Half Adder

[I/P: A augent & B addend bits O/P: sum : carry]

The logic circuit which performs two bit binary Addition is called Half Adder.

(1) Truth table

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(2) Kmap

Sum

A \ B	0	1
0	0	1
1	1	0

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

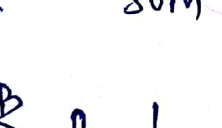
$= A \oplus B$

Carry

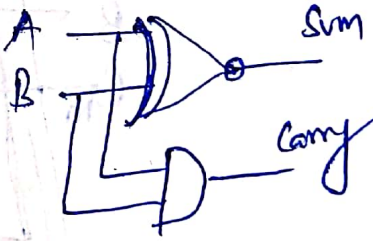
A \ B	0	1
0	0	0
1	0	1

$C = AB$

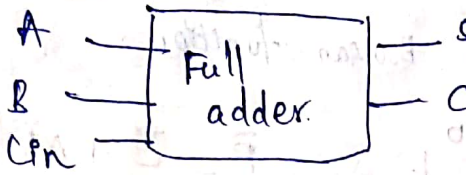
(3)



(4)



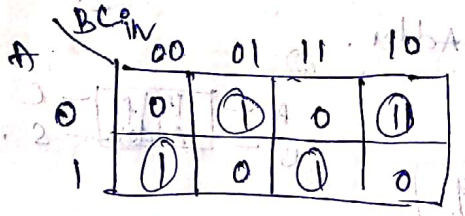
Full Adder Circuits that performs 3 input Addition is called Full Adder.



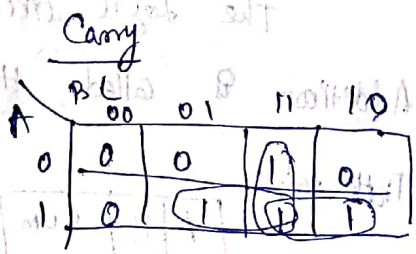
Truth table

A	B	C_{in}	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

Kmap for Sum



Carry

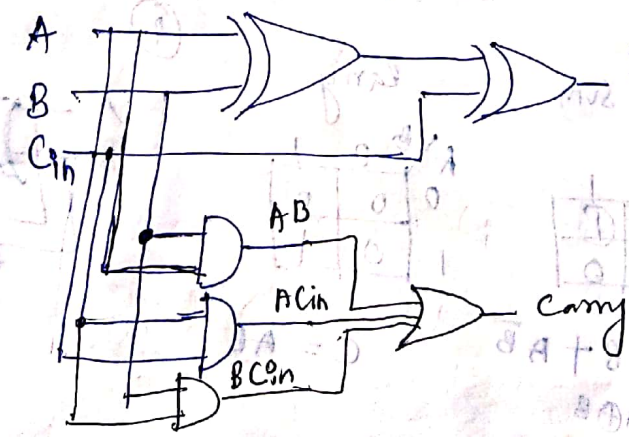


$Sum = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$

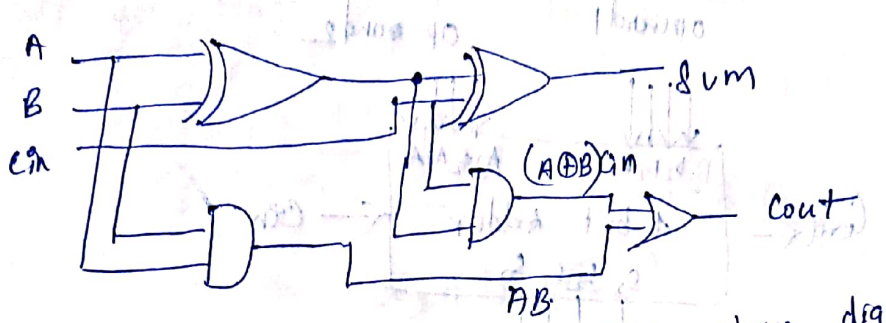
$= A \oplus B \oplus C_{in}$

Carry

$Cout = BC_{in} + AC_{in} + AB$ (1)



Implementation of Full adder using Two Half Adders



From above diagram

$C_{out} = AB + (A \oplus B) C_{in}$

$= AB + C_{in}(A\bar{B} + \bar{A}B)$

$= AB + A\bar{B}C_{in} + \bar{A}BC_{in}$

$= AB(C_{in}+1) + A\bar{B}C_{in} + \bar{A}BC_{in}$

$= ABC_{in} + AB + A\bar{B}C_{in} + \bar{A}BC_{in}$

$= AC_{in}(B+\bar{B}) + AB + \bar{A}BC_{in}$

$= AC_{in} + AB + \bar{A}BC_{in}$

$= AC_{in} + AB(C_{in}+1) + \bar{A}BC_{in}$

$= AC_{in} + ABC_{in} + AB + \bar{A}BC_{in}$

$= AC_{in} + \cancel{AB(1+C_{in}) + A}$

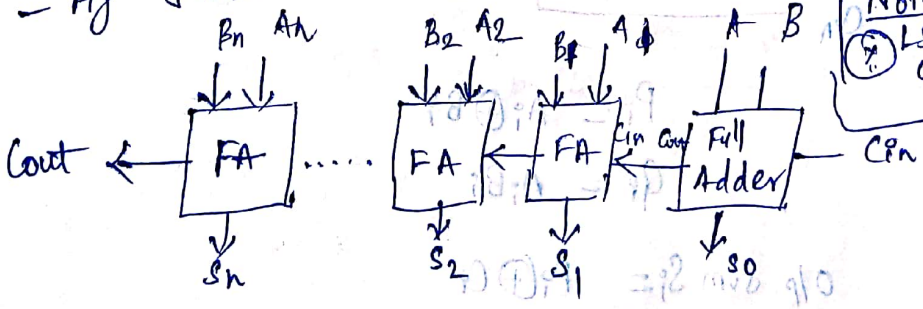
$+ AB + BC_{in}$ [A+A]

$C_{out} = AC_{in} + AB + BC_{in}$ — (II)

n - Bit Parallel Adder

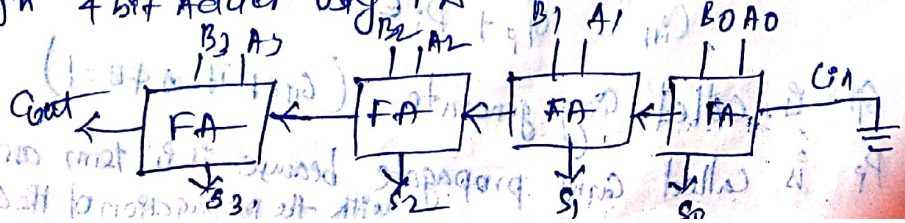
A single full adder is capable of adding two one bit numbers & an I/p carry. In order to Add binary numbers more than one bit additional full adders must be employed. A n bit parallel Adder can be constructed using number of full adder ckts Connected in ||el

- Fig shows the n bit ||el Adder

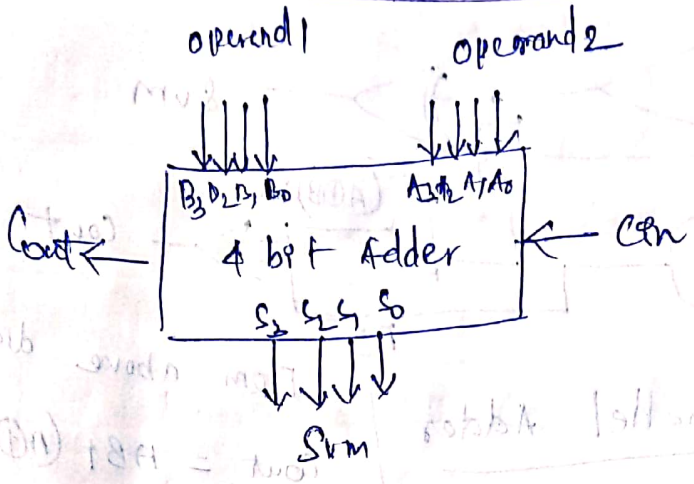


Note:
(i) LSB FA $C_{in} = 0$

Ex Design 4 bit Adder using FA.



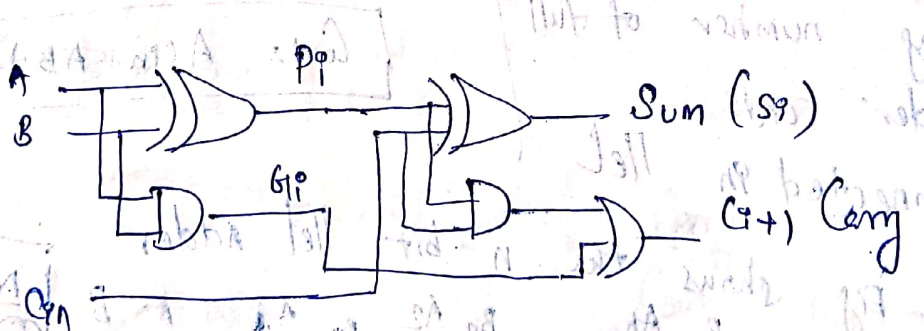
Logic diagram of 4 bit Ilel Adder



Look Ahead carry Generator

Speeding up this process by eliminating inter stage carry delay is called look ahead carry addition. This method utilizes the logic gates to look at the lower order bits of the augend and addend to see if a higher order carry is to be generated. It uses two functions : Carry generate & carry propagate

Consider the full adder ckt as shown



$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

O/p Sum $S_i = P_i \oplus C_i$

$$C_{i+1} = G_i + P_i C_i$$

Now the Boolean Expression for the carry o/p of Each stage can be written as follows.

$$C_2 = G_1 + P_1 C_1$$

$$C_3 = G_2 + P_2 C_2 = G_2 + (G_1 + P_1 C_1) P_2$$

$$= G_2 + P_2 G_1 + P_2 P_1 C_1$$

$$C_4 = G_3 + P_3 C_3$$

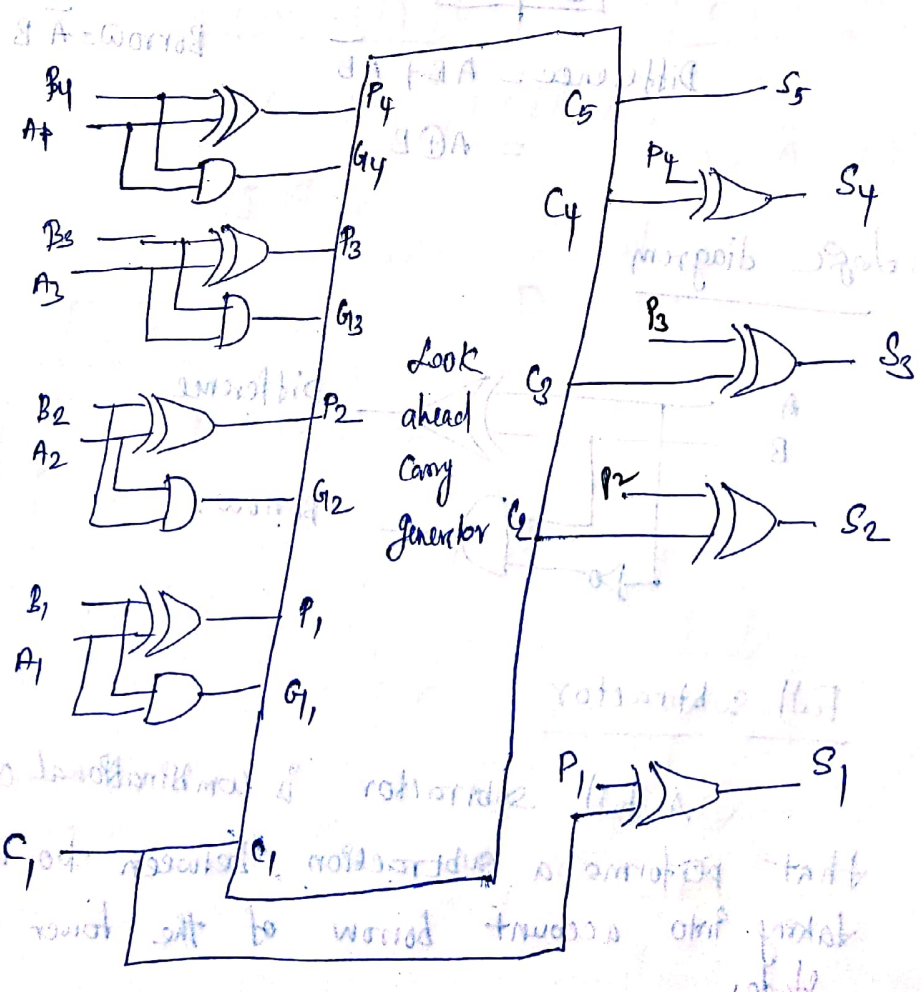
$$= G_3 + P_3 (G_2 + P_2 C_2)$$

$$= G_3 + P_3 G_2 + P_3 P_2 C_2 = G_3 + P_3 G_2 + P_3 P_2 [G_1 + P_1 C_1]$$

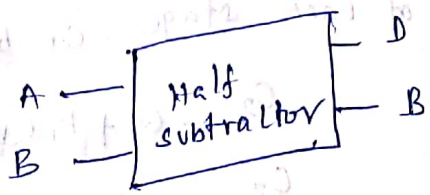
$$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1$$

From above Boolean Expression it can be seen than C_4 does not have to wait for C_3 & C_2 to propagate In fact C_4 is propagated at the same time C_2 C_3.

4 bit Parallel adder with look ahead carry generator



Subtractor (Half-subtractor)



Truth table

I/P		O/P	
A	B	Diff	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

K map

Difference

A \ B	0	1
0	0	1
1	1	0

Borrow

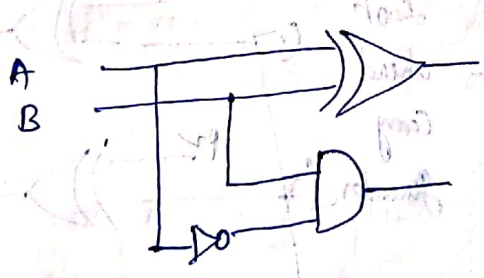
A \ B	0	1
0	0	1
1	0	0

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

$= A \oplus B$

Borrow = $\bar{A}B$.

Logic diagram



Full Subtractor

A full subtractor is combinational circuits that performs a subtraction between two bits, taking into account borrow of the lower significa

Inputs			outputs	
A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
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

Kmap D

A \ BBin	00	01	11	10
0	0	1	0	1
1	1	0	1	0

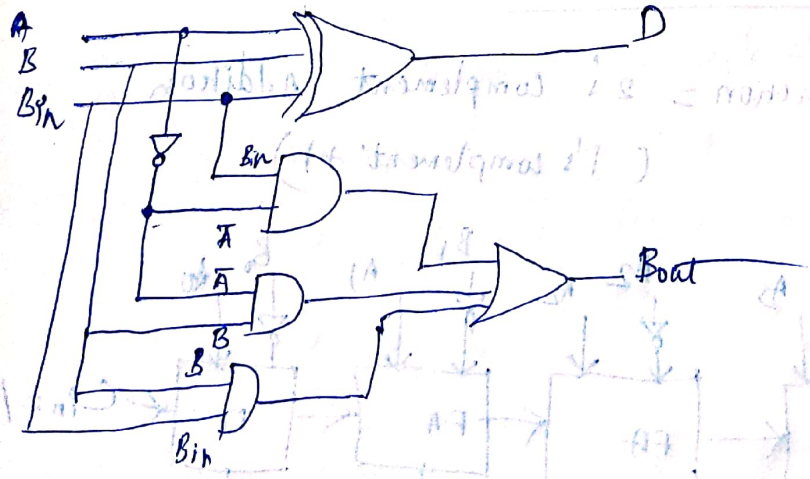
Bout

A \ BBin	00	01	11	10
0	0	1	1	1
1	0	0	1	0

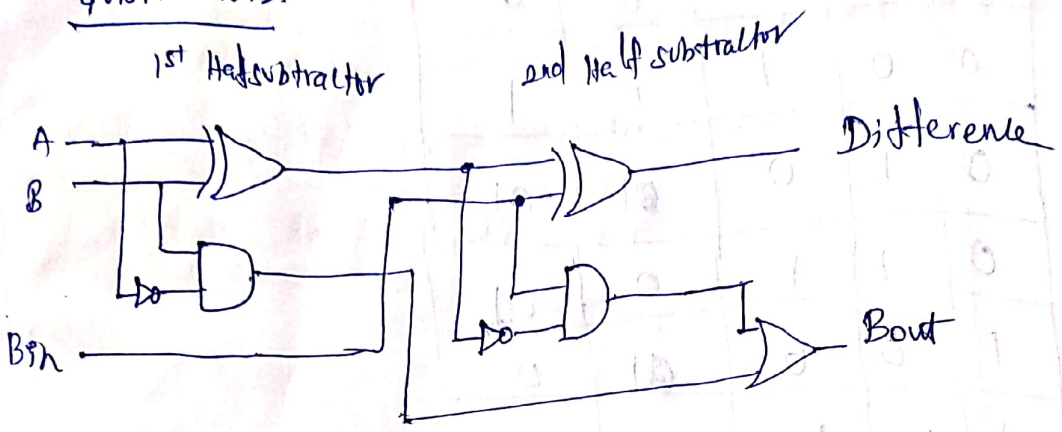
$D = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + ABB_{in}$

$= A \oplus B \oplus B_{in}$

$B_{out} = \bar{A}B_{in} + \bar{A}B + BB_{in}$



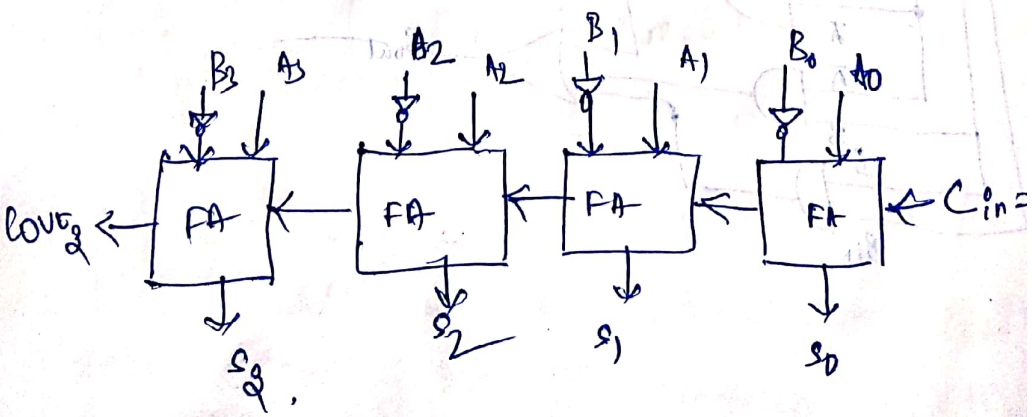
Implementation of full subtractor with two subtractors.



$$B_{out} = \bar{A}B + \overline{(A \oplus B)}\, B_{in}$$

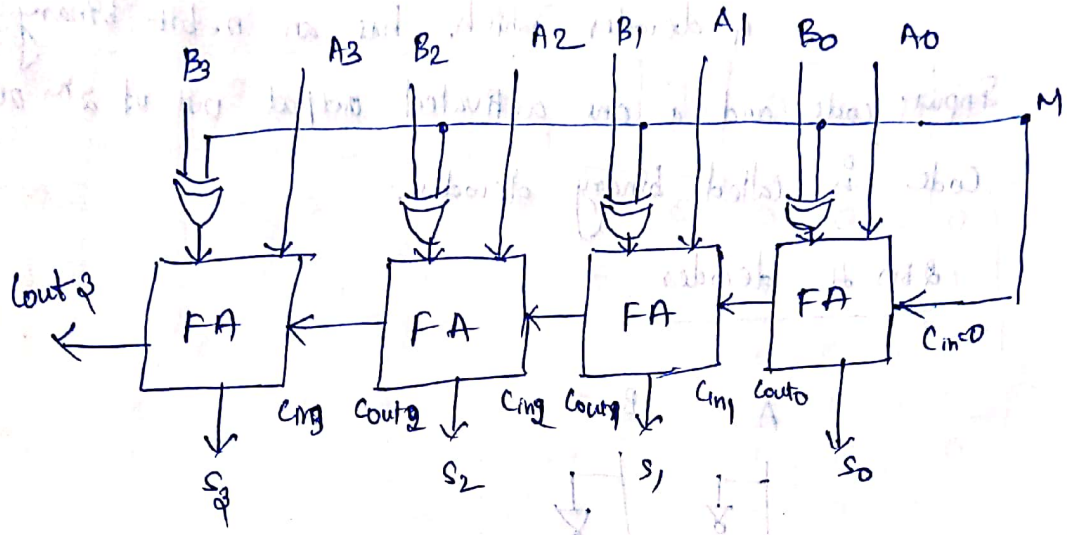
n Bit Parallel subtractor

Subtraction = 2's complement Addition
(1's complement + 1)



Binary Adder-subtractor

4 bit adder-subtractor.



The addition & subtraction operations can be combined into one circuit with one common binary adder

M controls the operation.

EXOR

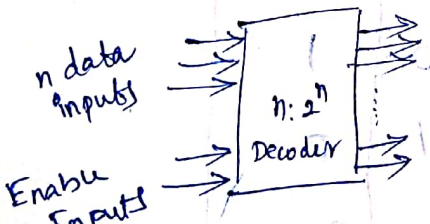
If M=0 circuit is adder ⇒ B⊕0 = B Cin=0

M=1 " " subtractor. B⊕1 = $\overline{B}$ Cin=1

(2's complement of B)

The ckts operates A−B.

Decoder



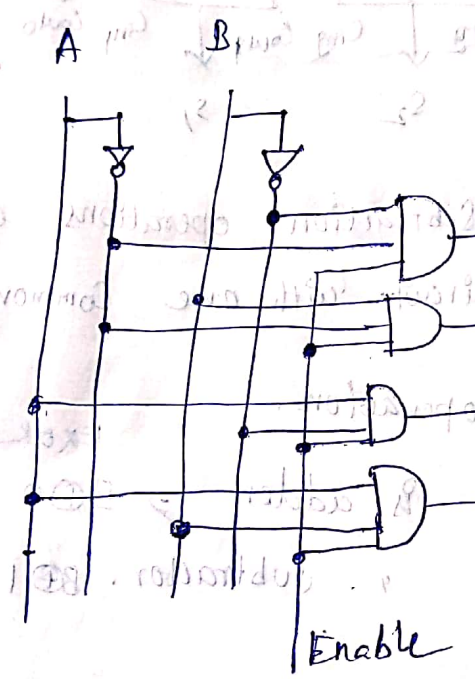
- A decoder is a multiple-input, multiple-output logic circuit which converts coded inputs into coded outputs where I/p & O/p codes are different.

- n inputs produced 2^n possible outputs.
- 2^n output values are 0 through 2^n-1.

Binary decoder

A decoder which has an n-bit binary input code and a one activated output out of 2^n output code is called binary decoder.

2 to 4 decoder

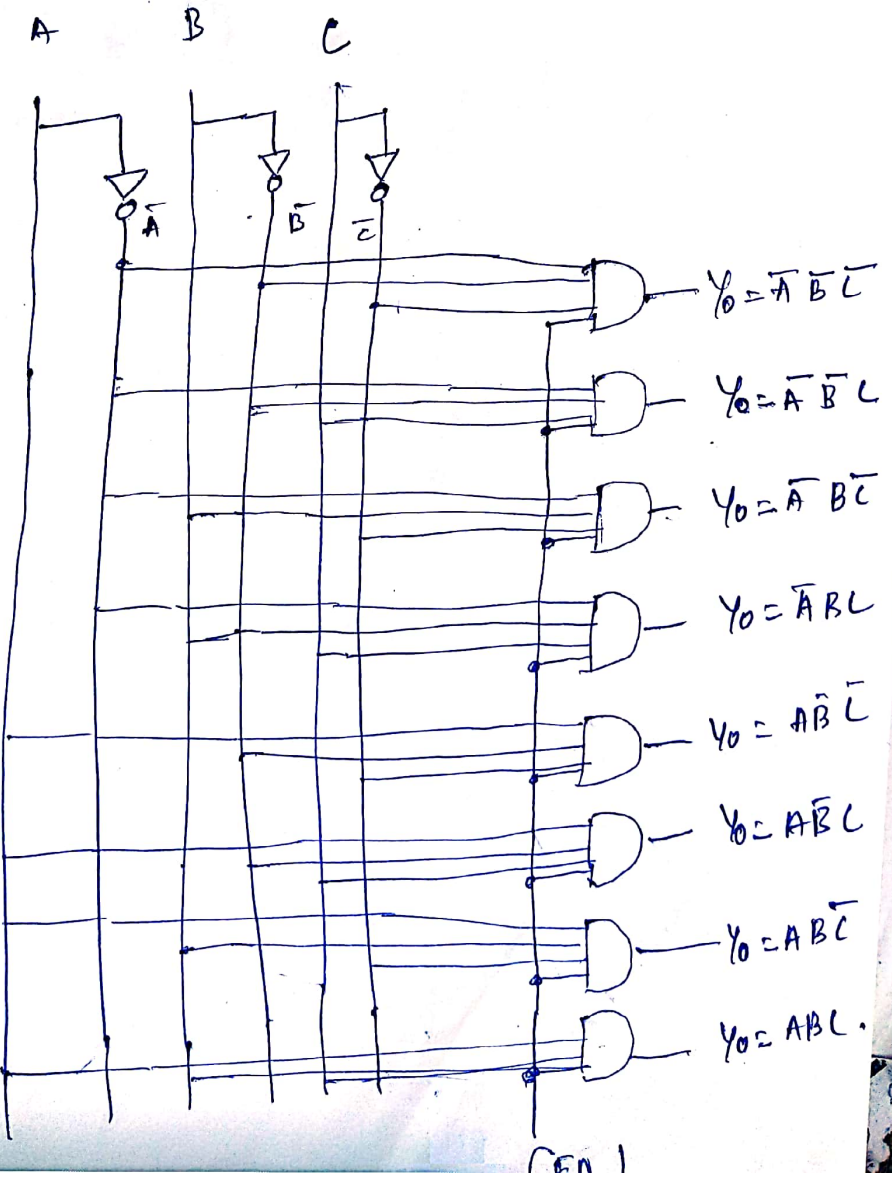


Truth table

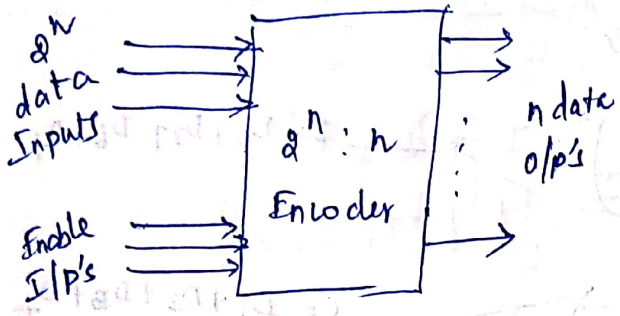
Inputs			Outputs			
EN	A	B	Y_3	Y_2	Y_1	Y_0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

(2) Draw the circuit for 3 to 8 decoder

Inputs				outputs							
EN	A	B	C	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0



Encoder



- A Encoder is a digital circuits that Performs Inverse operation of decoder.
2^N I/p lines to n o/p lines

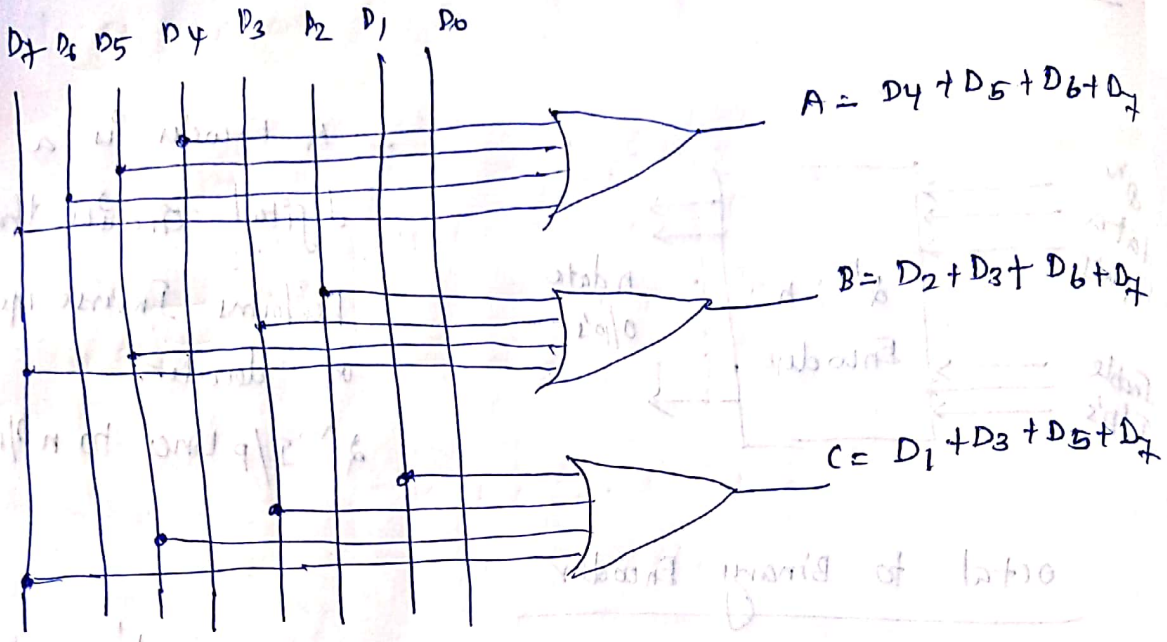
octal to Binary Encoder

It has 8 I/p's & 3 o/p's generate corresponding binary code. In Encoders It is assumed that only one input has a value of 1 at any given time otherwise Circuit is meaningless.

Truth table

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

A = 4+5+6+7 C = 1+3+5+7
B = 2+3+6+7

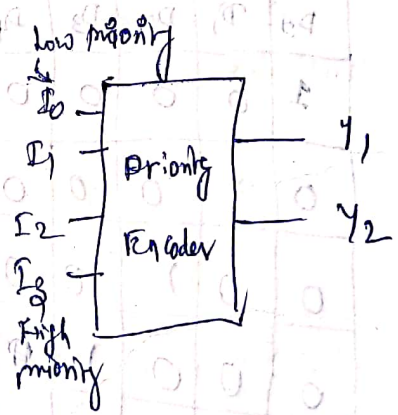


Priority Encoder

A priority Encoder is an Encoder circuit that includes the priority function. In priority Encoder if two or more inputs are equal to 1 at same time, the input having the highest priority will take precedence.

Truth table

I_3	I_2	I_1	I_0	Y_1	Y_0
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

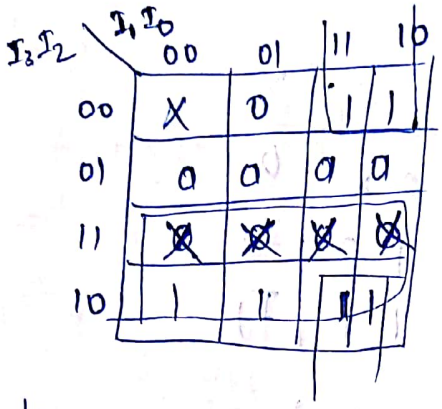


For Y_1

$I_3 I_2$ \ CD $I_1 I_0$	00	01	11	10
00	x	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

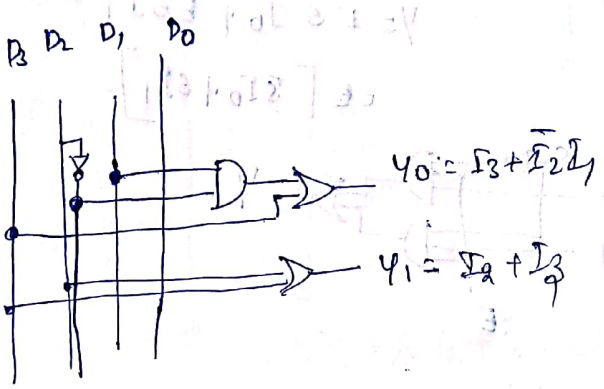
$Y_1 = I_2 + I_3$

For Y_0



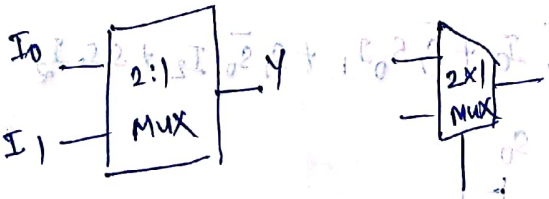
$$Y_0 = I_3 + I_2\bar{I}_1$$

diagram



Multiplexer (many to one) MUX

- Multiplexer is a digital switch.
- Several data-Input line to single output line by using selection lines.
- 2^n to n selection lines decide which Input going to select
- Data selector.



$n = 2^m$

↙ no. of i/puts

$\Rightarrow m \log_2^n$

$n = 4$

$m = \log_2^4$

$= 2\log_2^2 = 2.$

Adv

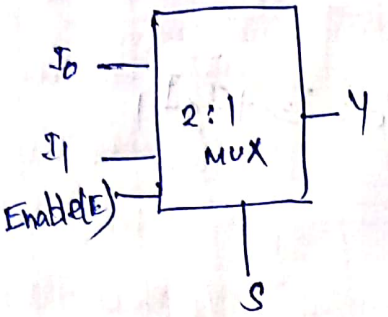
(i) It reduces no. of wires.

(ii) Reduces circuit complexity & cost.

(iii) Implementation of various circuit using Mux.

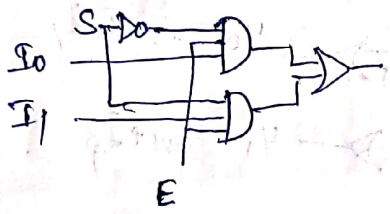
Types :→ 2:1 MUX, 4:1 mux 8:1 mux, 16:1 MUX, 32:1 MUX

2:1 MUX



E	S	Y
0	X	0
1	0	I_0
1	1	I_1

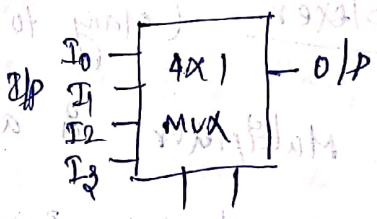
$$Y = E\bar{S}I_0 + ESI_1$$
$$= E[\bar{S}I_0 + SI_1]$$



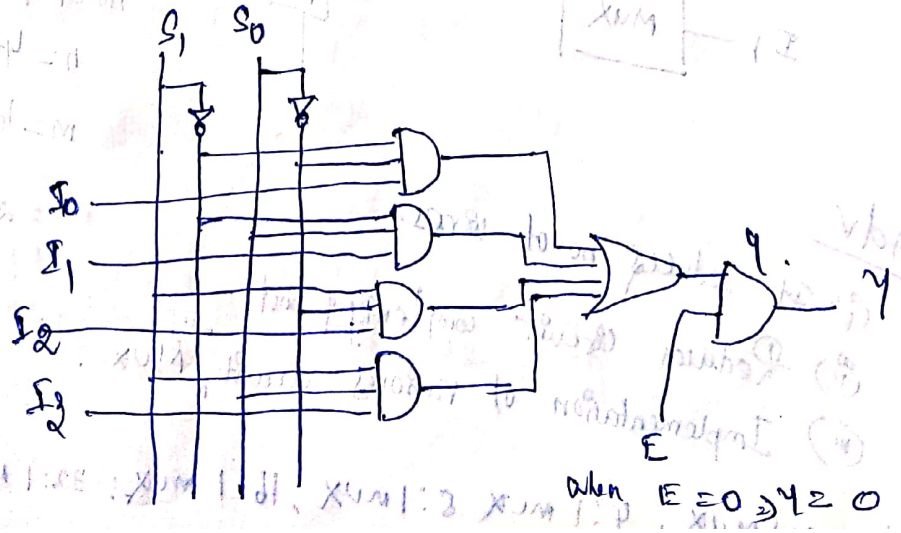
4x1 MUX

Truth table

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



$$Y = \bar{S}_1\bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1\bar{S}_0 I_2 + S_1 S_0 I_3$$

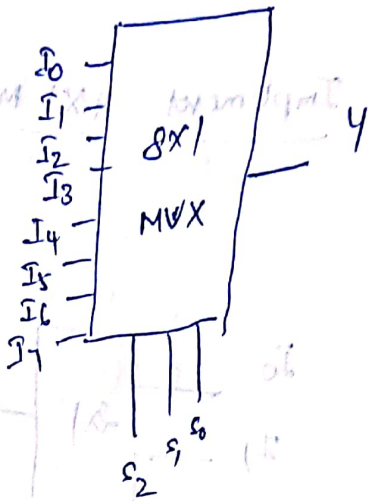


When E = 0 ⇒ Y = 0

8x1 MUX :

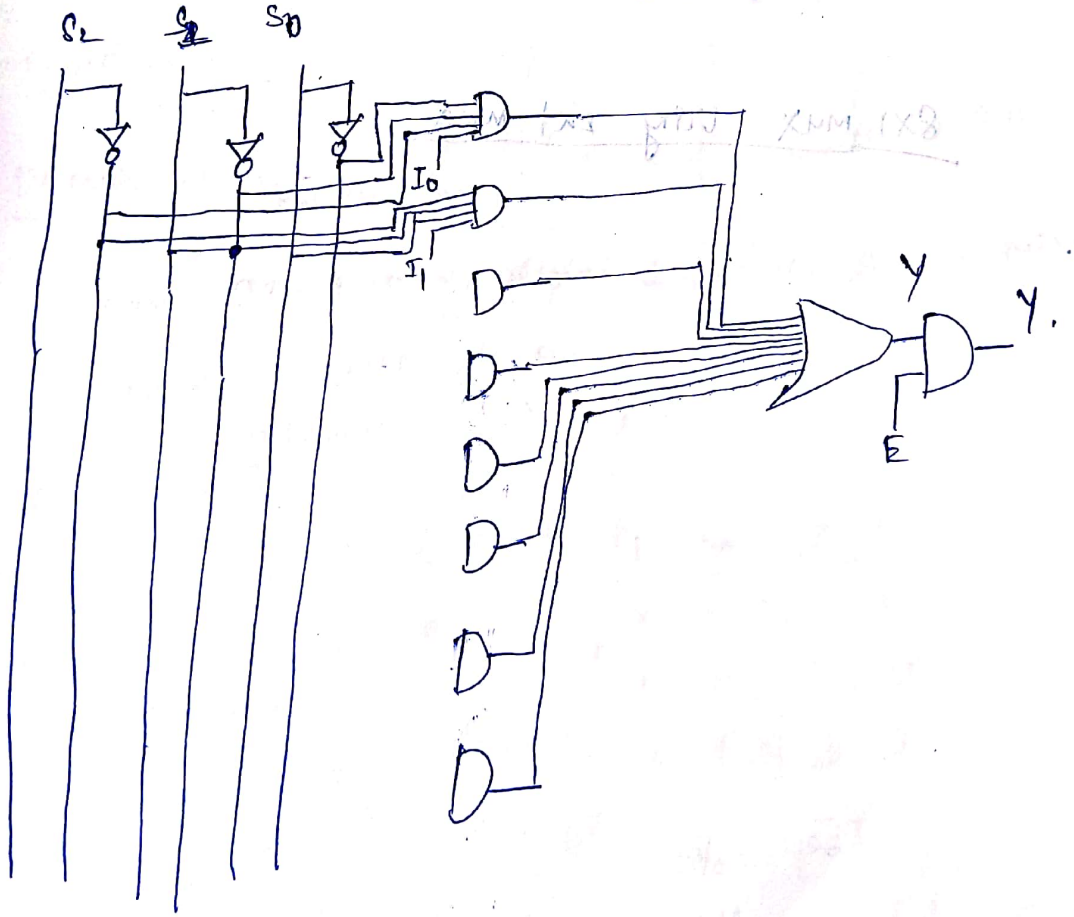
Truth table

S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7



Expression

$$Y = \bar{S}_2\bar{S}_1\bar{S}_0 I_0 + \bar{S}_2\bar{S}_1 S_0 I_1 + \bar{S}_2 S_1\bar{S}_0 I_2 + \bar{S}_2 S_1 S_0 I_3 + S_2\bar{S}_1\bar{S}_0 I_4 + S_2\bar{S}_1 S_0 I_5 + S_2 S_1\bar{S}_0 I_6 + S_2 S_1 S_0 I_7$$



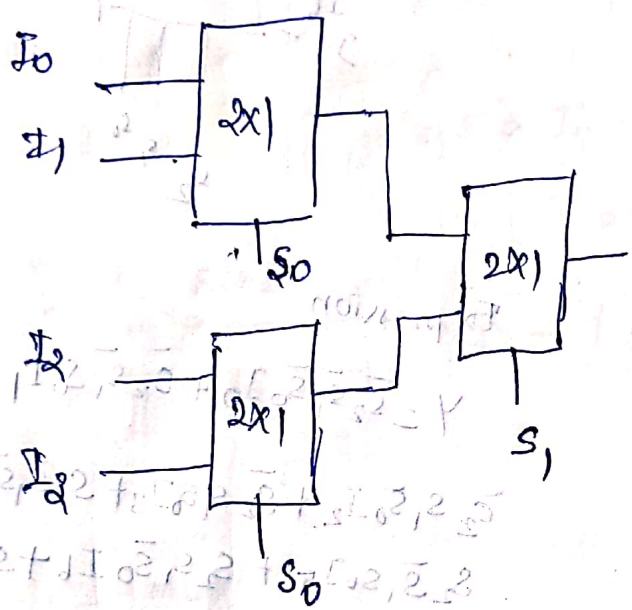
MUX Tree (obtaining higher order MUX using lower order MUX)

Implement 4×1 MUX using 2×1 MUX

$n = 4 \times 1$

$= 4/2 = (2)$

$\frac{2}{2} = (1)$ + (3) req MUX



Truth table

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

8×1 MUX using 2×1 MUX

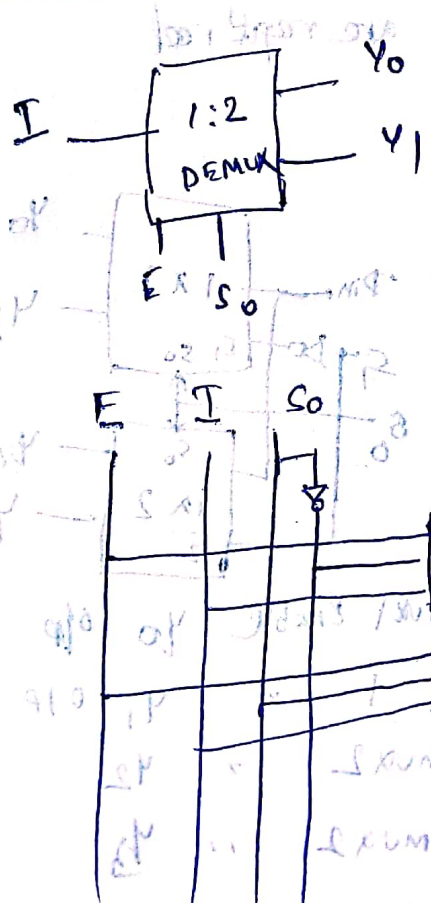
Demux (one to many)

O/P

$n = 2^m$ ← selection lines

- One I/p to many output depends on select Inputs.
- Reverse operation of MUX
- Data distributor

1:2 DEMUX



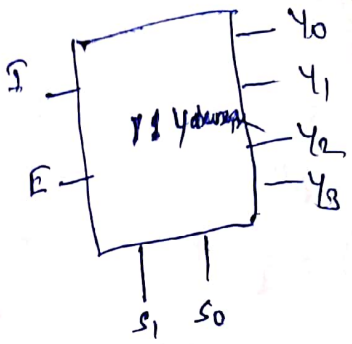
E	S_0	Y_0	Y_1
X	0	0	0
1	0	I	0
1	1	0	I

$Y_0 = E\bar{S}_0 I$

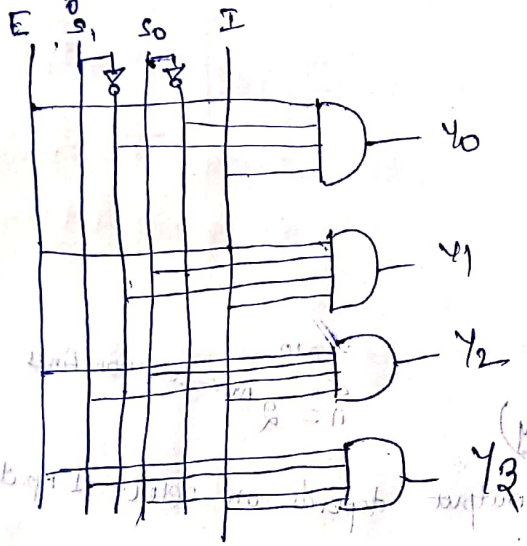
$Y_1 = E S_0 I$.

1:4 DEMUX

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



$Y_0 = E\bar{S}_1\bar{S}_0 I$ $Y_1 = E\bar{S}_1 S_0 I$ $Y_2 = E S_1 \bar{S}_0 I$ $Y_3 = E S_1 S_0 I$



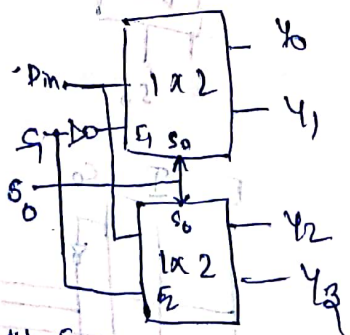
1:4 DEMUX using 1:2 MUX

2 no's of 1:2 demux are required

$S_1 S_0$ Select lines.

Y_0, Y_1, Y_2, Y_3 outputs

Din - Data input



working

Din Given

S_0	S_1	
0	0	MUX1 Enable Y_0 o/p
0	1	1 " Y_1 o/p
1	0	MUX2 " Y_2
1	1	MUX2 " Y_3

Code converters [Binary to Gray converter]

Binary code				Gray code				
D	C	B	A	G_3	G_2	G_1	G_0	
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	3
0	0	1	1	0	0	1	0	2
0	1	0	0	0	1	1	0	6
0	1	0	1	0	1	1	1	7
0	1	1	0	0	1	0	1	5
0	1	1	1	0	1	0	0	4
1	0	0	0	1	1	0	0	12
1	0	0	1	1	1	0	1	13
1	0	1	0	1	1	1	1	15
1	0	1	1	1	1	1	0	14
1	1	0	0	1	0	1	0	10
1	1	0	1	1	0	1	1	11
1	1	1	0	1	0	0	1	9
1	1	1	1	1	0	0	0	8

Kmap

G_3

DC \ BA	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$G_3 = D$

G_2

DC \ BA	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$G_2 = \bar{D}C + D\bar{C}$

$= D \oplus C$

G_1

DC \ BA	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

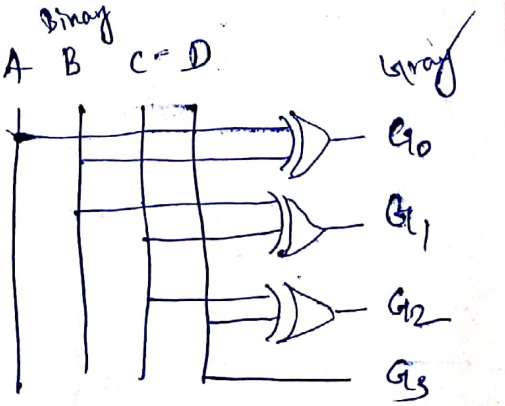
$G_1 = C\bar{B} + \bar{C}B$

$= C \oplus B = B \oplus C$

G_0

DC \ BA	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

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



Gray to Binary converter

Gray code				Binary code			
G_3	G_2	G_1	G_0	D	C	B	A
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

Kmap

For D

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$= G_3$

For C

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$C = \overline{G_3}G_2 + G_3\overline{G_2}$

$= G_3 \oplus G_2$

For B

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	1	1
10	1	1	0	0

G_3G_2 \ G_1G_0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

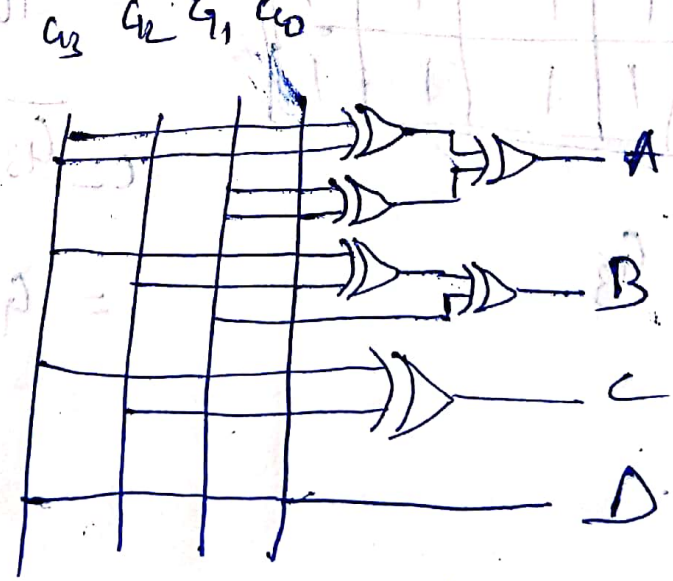
$$B = \bar{G}_3\bar{G}_2G_1 + \bar{G}_3G_2\bar{G}_0 + G_3G_2G_1 + G_3\bar{G}_2\bar{G}_0$$

$$= G_1(\bar{G}_3\bar{G}_2 + G_3G_2) + \bar{G}_0(\bar{G}_3G_2 + G_3\bar{G}_2)$$

$$= G_1(\overline{G_3 \oplus G_2}) + \bar{G}_0(G_3 \oplus G_2)$$

$$= G_1 \oplus G_2 \oplus G_3$$

$G_3 \oplus G_2 \oplus G_1 \oplus G_0$

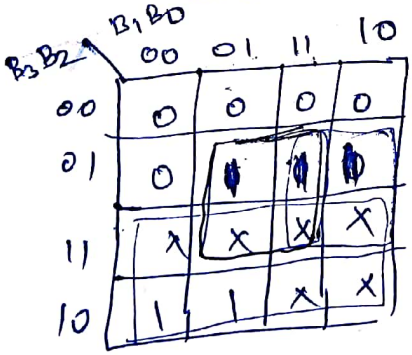


BCD to Excess 3 converter

Decimal	BCD code $B_3\ B_2\ B_1\ B_0$	Excess-3 code $E_3\ E_2\ E_1\ E_0$
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

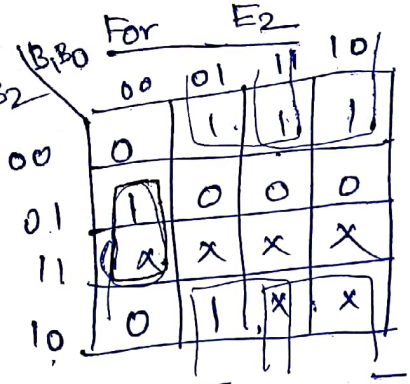
K map

For E_3



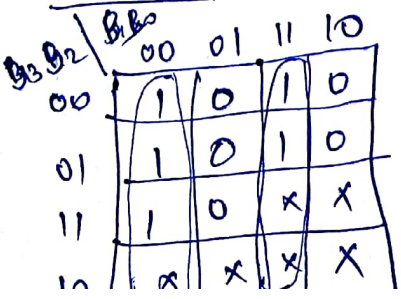
$B_3 + B_2\bar{B_0} + B_2\bar{B_1}$

For E_2



$\bar{B_2}B_1 + \bar{B_2}B_0 + B_2\bar{B_1}\bar{B_0}$

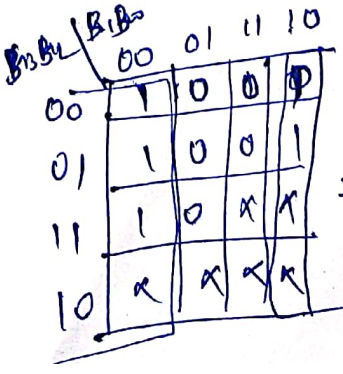
For E_1



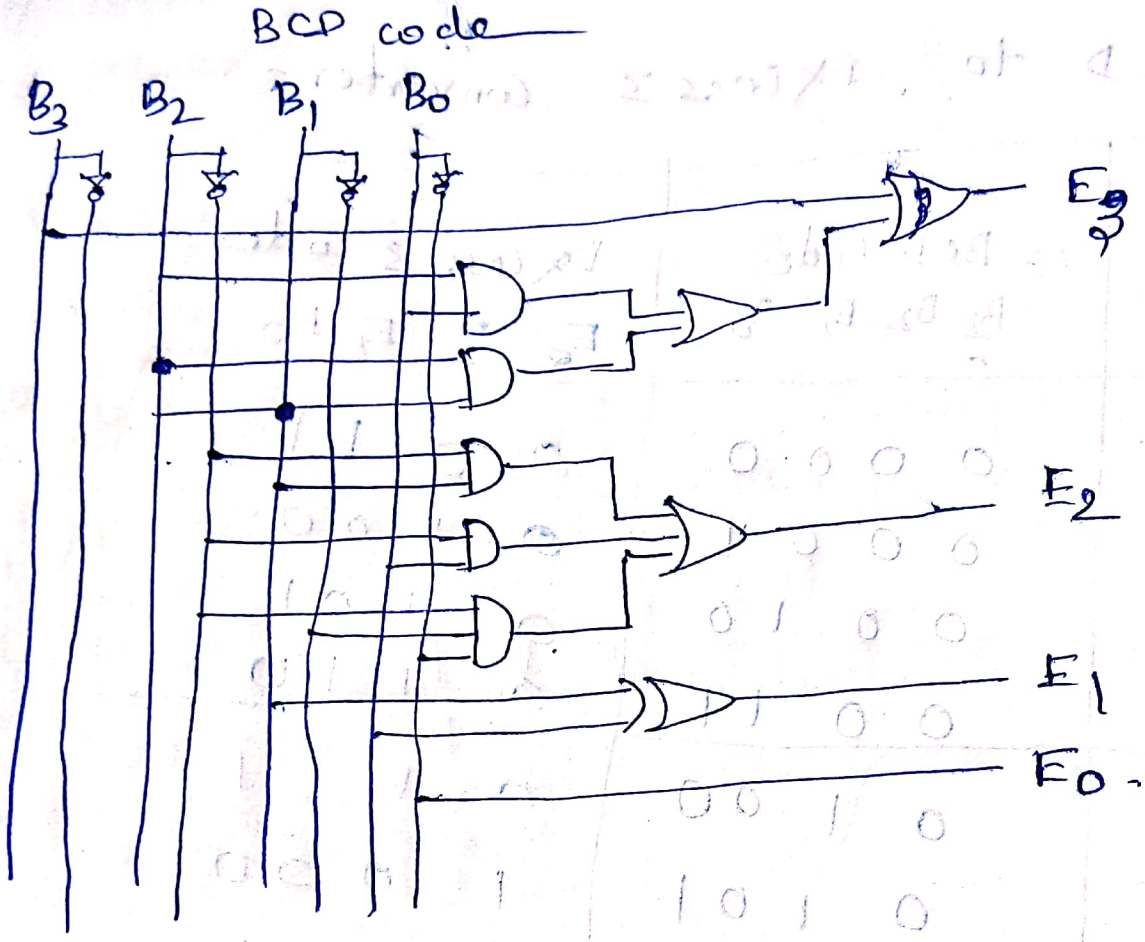
$= \bar{B_1}\bar{B_0} + B_1B_0$

$= B_1 \odot B_0$.

For E_0



$\Rightarrow \bar{B_0}$



Excess 3 Code