2, 5, 7)$ by using 1:8 DMUX and this f^n is dependent upon ABC variable.



Q. Implement full adder using 1:8 DMUX

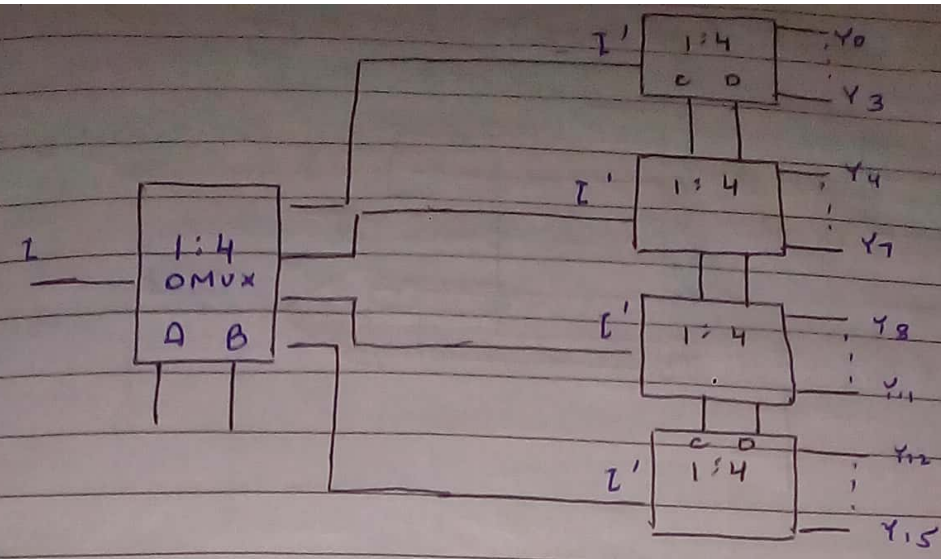
$S = \Sigma m(1, 2, 4, 7)$



Q. Implement 1:16 DMUX by using 1:4 DMUX

1:4 1:4

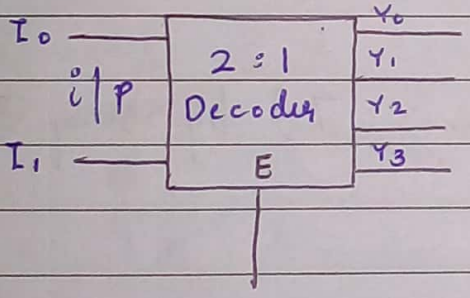
$\frac{16}{4} = (4) + (1) = (5)$ 1:4 DMUX required.



DECODER

It converts m lines to 2^m lines. i.e.
It has m inputs and 2^m outputs.
It has one enable line. Decoder will only works if enable line is activated. e.g 2:4, 3:8, 4:16, 5:32

2:4 DECODER

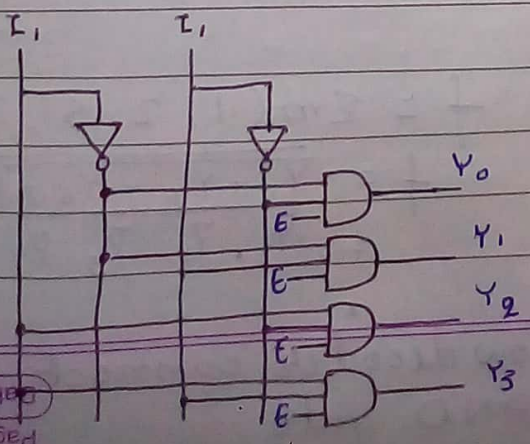


E	I_1	I_0	Y_3	Y_2	Y_1	Y_0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

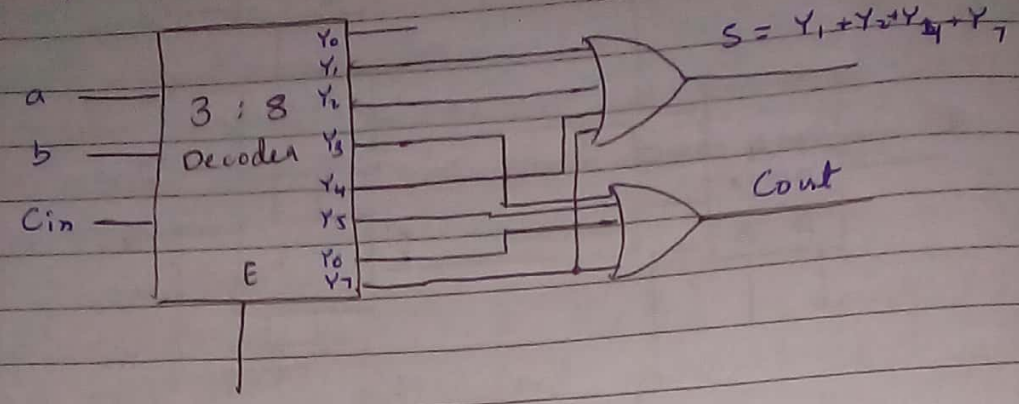
Active High 2:4 Decoder

0	x	x	Not Working

$Y_0 = \bar{I_0}\,\bar{I_1}\,E$ $\quad Y_1 = \bar{I_1}\,I_0\,E$ $\quad Y_2 = I_1\,\bar{I_0}\,E$ $\quad Y_3 = I_1 I_0 E$

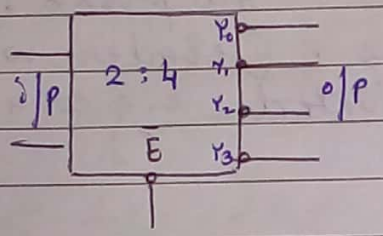


Q. Implement full adder by using 3:8 Decoder.



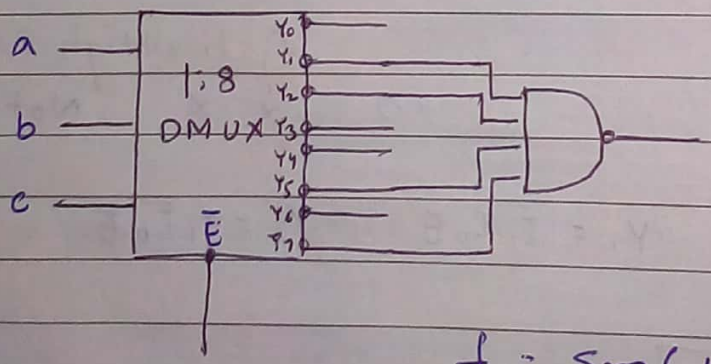
2:4 Active LOW DECODER

In this decoder will only work when value of enable line is 0.



$\bar{E}$	I_1	I_0	Y_3	Y_2	Y_1	Y_0
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	x	x	not working.			

Q. Implement $f = \Sigma m(1, 2, 5, 7)$ by using 3:8 active low decoder.



$$f = \Sigma m(1, 2, 5, 7)$$

$$f = \overline{\overline{Y_1 + Y_2 + Y_5 + Y_7}}$$

$$= \bar{Y}_1 \cdot \bar{Y}_2 \cdot \bar{Y}_5 \cdot \bar{Y}_7$$

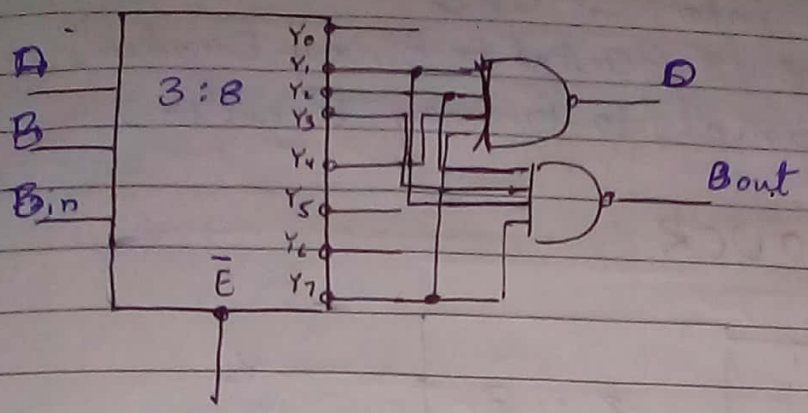
In active low decoder connect output with NAND gate

Date: / /201

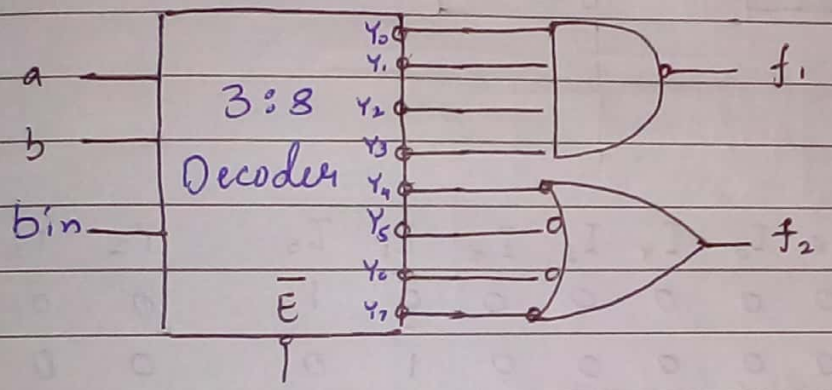
Page No.

01/09/17

Q. Implement full subtracter by using active low (-ve logic) 3:8 decoder.



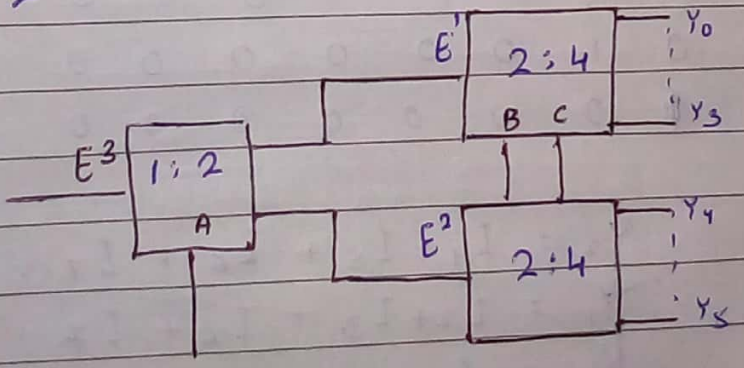
Q. What will be value of f_1 and f_2 in the following circuit



$f1 . f2 = 0$

Q. Implement 3:8 decoder by using 2:4 and 1:2 decoder

$\frac{8}{4} = \frac{②}{2} = ①$ (2:4, 1:2)

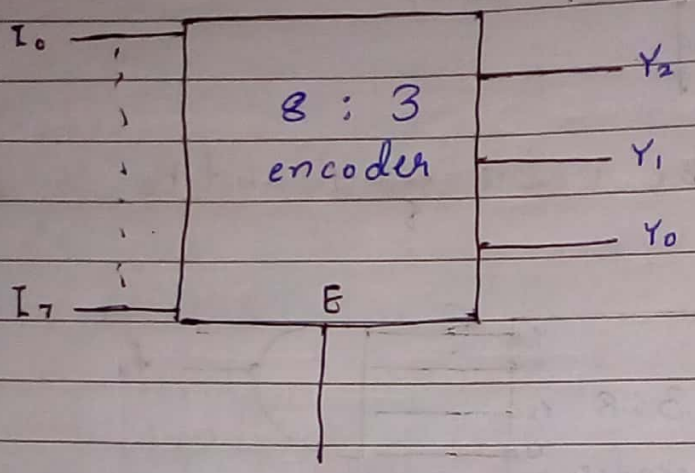


ENCODER

Reverse process of decoder. It converts 2^m lines into m lines.

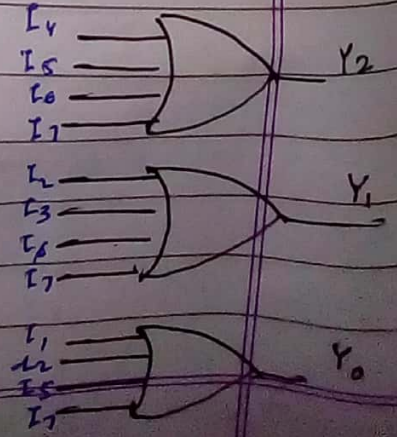
e.g 4:2, 8:3 (Octal to Binary Encoder), 16:4 (Hexadecimal to Binary Encoder).

8:3 ENCODER



E	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	1	0	0	0	1
1	0	0	0	0	0	1	0	0	0	1	0
1	0	0	0	0	1	0	0	0	0	1	1
1	0	0	0	1	0	0	0	0	1	0	0
1	0	0	1	0	0	0	0	0	1	0	1
1	0	1	0	0	0	0	0	0	1	1	0
1	1	0	0	0	0	0	0	0	1	1	1

$$Y_2 = I_4 + I_5 + I_6 + I_7$$
$$Y_1 = I_2 + I_3 + I_6 + I_7$$
$$Y_0 = I_1 + I_3 + I_5 + I_7$$



PRIORITY ENCODER

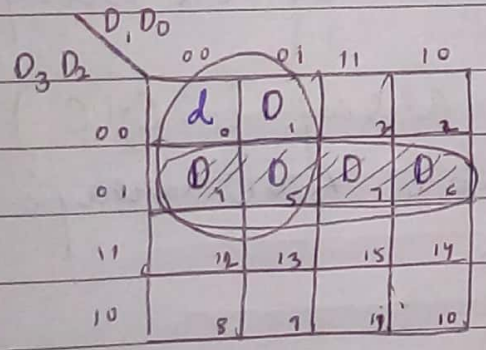
In octal to binary encoder, if at one time, more than one i/p are high or if all i/p are low we do not get any output and this problem is avoided by priority encoder. In this we set highest priority to that no. which has highest binary value.

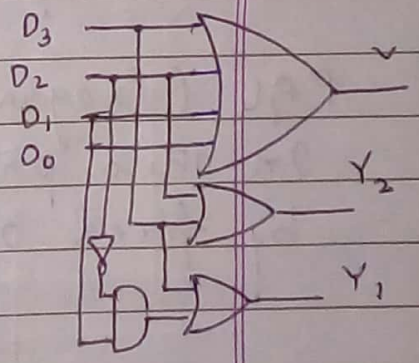
D_3	D_2	D_1	D_0	Y_2	Y_1	V
1	x	x	x	1	1	1
0	1	x	x	1	0	1
0	0	1	x	0	1	1
0	0	0	1	0	0	1
0	0	0	0	x	x	0

$$V = \pi M(0) = D_3 + D_2 + D_1 + D_0$$

$$Y_2 = \pi M(1,2,3) + \Sigma d(0)$$

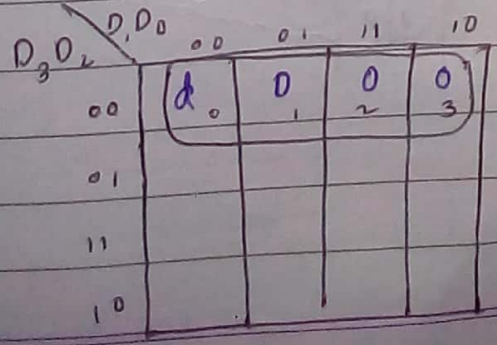
$$Y_1 = \pi M(1,4,5,6,7) + \Sigma d(0)$$





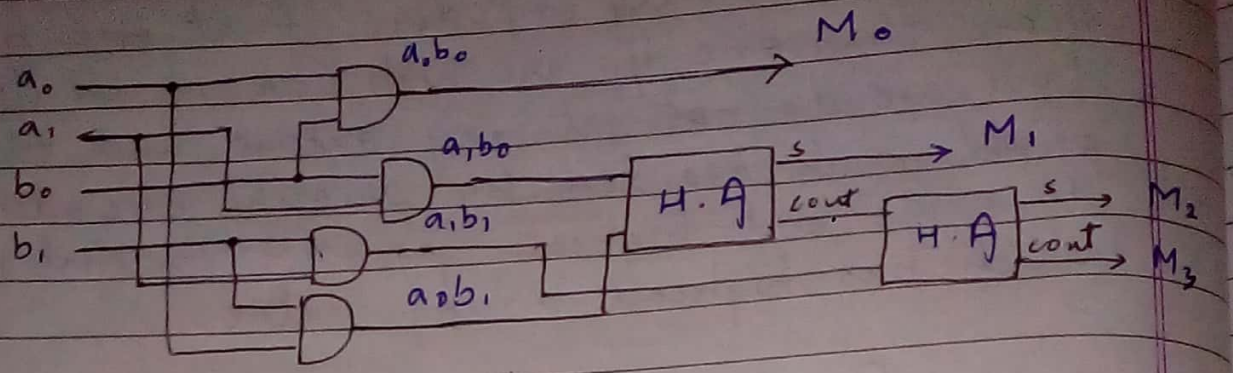
$$Y_1 = (D_3 + D_1)(D_3 + \bar{D_2})$$

$$= D_3 + D_1\bar{D_2}$$

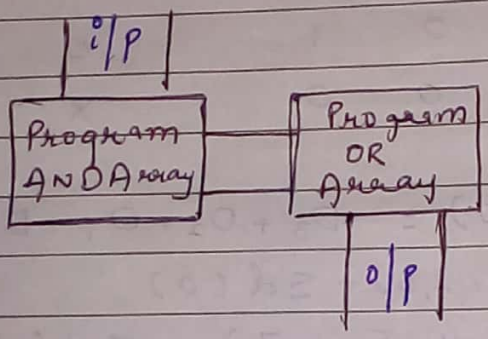


$$Y_2 = D_3 + D_2$$

2 BIT BINARY MULTIPLIER

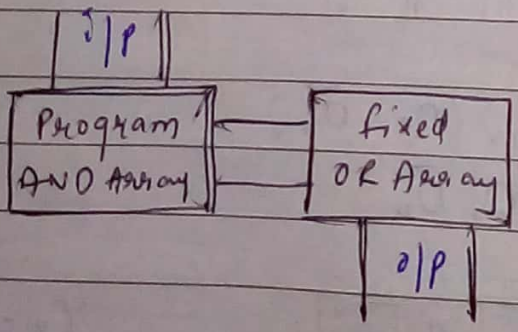


PLA (Programmable LOGIC Array)
In this both AND & OR array are programmable



AND array's are followed by OR array's.

PAL (Programmable array LOGIC)
In this programmable AND arrays are followed by fixed OR arrays

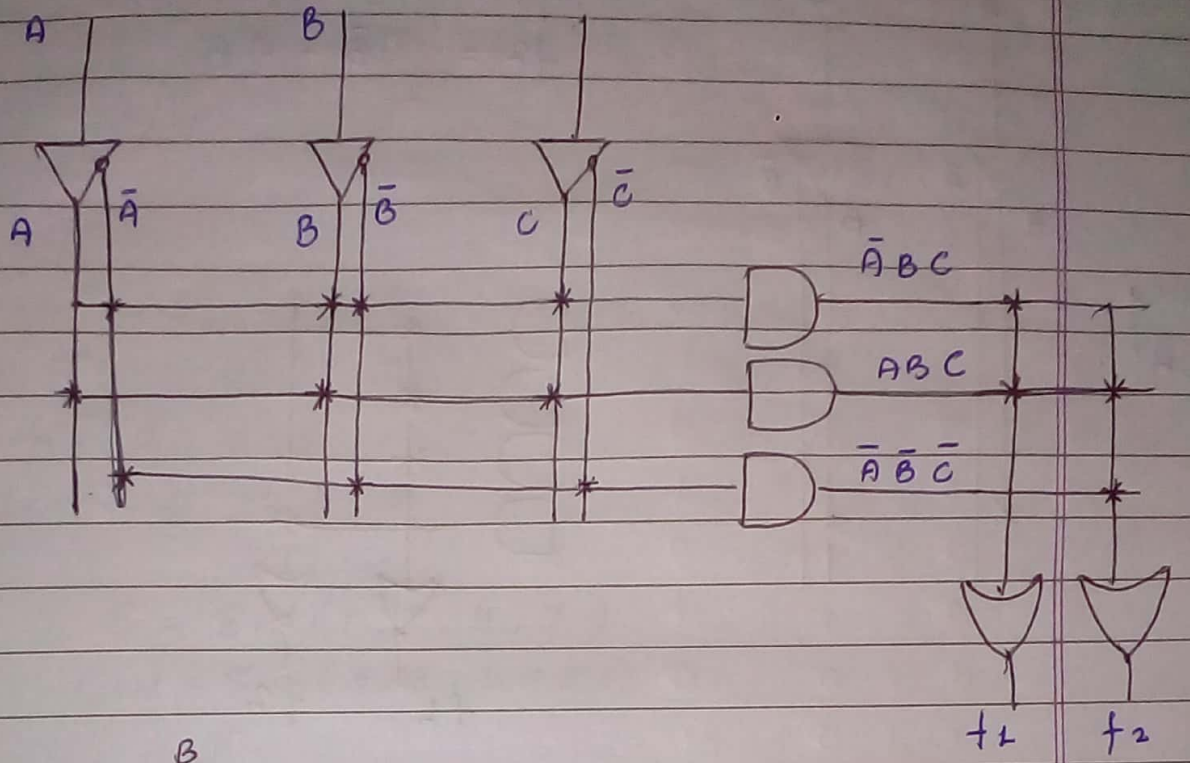


Q. Implement the following f^n by using PLA & PAL

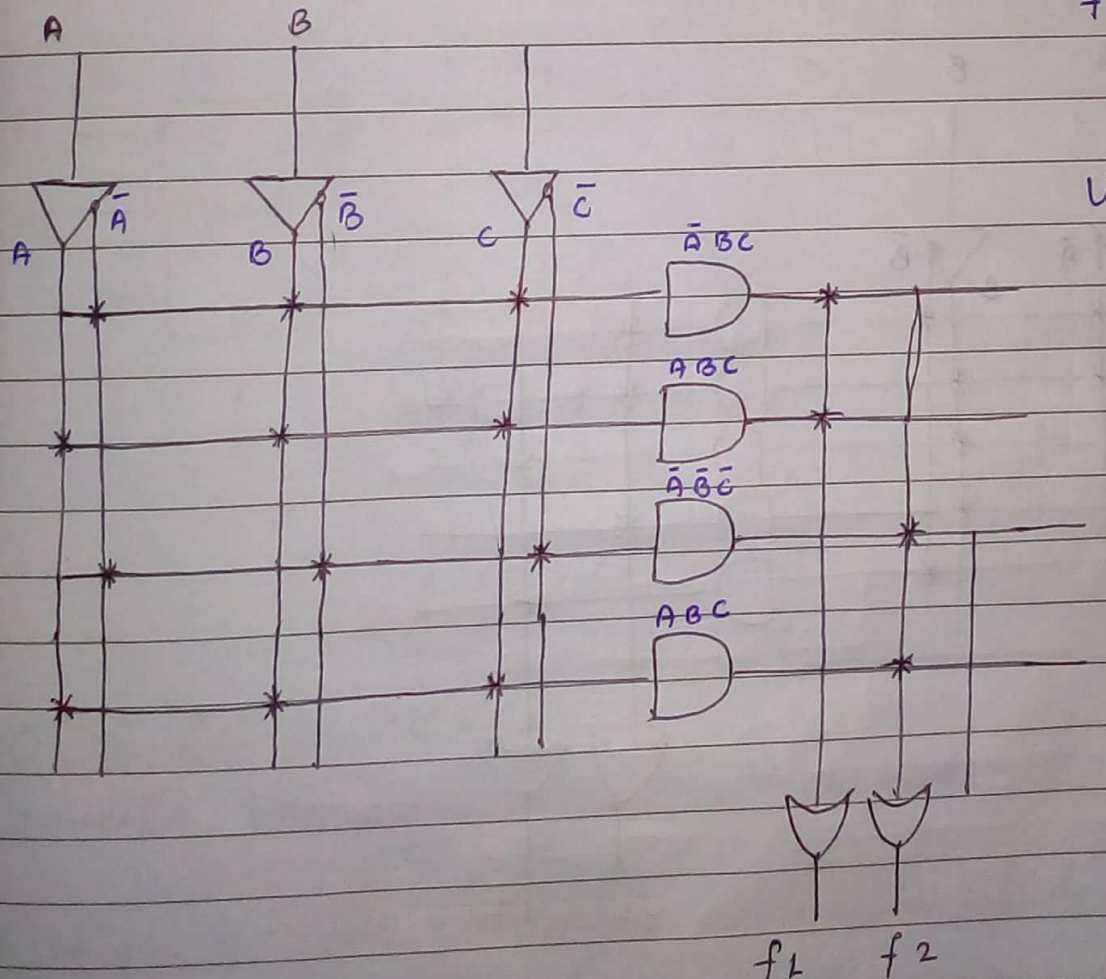
$f_1 = \bar{A}BC + ABC$

$f_2 = \bar{A}\bar{B}\bar{C} + ABC$

Using PLA



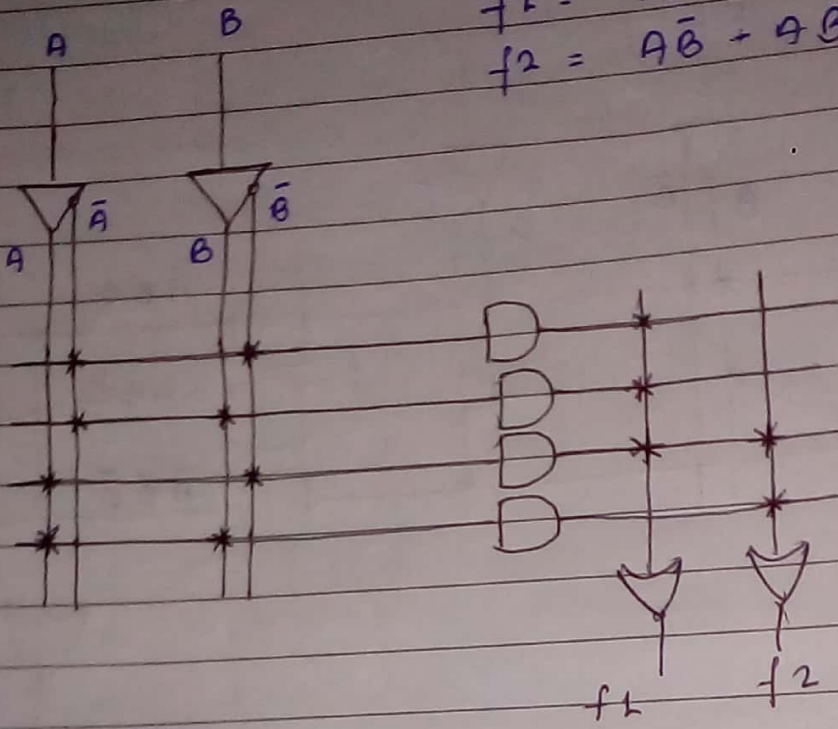
Using PAL



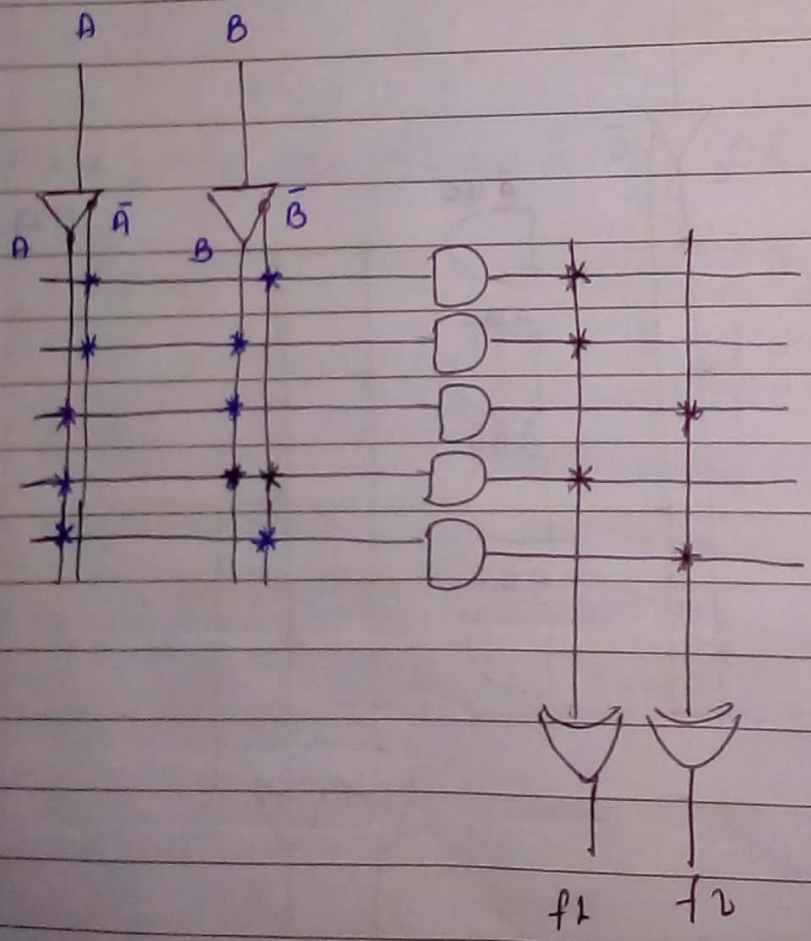
Q. Implement $f1 = \Sigma m(0,1,2)$ by PAL & PLA
$f2 = \Sigma m(2,3)$

$f1 = \bar{A}\bar{B} + \bar{A}B + A\bar{B}$
$f2 = A\bar{B} + AB$

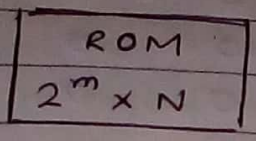
Using PLA



Using PAL



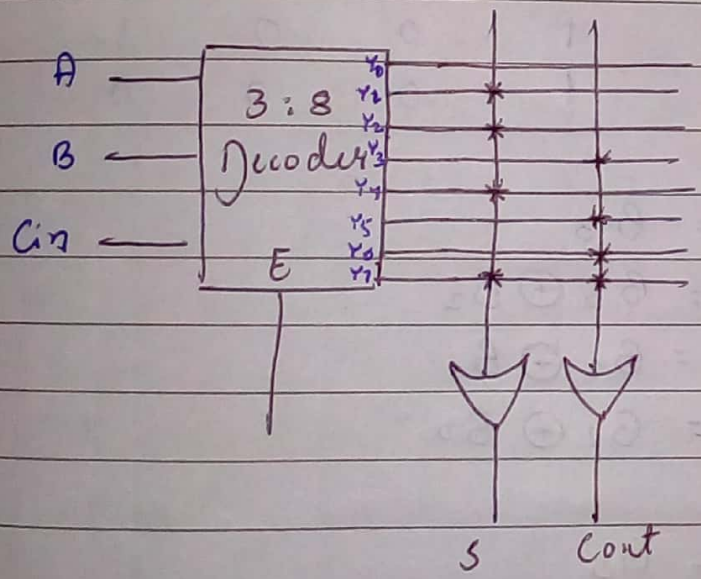
E-PROM [ROM]



m = total address lines
N = length of data bus

Q. Design full adder by using ROM
In ROM both AND arrays and OR arrays are fixed and with the help of decoder we can implement any fn.

$S = \Sigma m(1, 2, 4, 7)$
$Cout = \Sigma m(3, 5, 6, 7)$



CODE CONVERTER

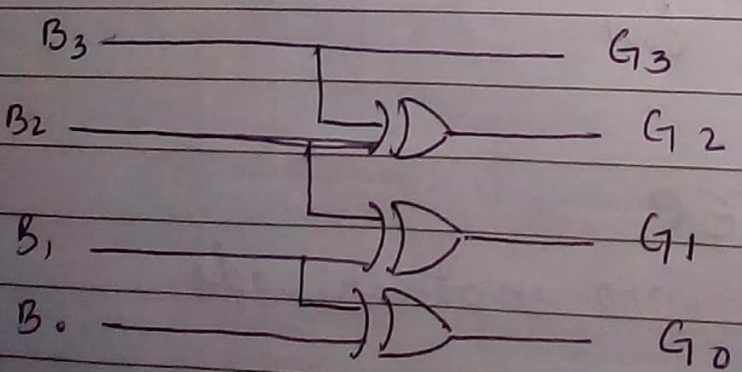
Converts one code into another code.

Q. Design a code converter of 4 Bits which converts binary code into Grey code

Date: / /201

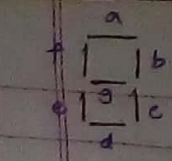
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

$$G_3 = B_3$$
$$G_2 = B_3 \oplus B_2$$
$$G_1 = B_2 \oplus B_1$$
$$G_0 = B_1 \oplus B_0$$



07/09/17

SEVEN SEGMENT CODE :



BCD	a	b	c	d	e	f	g
8	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	0	0	1	1

Q. Seven Segment has two push buttons P_1 and P_2, when a button is pressed the prize of displayed in Seven Segment display.
If no buttons are pressed 0 is displayed.
If P_1 is pressed but P_2 not 2 is displayed.
If P_2 is pressed only 5 is displayed.
If both are pressed there is error and it signifying by E.

* If segment a to g are considered with option is correct

a) $g = \bar{P_1} + P_2$, $d = c + e$

b) $g = P_1 + P_2$, $e = b + c$

c) $g = P_1 + P_2$ $d = c + e$

* Minimum no. of NOT & OR gates are required to design this display.

P_1	P_2	a	b	c	d	e	f	g
0	0	1	1	1	1	1	1	0
0	1	1	0	1	1	0	1	1
1	0	1	1	0	1	1	0	1
1	1	1	0	0	1	1	1	1

Page No.
Date : / /201

$a = 1$

$b = \overline{P_2}$

$c = \overline{P_1}$

$e = P_1 + \overline{P_2}$

$f = \overline{P_1} + P_2$

$g = P_1 \cdot P_2$

$d = \overline{P_2} + P_1 + P_2$

$d = 1 + P_1 = 1$

2 NOT 3 OR