MODULE 4 : DESIGN AND ANALYSIS OF COMBINATIONAL CIRCUITS

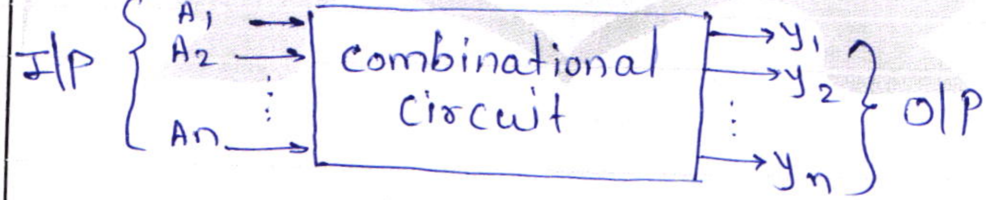
Introduction, Half and Full Adder, Half and Full Subtractor, Four Bit Binary Adder, One digit BCD Adder, code conversion, Encoder and Decoder ,Multiplexers and De- multiplexers, Decoders, Binary comparator (2,3 variable)4-bit Magnitude Comparator IC 7485 and ALU IC74181.

CHAPTER-4

Design & Analysis of Combinational Circuit

* Combinational logic ckt :-

- Do not use any memory.
- so, the previous state of i/p does not have any effect on the present state of the ckt.
- It is a logic ckt the o/p of which depends only on the combinational of the i/p's.



Block diagram of combinational ckt.

- It is made up of logic gates.
- It can have number of inputs & outputs.

Combinational ckt operates in three steps

1) It accepts n-different i/p's

2] The combination of gates operates on the i/p's.

3) "m" different o/p's are produced as per requirement.

Examples

→ Adders, subtractors
→ Comparator
→ Code converters
→ Encoders, Decoders
→ Multiplexers, Demultiplexers.

★ Design combinational logic using statements.

1] Design a combinational logic ckt that will allow i/p signal A to pass through to the o/p only when the control i/p B & C are same otherwise the o/p is high.

- Step I

I/P → A

control i/p → B & C

step II Draw the truthtable

I/P A	control I/P B	control I/P C	output y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

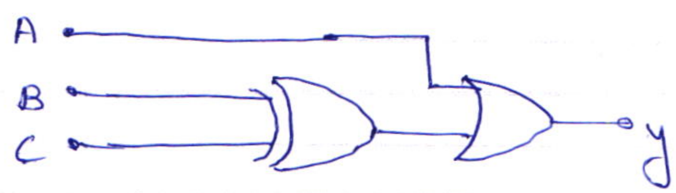
step III Write k-map for each o/p & get simplified expression

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

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

$= A + B \oplus C$

step 4 - Combinational ckt



* Binary Adders

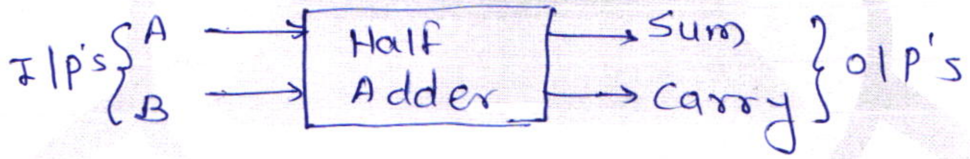
Types

1] Half Adder

2] Full Adder

I] Half Adder

- It has two i/p's & 2 o/p's.
- This ckt has two o/p's namely carry & sum.



Block Diagram

Truth table

A	B	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

k-map

Sum

A \ B	0	1
0		1
1	1	

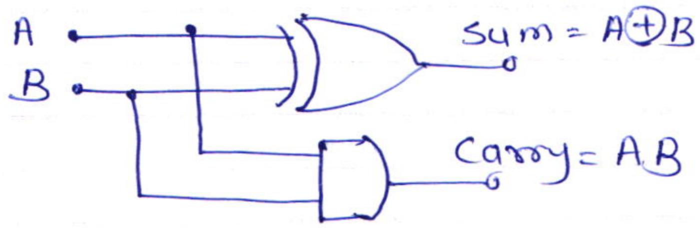
$\boxed{sum = A \oplus B}$

Carry

A \ B	0	1
0		
1		1

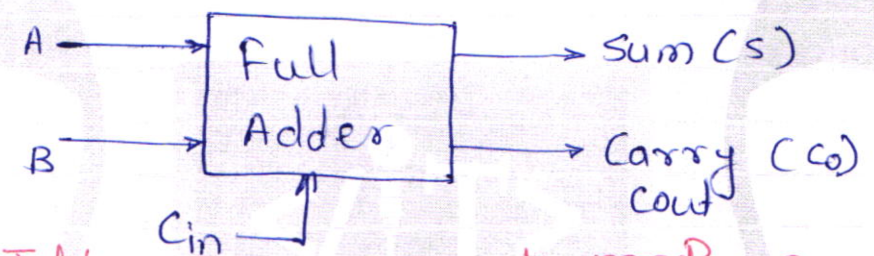
$\boxed{Carry = AB}$

Circuit



II] Full Adder :

- A three single bit adder ckt called Full adder.
- Two i/p's & carry $C_{in} \Rightarrow$ 3 i/p's
- Three i/p's & two o/p combinational ckt.



Truth Table

Inputs			outputs	
A	B	Cin	S	Co
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

K-map - sum

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

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

$S = C_{in}(\underbrace{\bar{A}\bar{B} + AB}_{\text{EX-NOR}}) + \bar{C}_{in}(\underbrace{\bar{A}B + A\bar{B}}_{\text{EX-OR}})$

$=$

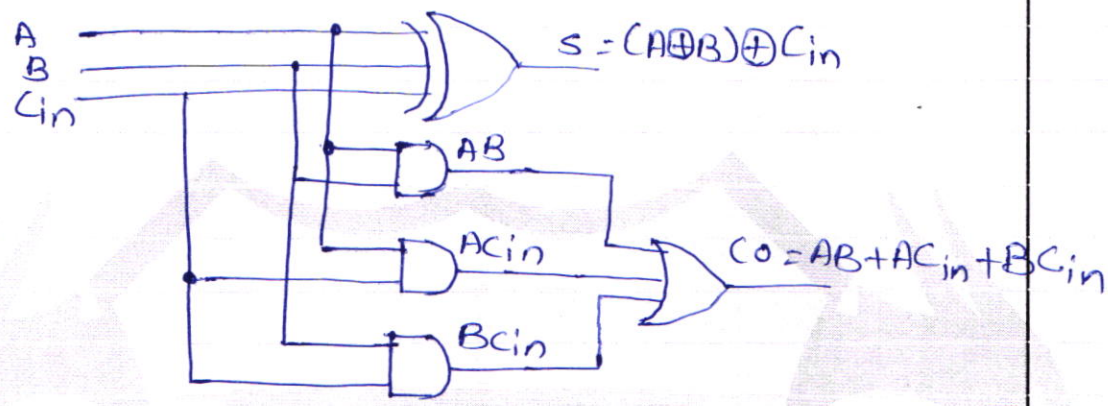
$\boxed{S = C_{in} \oplus A \oplus B}$

K-map - Carry $\rightarrow$ $\cancel{C_0 = AB}$

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

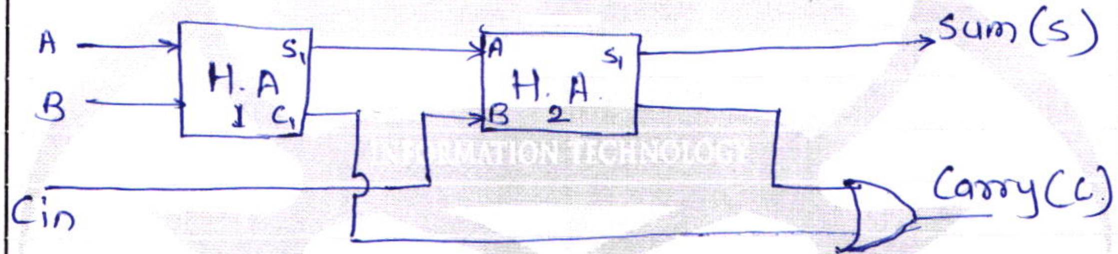
$\boxed{C_0 = AB + AC_{in} + BC_{in}}$

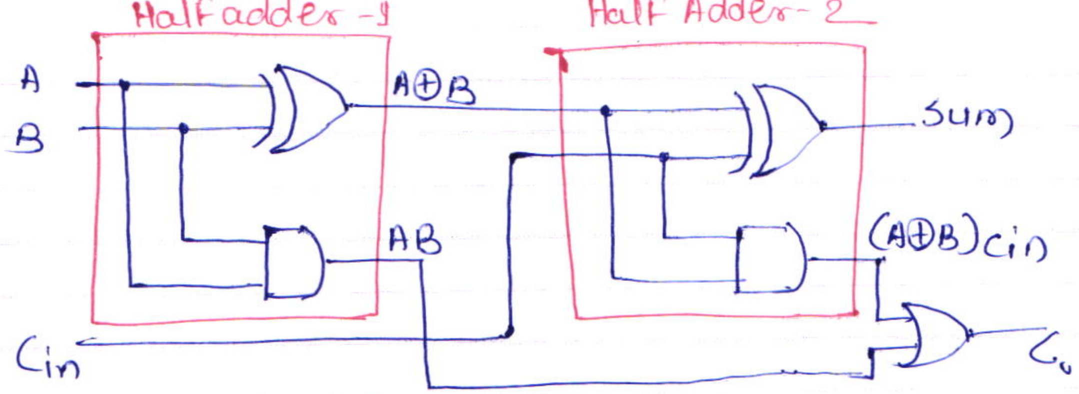
Logic diagram for Full Adder



* Design Full Adder using half adder

- Full adder ckt can be constructed b. using two half adders





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

$$\boxed{C_o = AB + BC_{in} + AC_{in}}$$

* Binary subtractors

Types

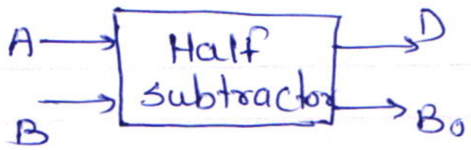
1] Half subtractor
2] Full "

I] Half subtractor:

- with two inputs & two outputs
- outputs namely difference & borrow
- subtraction (A-B)

where A → minuend bit
B → subtrahend bit

ckt diagram



Truth Table

I/P's		outputs	
A	B	D	B_o
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

~~kam~~

K-map For difference (D)

A \ B	0	1
0		1
1	1	

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

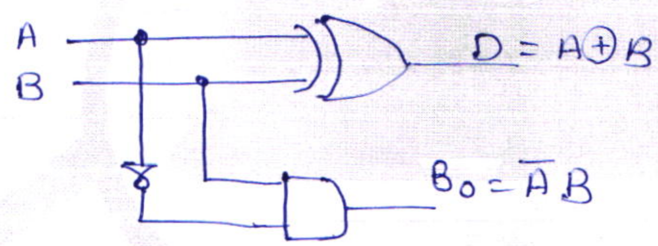
$\boxed{D = A \oplus B}$

K-map For Borrow (B_o)

A \ B	0	1
0		1
1		

$B_o = \bar{A}B$

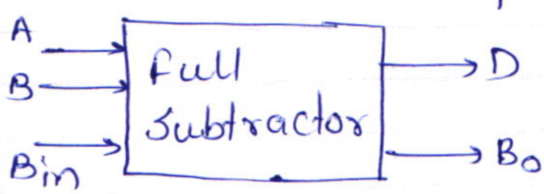
Logic diagram



II) Full subtractor :—

- It is combinational ckt with three inputs A, B & Bin & two outputs D & B_o
- $(A - B - B_{in})$

where $A \rightarrow$ minuend bit
$B \rightarrow$ subtrahend bit
$B_{in} \rightarrow$ borrow produced by previous stage



$\Longrightarrow$ Circuit diagram

Truthtable

Inputs			outputs	
A	B	B_{in}	D	B_0
minuend	subtrahend	previous borrow	$(A-B-B_{in})$	
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

K-map For Difference (D)

A \ BB_{in}	00	01	11	10
0		1		1
1	1		1	

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

K-map For Borrow (B_0)

A \ BB_{in}	00	01	11	10
0		1	1	1
1			1	

$$\boxed{B_0 = \bar{A}B_{in} + \bar{A}B + BB_{in}}$$

Logic diagram For full subtractor

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

$= B_{in}\underbrace{(\bar{A}\bar{B} + AB)}_{\text{EX-NOR}} + \bar{B}_{in}\underbrace{(\bar{A}B + A\bar{B})}_{\text{EX-OR}}$

$D = B_{in}(\overline{A \oplus B}) + \bar{B}_{in}(A \oplus B)$

Let $\boxed{A \oplus B = C}$

$= B_{in}(\bar{C}) + \bar{B}_{in}(C)$

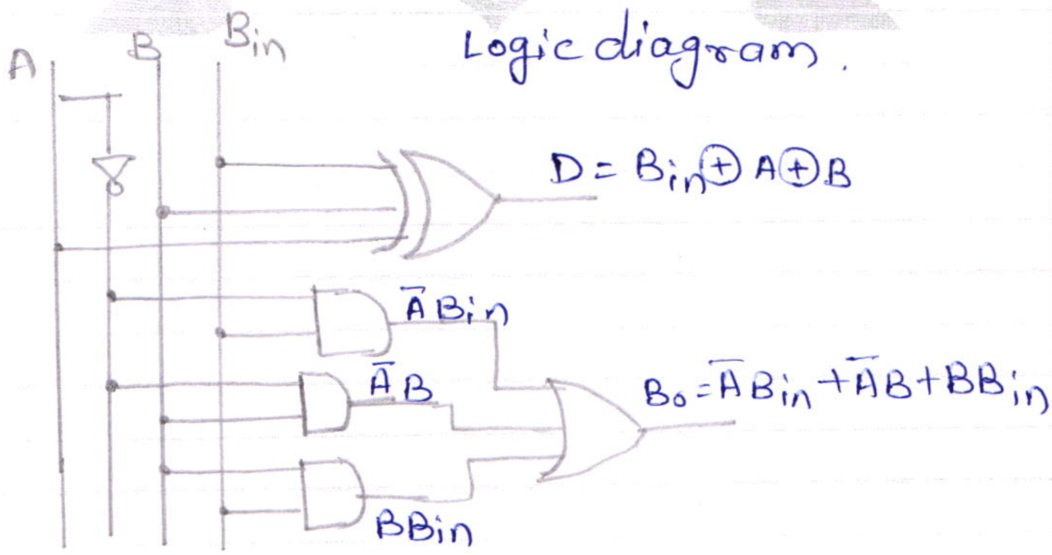
$= B_{in}\bar{C} + \bar{B}_{in}C$

$= \cancel{B_{in}(\bar{C}+C)} = B_{in} \oplus C$

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

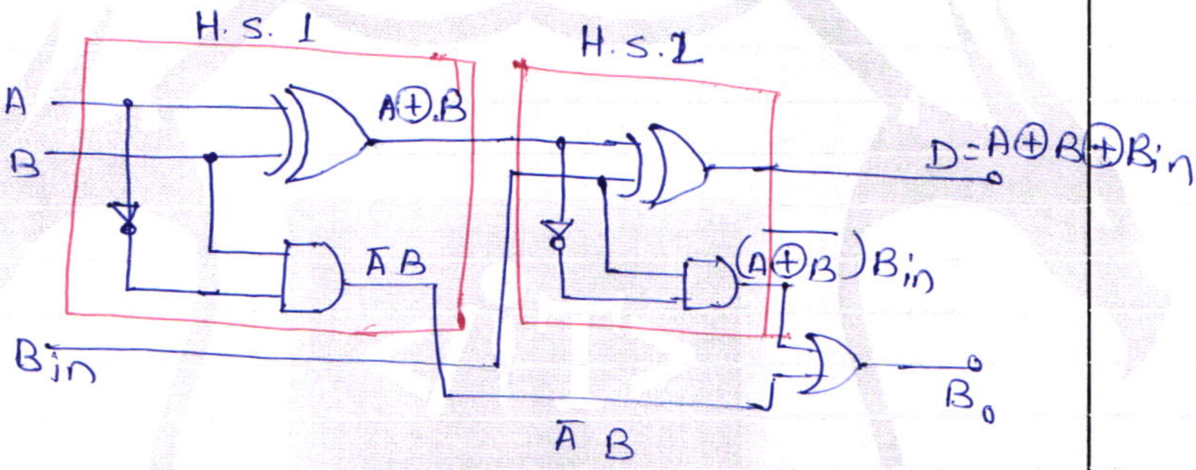
$\boxed{B_o = \bar{A}B_{in} + \bar{A}B + BB_{in}}$

Logic diagram.



* Full subtractor using Half subtractors

Implementation of full subtractor using two half subtractors & OR gate.



* Code Converters :-

I] Binary to BCD code Converter

- We will convert 4-bit binary code into BCD code.

Truth table

Binary I/P				BCD O/P				
A	B	C	D	B_4	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	1	1
0	1	0	0	0	0	1	0	0
0	1	0	1	0	0	1	0	1
0	1	1	0	0	0	1	1	0
0	1	1	1	0	0	1	1	1
1	0	0	0	0	1	0	0	0
1	0	0	1	0	1	0	0	1
1	0	1	0	1	0	0	0	0
1	0	1	1	1	0	0	0	1
1	1	0	0	1	0	0	1	0
1	1	0	1	1	0	0	1	1
1	1	1	0	1	0	1	0	0
1	1	1	1	1	0	1	0	1

K-map simplification for each O/P

For B_4

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

$B_4 = AB + AC$

$\boxed{B_4 = A(B+C)}$

For B_3

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

$\boxed{B_3 = A\bar{B}\bar{C}}$

For B_2

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

$B_2 = \bar{A}B + BC$

$\boxed{B_2 = B(\bar{A}+C)}$

For B_1

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

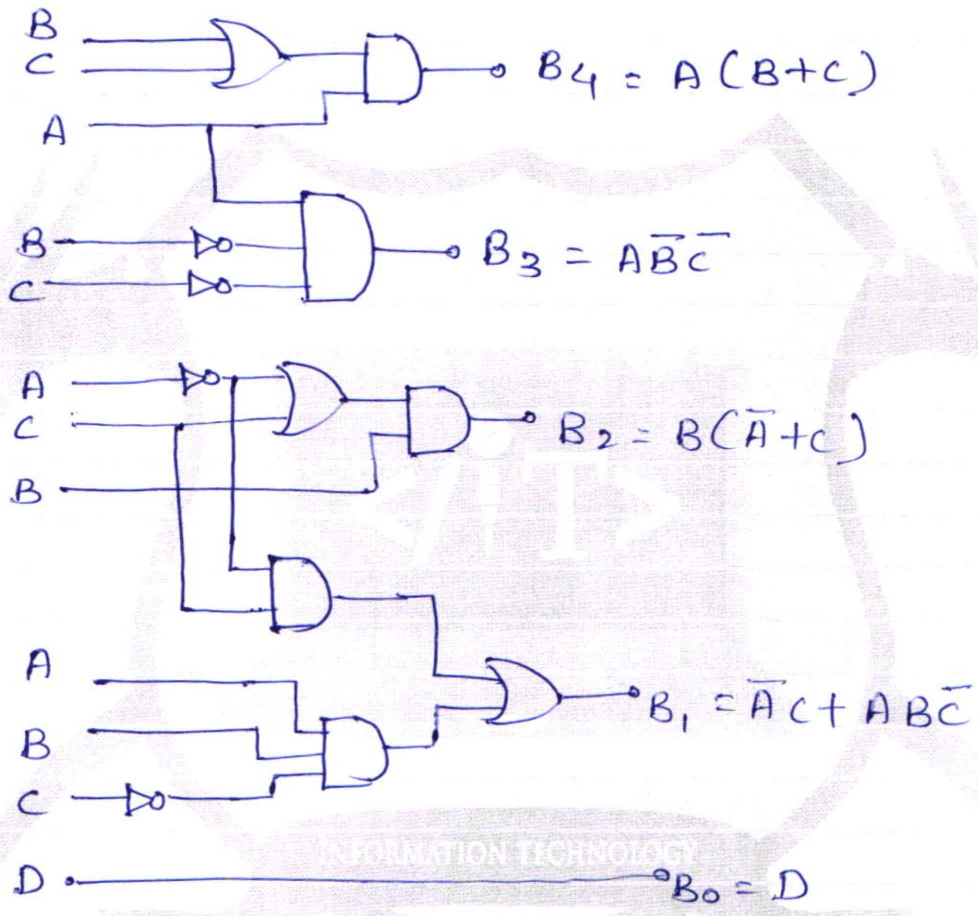
$\boxed{B_1 = \bar{A}C + AB\bar{C}}$

For B_0

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

$\boxed{B_0 = D}$

Combinational ckt



II] BCD to Gray code converter :-

- BCD represents 0 to 9 & 10 to 15 are don't care. Truth-Table

Decimal	BCD I/P's D_3	D_2	D_1	D_0	Gray O/P's G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1

with the help of EX-OR table

Step II prepare K-map

For G_3

D_3D_2 \ D_1D_0	00	01	11	10
00				
01				
11	X	X	X	X
10	1	1	X	X

$G_3 = D_3$

For G_2

D_3D_2 \ D_1D_0	00	01	11	10
00				
01	1	1	1	1
11	X	X	X	X
10	1	1	X	X

$G_2 = D_2 + D_3$

For G_1

$D_3D_2 \backslash D_1D_0$

		1	1
1	1		
X	X	X	X
		X	X

For G_0

$D_3D_2 \backslash D_1D_0$

	1		1
	1		1
X	X	X	X
	1		X

$G_1 = D_2\overline{D}_1 + \overline{D}_2 D_1$

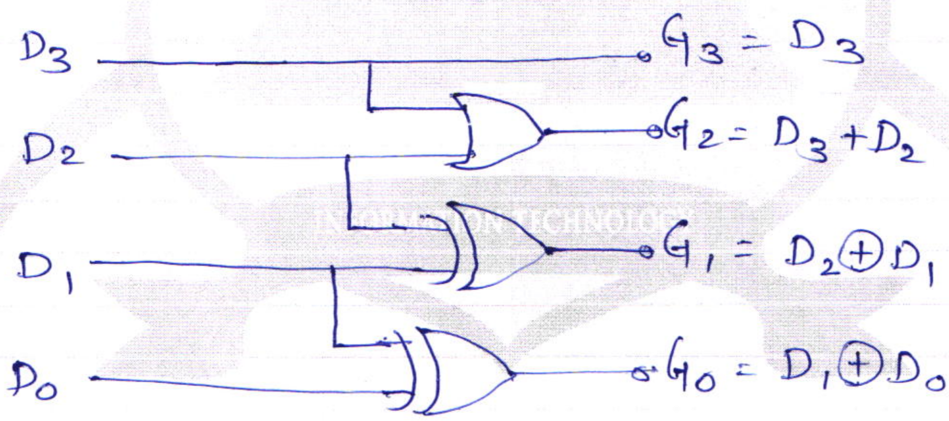
$= D_2 \oplus D_1$

$G_0 = (\overline{D}_1 D_0 + D_1\overline{D}_0)$

$= D_1 \oplus D_0$

Step III Combinational ckt

BCD to gray converter



III] Gray to BCD code converter

Convert 4-bit gray code into corresponding BCD code, show thruth table & implement using gates.

Step I: Truth table

Decimal	Grey input				BCD output			
	G_3	G_2	G_1	G_0	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1
8	1	1	0	0	1	0	0	0
9	1	1	0	1	1	0	0	1

Step II → K-map

For D_3

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

$D_3 = G_3$

For D_2

G_3G_2 \ G_1G_0				
	1	1	1	1
	X	X	X	X
			X	X

$D_2 = G_2$

For D_1

$G_3G_2 \backslash G_1G_0$	00	01	11	10
00			1	1
01			1	1
11	X	X	X	X
10			X	X

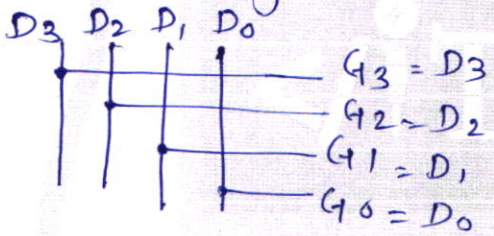
$D_1 = G_1$

For D_0

$G_3G_2 \backslash G_1G_0$	00	01	11	10
00		1	1	
01		1	1	
11	X	X	X	X
10		1	X	X

$D_0 = G_0$

Step III Gray to BCD converter



IV] BCD to Excess 3 code converter

- Adding 3 (0011) in BCD number

Step I Truth table

Decimal	BCD I/P's				Excess - 3 O/P's			
	D_3	D_2	D_1	D_0	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

Step II - K-map

For E3

$D_3D_2 \backslash D_1D_0$	00	01	11	10
00				
01		1	1	1
11	X	X	X	X
10	1	1	X	X

For E2

$D_3D_2 \backslash D_1D_0$	00	01	11	10
00		1	1	1
01	1			
11	X	X	X	X
10		1	X	X

$E3 = D_3 + D_2D_0 + D_2D_1$

$\boxed{E3 = D_3 + D_2(D_0 + D_1)}$

$E2 = \overline{D}_2D_1 + \overline{D}_2D_0 + D_2\overline{D}_1\overline{D}_0$

$\boxed{E2 = \overline{D}_2(D_1 + D_0) + D_2\overline{D}_1\overline{D}_0}$

For E1

$D_3D_2 \backslash D_1D_0$	00	01	11	10
00	1		1	
01	1		1	
11	X	X	X	X
10	1		X	X

For E0

$D_3D_2 \backslash D_1D_0$	00	01	11	10
00	1			1
01	1			1
11	X	X	X	X
10	1		X	X

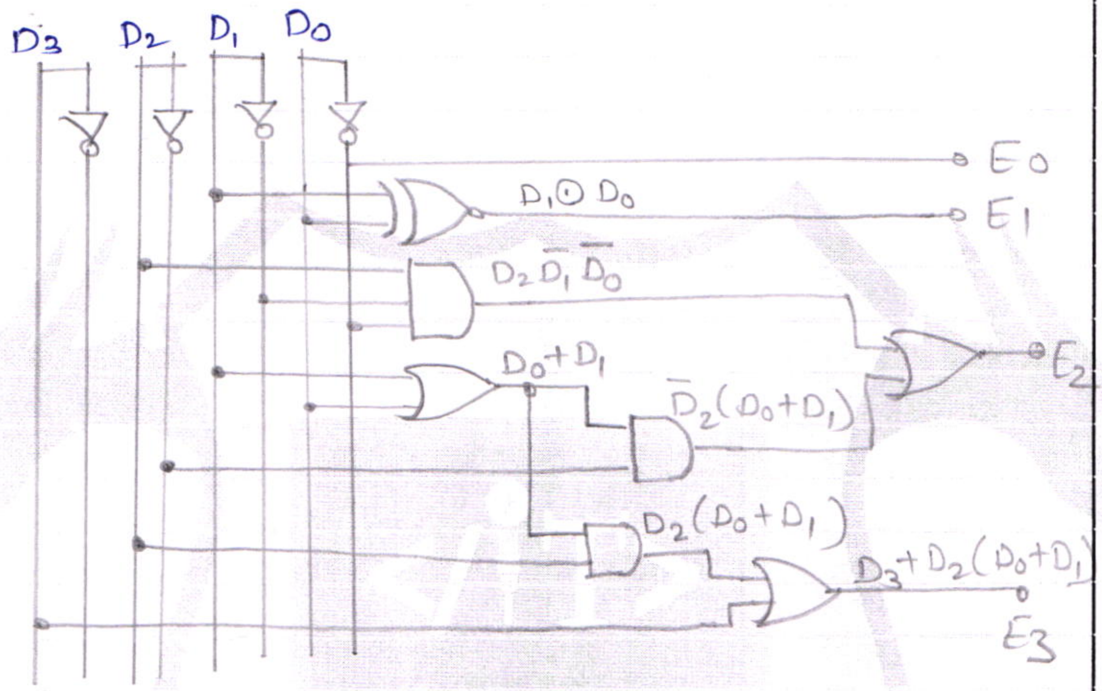
$E_1 = \overline{D}_1\overline{D}_0 + D_1D_0$

$\boxed{E_1 = D_1 \odot D_0}$

(Ex-NOR)

$\boxed{E_0 = \overline{D}_0}$

Step III -

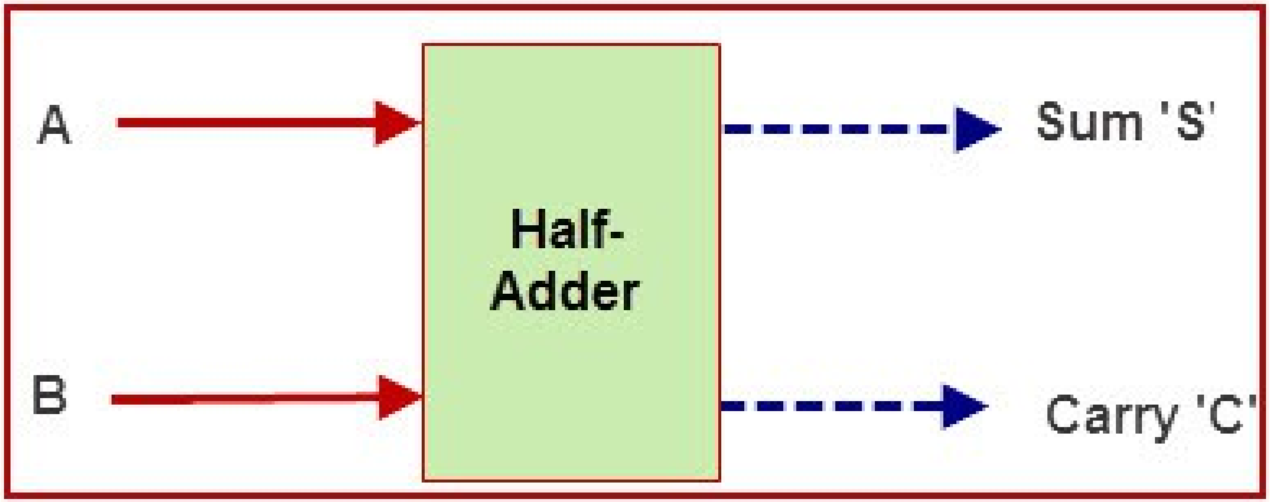


BCD to Excess-3 code converter.

V] Binary to Gray Code Converter

VI] Gray to Binary Code Converter

Half Adder



Half Adder Truth Table

INPUTS		OUTPUTS	
A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

1-bit adder can be easily implemented with the help of the XOR Gate for the output ‘SUM’ and an AND Gate for the ‘Carry’.

K-map for sum and carry

Truth Table		
A	B	SUM
0	0	0
0	1	1
1	0	1
1	1	0

A \ B	0	1
0	0	1
1	1	0

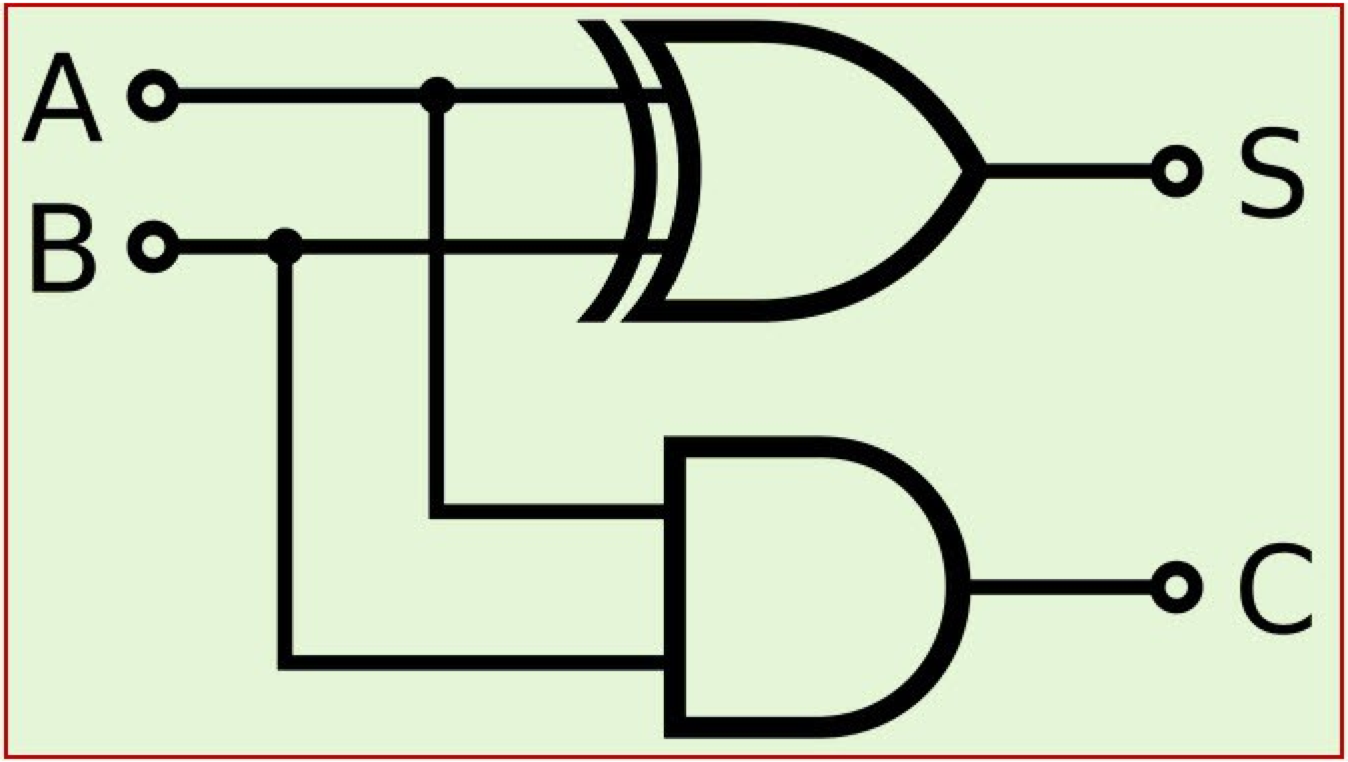
$\text{Sum} = A\bar{B} + \bar{A}B.$

Truth Table		
A	B	Carry
0	0	0
0	1	0
1	0	0
1	1	1

A \ B	0	1
0	0	0
1	0	1

Carry = AB

Half Adder Logic Diagram



Truth Table for Full Adder

Input			Output	
A	B	Cin	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

K-map-for sum

A \ BC_{IN}	00	01	11	10
0	0	1	0	1
1	1	0	1	0

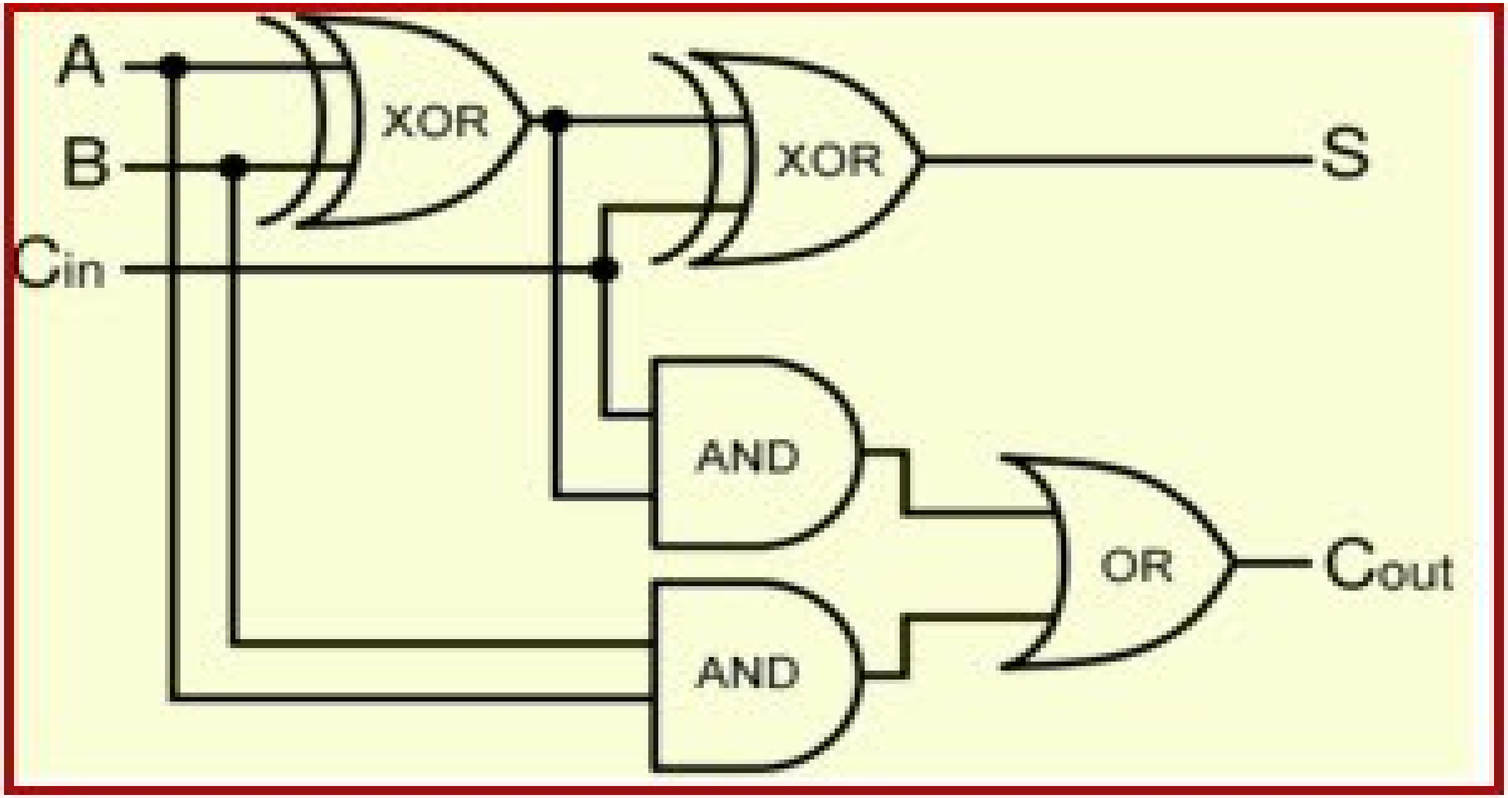
The simplified equation for sum is $S = \overline{A}\,\overline{B}C_{in} + \overline{A}B\overline{C}_{in} + ABC_{in}$

K-map-For Carry

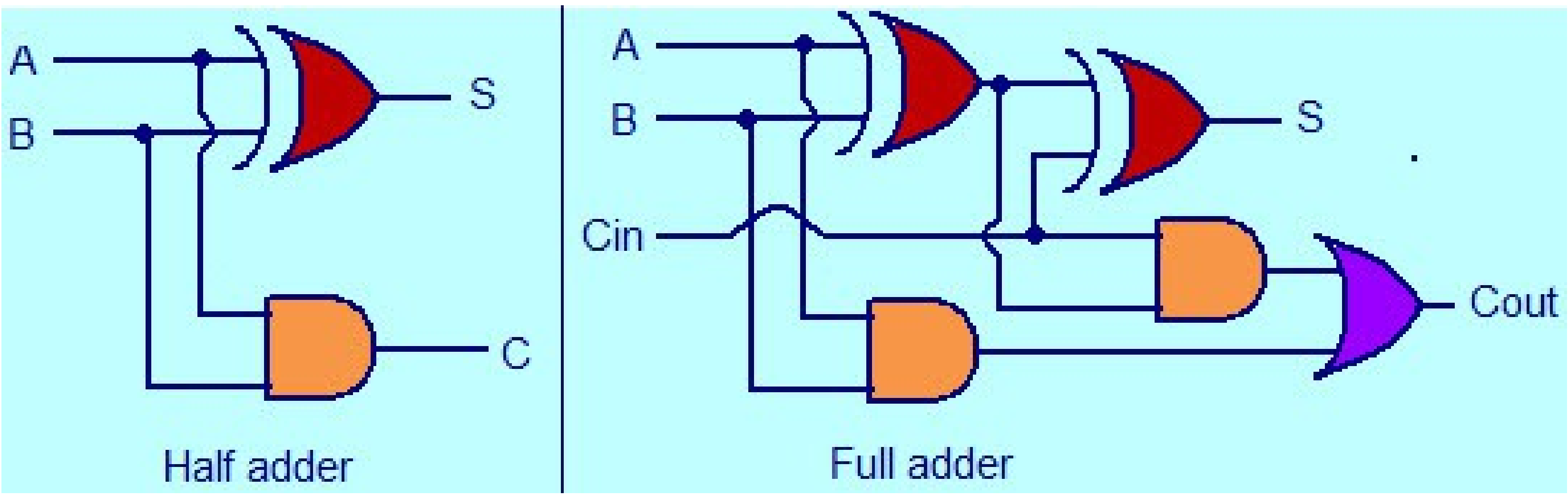
A \ BC_{IN}	00	01	11	10
0	0	0	1	0
1	0	1	1	1

The simplified equation for COUT is COUT = AB + ACIN + BCIN

Full Adder Logic Circuit



Adders Are Classified Into Two Types



Binary Subtractor

Subtractor :

Subtractor is the one which used to subtract two binary number(digit) and provides Difference and Borrow as a output.In digital electronics we have two types of subtractor.

1) Half Subtractor

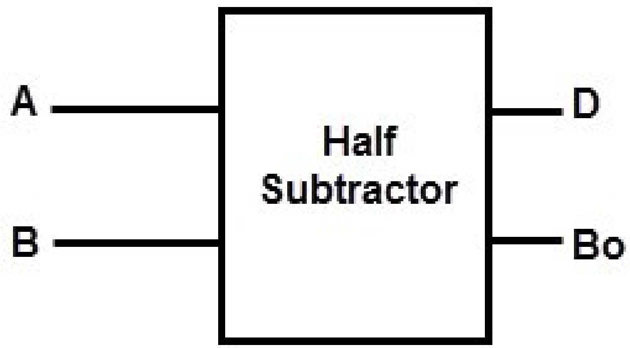
2) Full Subtractor

Half Subtractor

Half Subtractor :

Half Subtractor is used for subtracting one single bit binary digit from another single bit binary digit.The truth table of Half Subtractor is shown below.

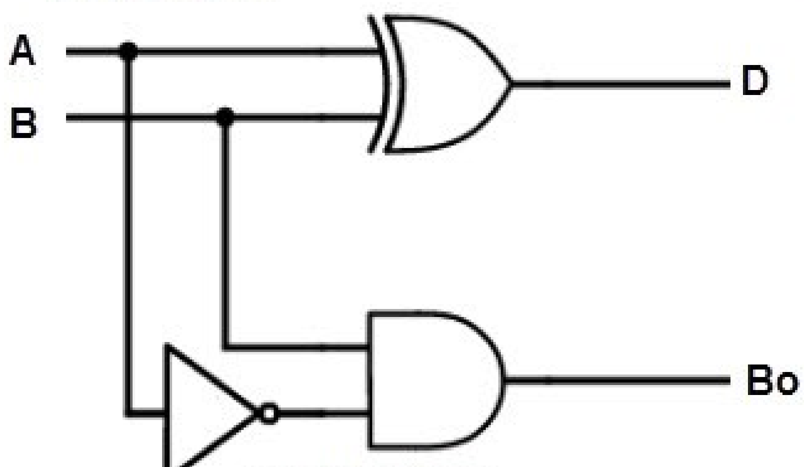
Half Subtractor



Block Diagram

A	B	D	B_o
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

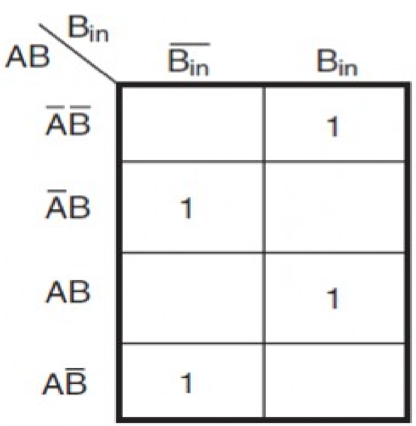
Truth Table



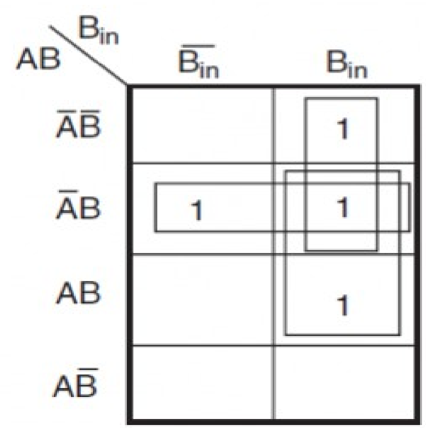
Logic Circuit of
Half Subtractor

Full Subtractor

Minuend (A)	Subtrahend (B)	Borrow In (B_{in})	Difference (D)	Borrow Out (B_o)
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

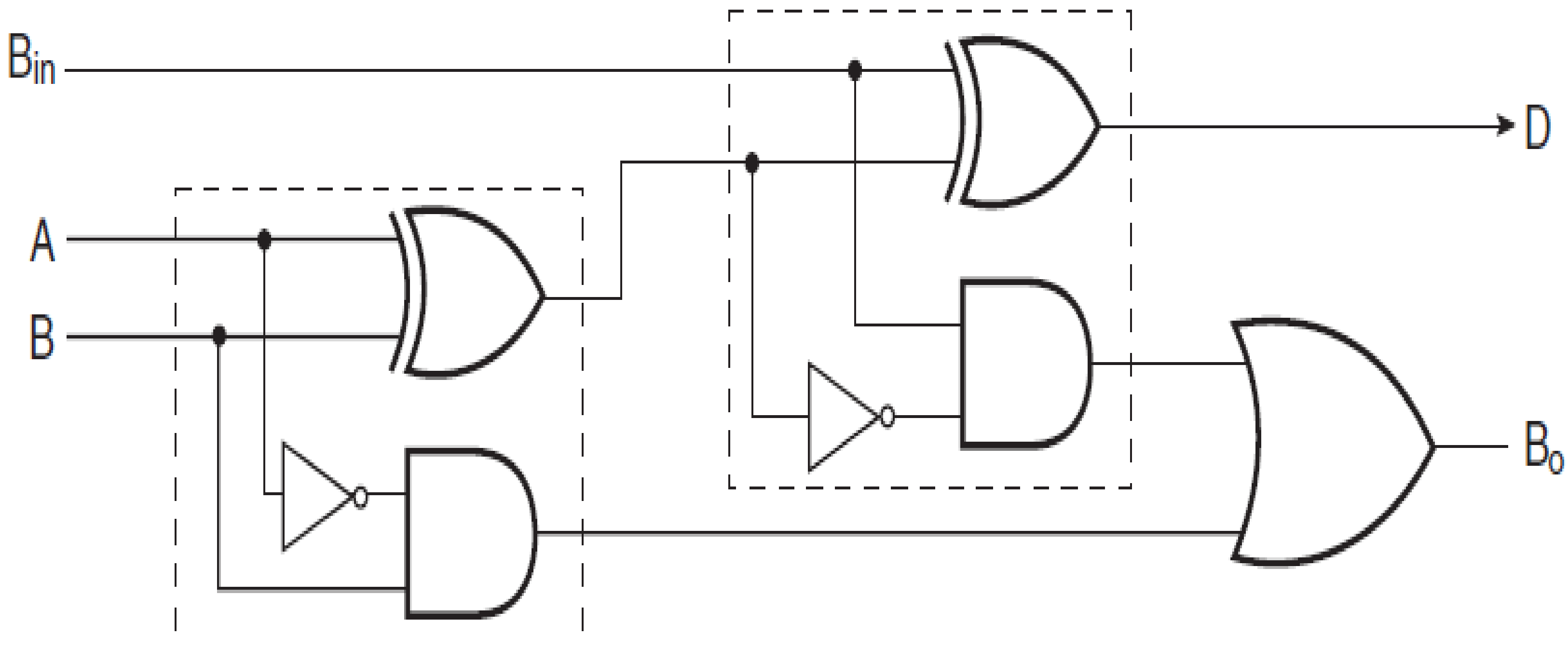


$$D = \overline{A}.\overline{B}.B_{in} + \overline{A}.B.\overline{B}_{in} + A.\overline{B}.\overline{B}_{in} + A.B.B_{in}$$



$$B_o = \overline{A}.\overline{B}.B_{in} + \overline{A}.B.\overline{B}_{in} + \overline{A}.B.B_{in} + A.B.B_{in}$$

full subtractor using logic gates.



* Combinational logic design using Multiplexer :

ex 1]

	I/P			O/P
Decimal	A	B	C	y
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Ans →

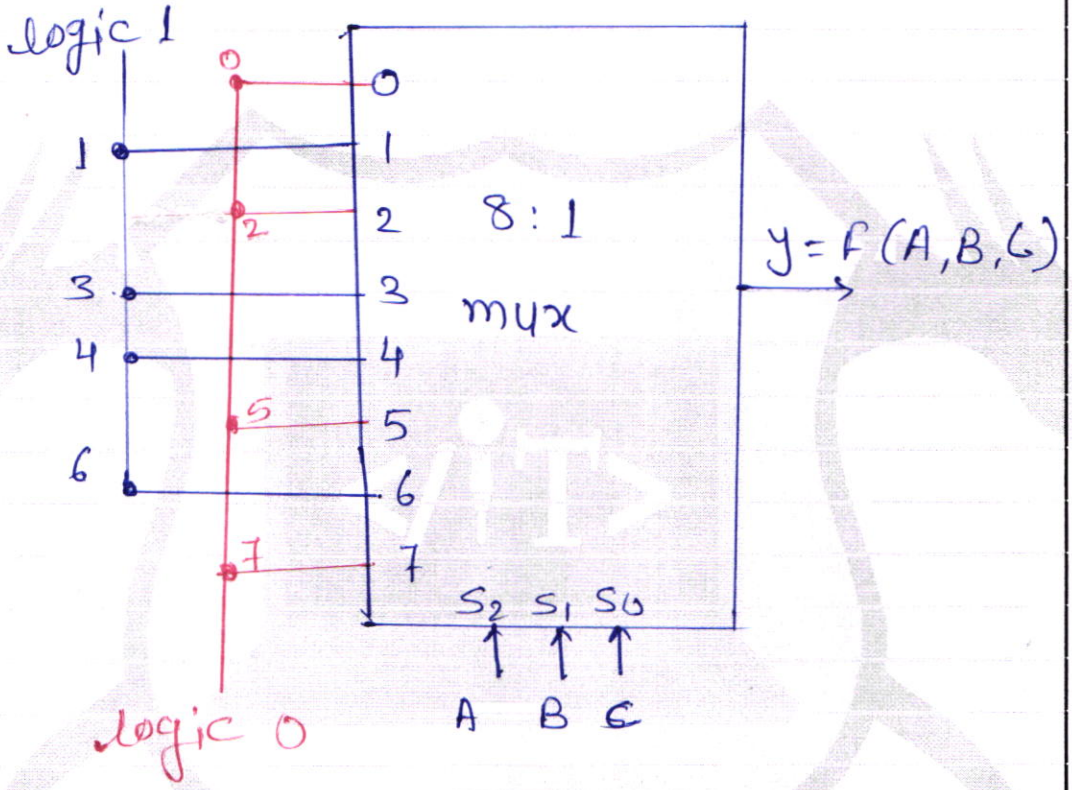
Step I : Express the o/p y in SOP Form

$$\boxed{y = \sum m(1,3,4,6)}$$

Step II : Connect data i/p's 1,3,4,6 to logic 1 & the remaining to logic 0

Step III : Connect i/p A, B, C to the select lines S_2, S_1, S_0 respectively

Step 4: Draw the logic Diagram



ex 2] Implement the logic fu^n EC

F(A, B, C) = Σm (1, 3, 4, 6) using 4:1 mux

⟹ Given F(A, B, C) = Σm (1, 3, 4, 6)

step I: Apply two variables B & C to the select i/p's lines S_1 & S_0 respectively.

Step II :- write the design table.

	D_0	D_1	D_2	D_3 ← Data i/p's	
$\bar{A}$	0	1	2	3	→ Row 1
A	4	5	6	7	→ Row 2
	↓ A	↓ $\bar{A}$	↓ A	↓ $\bar{A}$	

Encircle those minterms Corresponding to which the o/p is 1
(minterms 1, 3, 4, 6)

Step III : Check the column in the design table.

Rule 1 : It both minterms in a column are not circled, the apply logic 0 to the corresponding data i/p.

Rule 2 : IF only the minterm in the 2nd row is encircled the A should be applied to that data i/p.

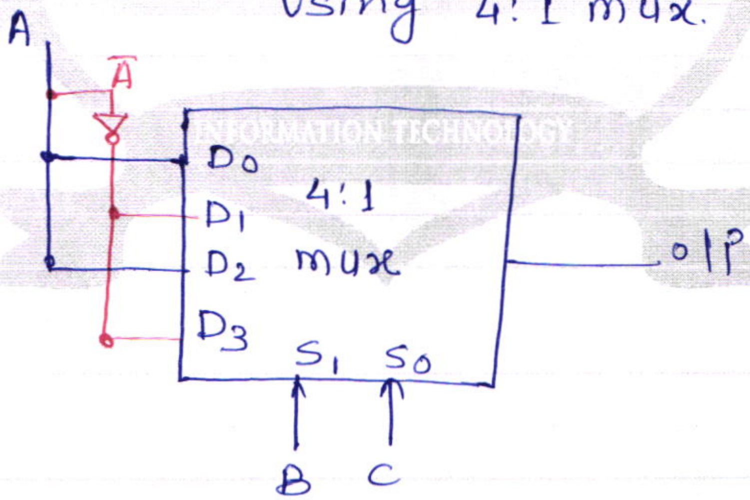
Note
In this example, we should apply A For data i/p D_0 & D_2

Rule 3: If only the minterm in 1st row is encircled, then $\overline{A}$ should be applied to that data i/p.

Note: In this example, we should apply $\overline{A}$ to data i/p D_1 & D_3

Rule 4:- If both minterms in the column are encircled, then apply logic 1 to the corresponding data i/p.

Step 4: Draw the logic diagram using 4:1 mux.



logic diagram

ex 3] Implement the following fu^n using 16:1 multiplexer

$F(A,B,C,D) = \Sigma m(2,4,5,7,10,14,15,16,17,25,26,30,31)$

→ Step I: Apply the i/p variables B, C, D, E to the select i/p's S_3, S_2, S_1 & S_0 respectively.

Step II: Write the design table & follows the rules from example 2

I/P's	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}	
$\bar{A}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	1^{st} row
A	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	2^{nd} row
I/P to mux	A	A	$\bar{A}$	0	$\bar{A}$	$\bar{A}$	0	$\bar{A}$	0	A	1	0	0	0	1	1	

Step III

Rule Please follow the rules from example 2

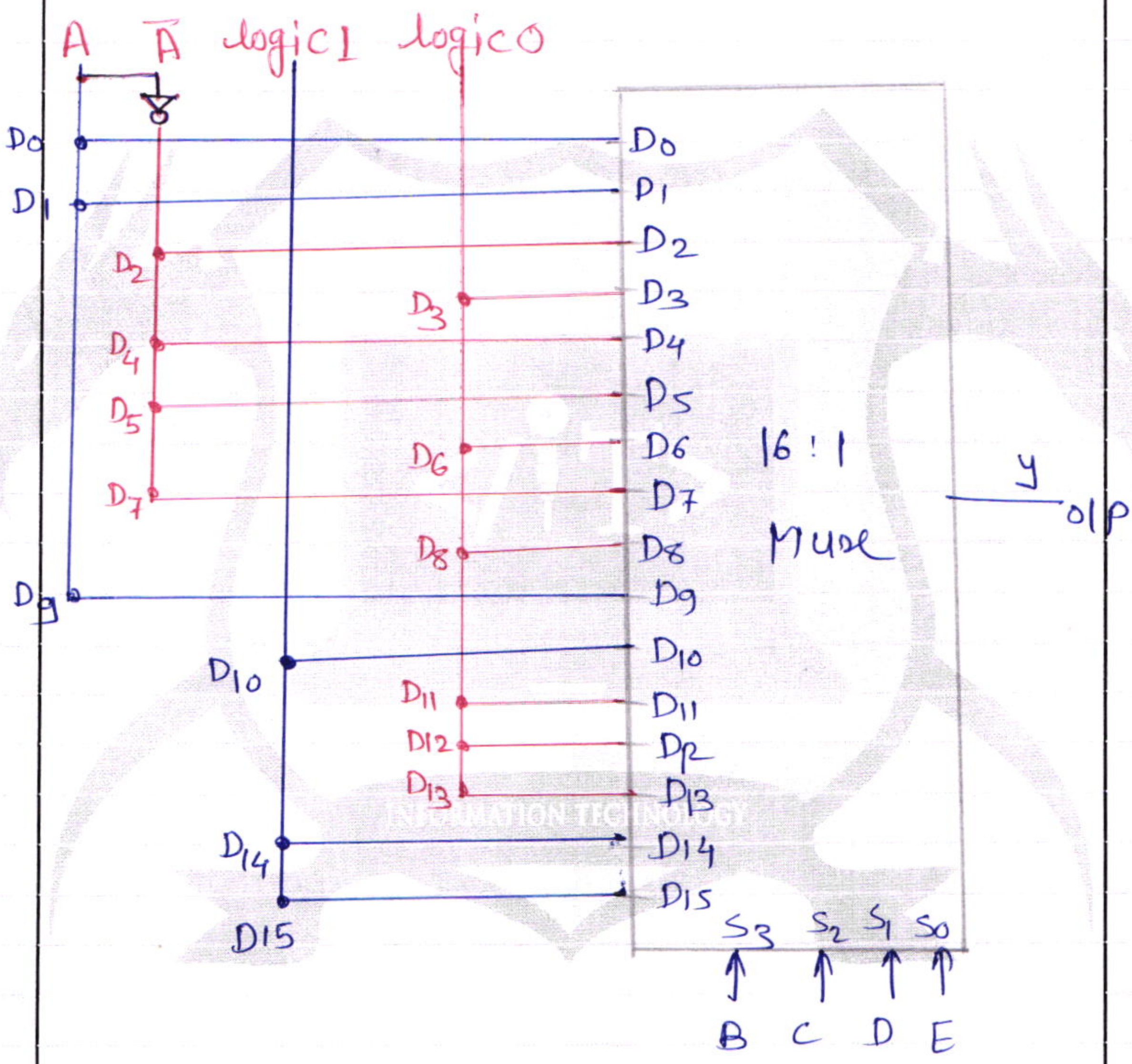
A → only encircled no. from 2^{nd} row

$\bar{A}$ → " " " " 1^{st} row

0 → both no.s are not encircled in 1^{st} & 2^{nd} row

1 → " " " encircled in 1^{st} & 2^{nd} row

Step IV : Relize the logic ckt.



Logic Diagram using 16:1 mux

* Multiplexer tree :

- The mux having more no. of i/p's can be obtained by cascading two or more multiplexers with less no. of inputs.

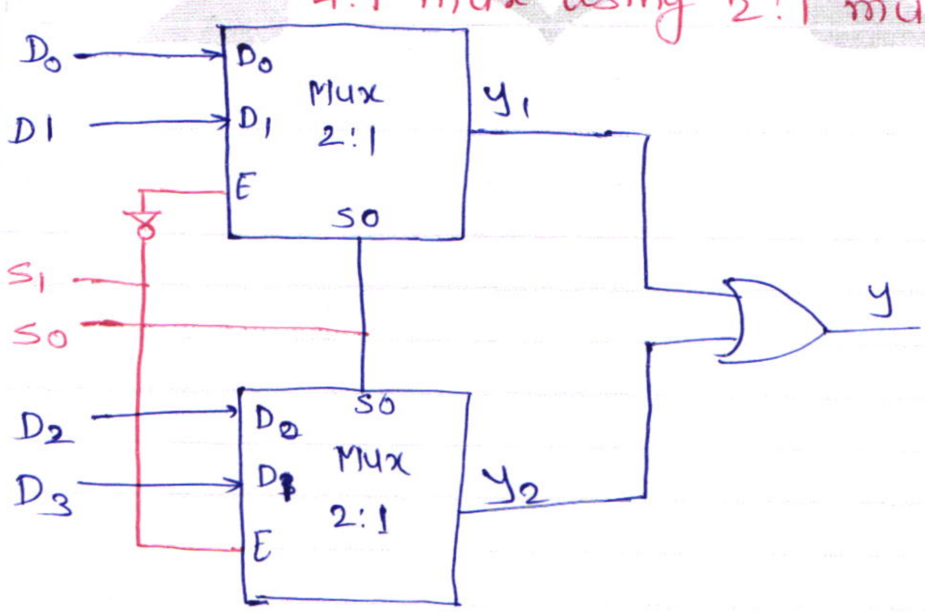
ex 1] Implement 4:1 mux using 2:1 mux

⇒

selects i/p's		o/p	
S_1	S_0	y	
0	0	D_0	Mux 1
0	1	D_1	
1	0	D_2	Mux 2
1	1	D_3	

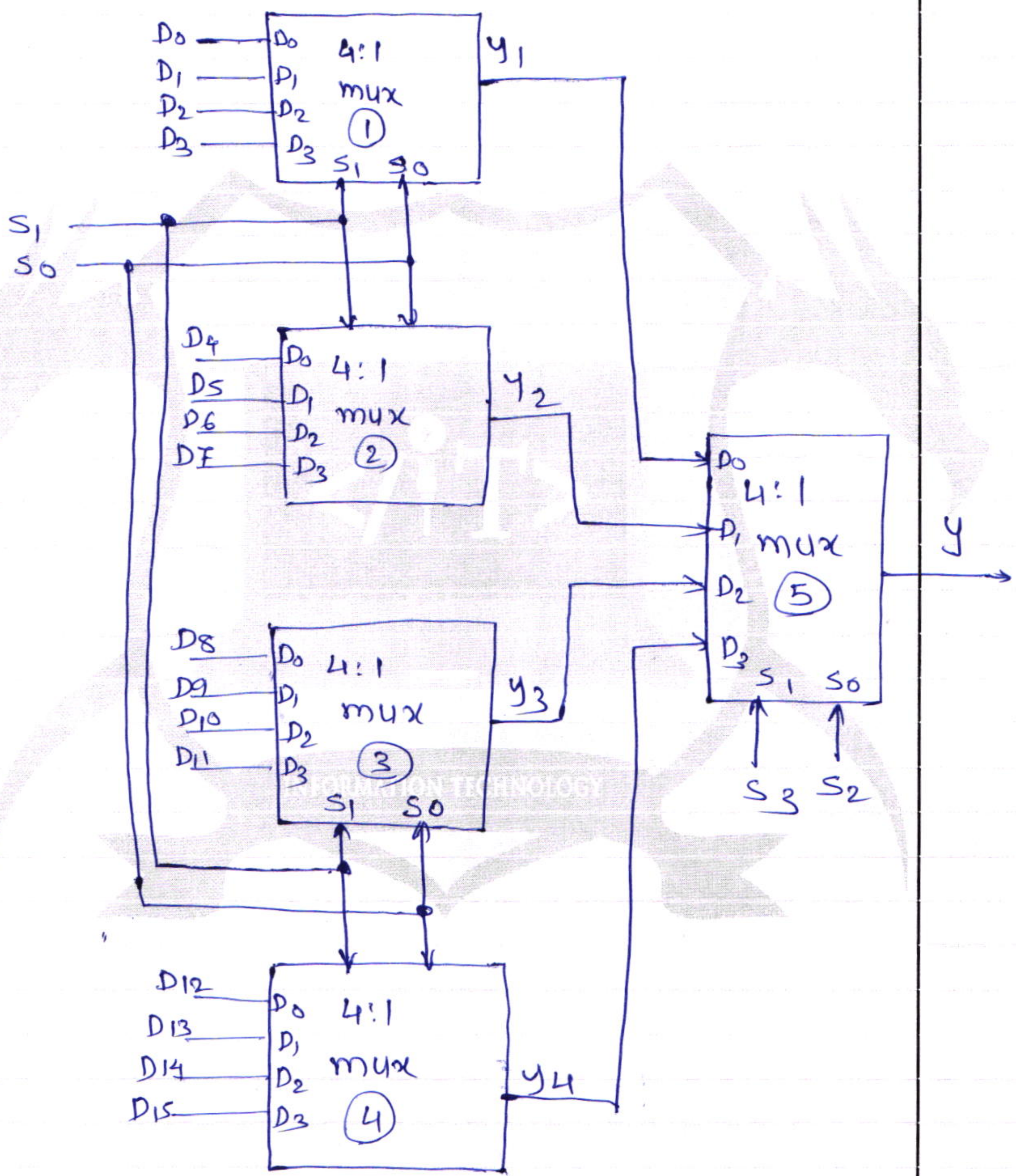
⟹ Truth Table

4:1 mux using 2:1 mux



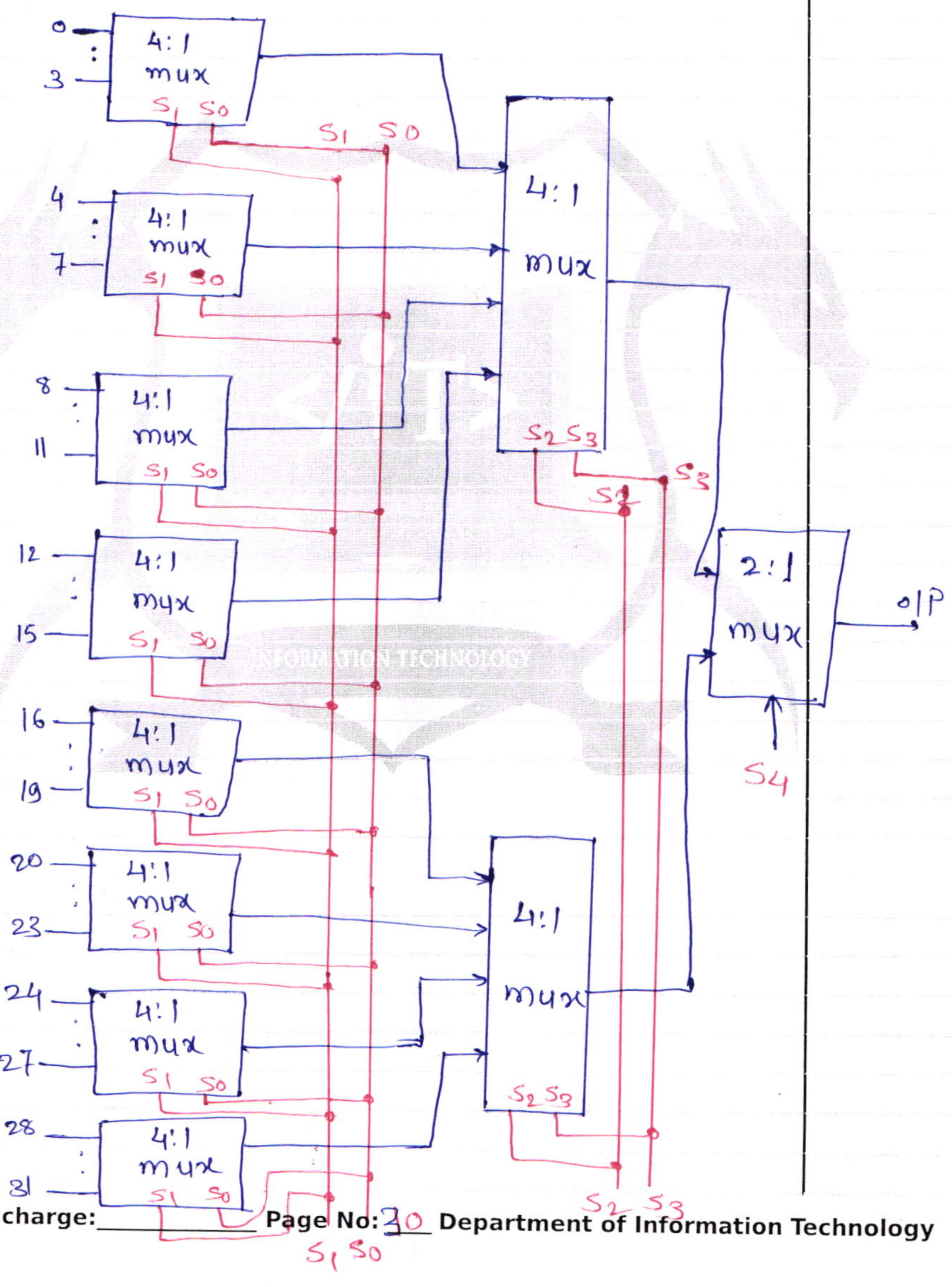
ex2] Design 16:1 multiplexer using 4:1 mux

select i/p's				o/p		
S_3	S_2	S_1	S_0	y		
0	0	0	0	D_0	Mux 1 $[D_0 - D_3]$	$S_3 S_2 = 00$ selects y_1
0	0	0	1	D_1		
0	0	1	0	D_2		
0	0	1	1	D_3		
0	1	0	0	D_4	Mux 2 $(D_4 - D_7)$	$S_3 S_2 = 01$ selects y_2
0	1	0	1	D_5		
0	1	1	0	D_6		
0	1	1	1	D_7		
1	0	0	0	D_8	Mux 3 $(D_8 - D_{11})$	$S_3 S_2 = 10$ selects y_3
1	0	0	1	D_9		
1	0	1	0	D_{10}		
1	0	1	1	D_{11}		
1	1	0	0	D_{12}	Mux 4 $(D_{12} - D_{15})$	$S_3 S_2 = 11$ selects y_4
1	1	0	1	D_{13}		
1	1	1	0	D_{14}		
1	1	1	1	D_{15}		



16:1 mux using 4:1 mux.

ex 3] Design 32:1 mux using 4:1 mux



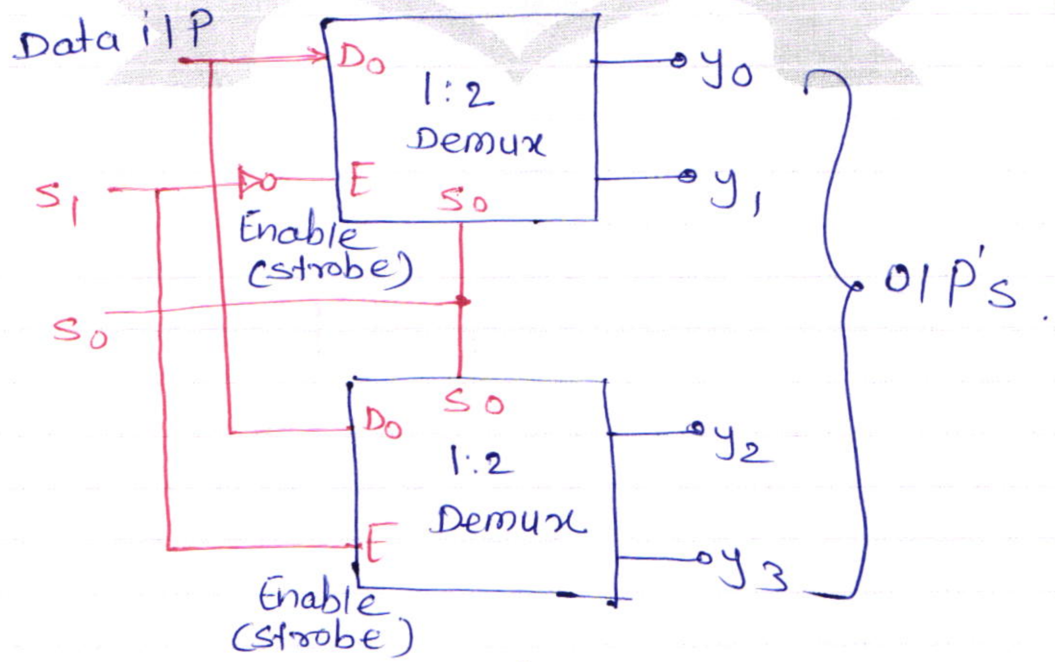
* Demultiplexer Tree :-

ex 1] Implement 1:4 demux using 1:2 demux

⟹

I/P's		O/P's				
S_0	S_1	Y_3	Y_2	Y_1	Y_0	
0	0	0	0	0	D_0	Demux 1
0	1	0	0	D_0	0	
1	0	0	D_0	0	0	Demux 2
1	1	D_0	0	0	0	

- select lines S_0 of the both 1:2 demux are connected together & brought out as S_0 line of the 1:4 demux.

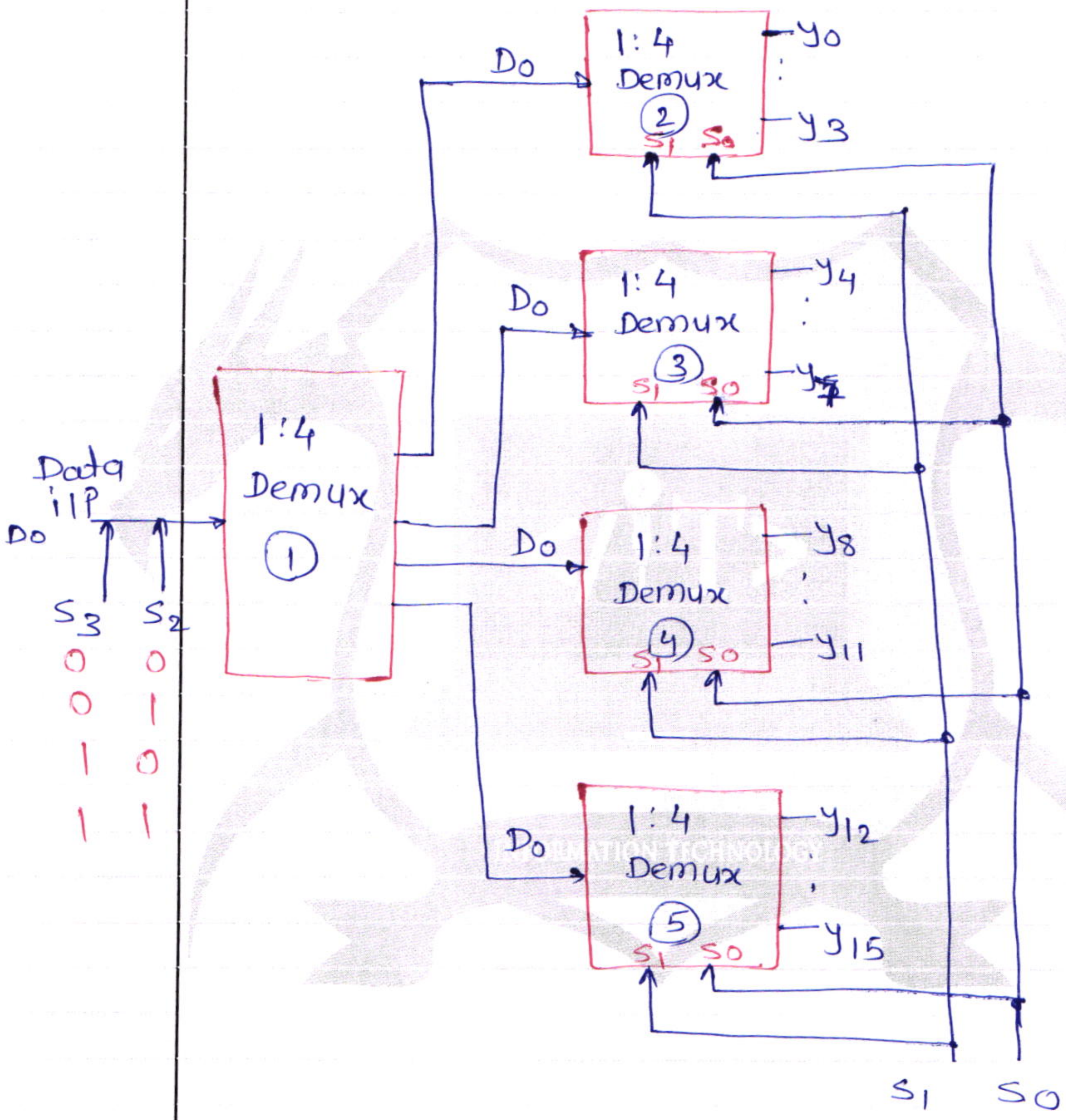


ex 2] Implement 1:16 demux using 1:4 demux

→ Truth table

	S_3	S_2	S_1	S_0	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}
$S_3 S_2$ 0 0 Demux (2) selected	0	0	0	0	D_0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	1	0	D_0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	0	0	0	D_0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	1	0	0	0	D_0	0	0	0	0	0	0	0	0	0	0	0	0
$S_3 S_2$ 0 1 Demux (3) selected	0	1	0	0	0	0	0	0	D_0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	1	0	0	0	0	0	D_0	0	0	0	0	0	0	0	0	0	0
	0	1	1	0	0	0	0	0	0	0	D_0	0	0	0	0	0	0	0	0	0
	0	1	1	1	0	0	0	0	0	0	0	D_0	0	0	0	0	0	0	0	0
$S_3 S_2$ 1 0 Demux (4) selected	1	0	0	0	0	0	0	0	0	0	0	0	D_0	0	0	0	0	0	0	0
	1	0	0	1	0	0	0	0	0	0	0	0	0	D_0	0	0	0	0	0	0
	1	0	1	0	0	0	0	0	0	0	0	0	0	0	D_0	0	0	0	0	0
	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	D_0	0	0	0	0
$S_3 S_2$ 1 1 Demux (5) selected	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D_0	0	0	0
	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	D_0	0	0
	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D_0	0
	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D_0

S_3	S_2	Demux selection
0	0	→ 2
0	1	→ 3
1	0	→ 4
1	1	→ 5



1:16 Demux using 1:4 Demux

* Use Demux in combinational logic Design.

ex1] Implement Full adder using 1:8 Demux

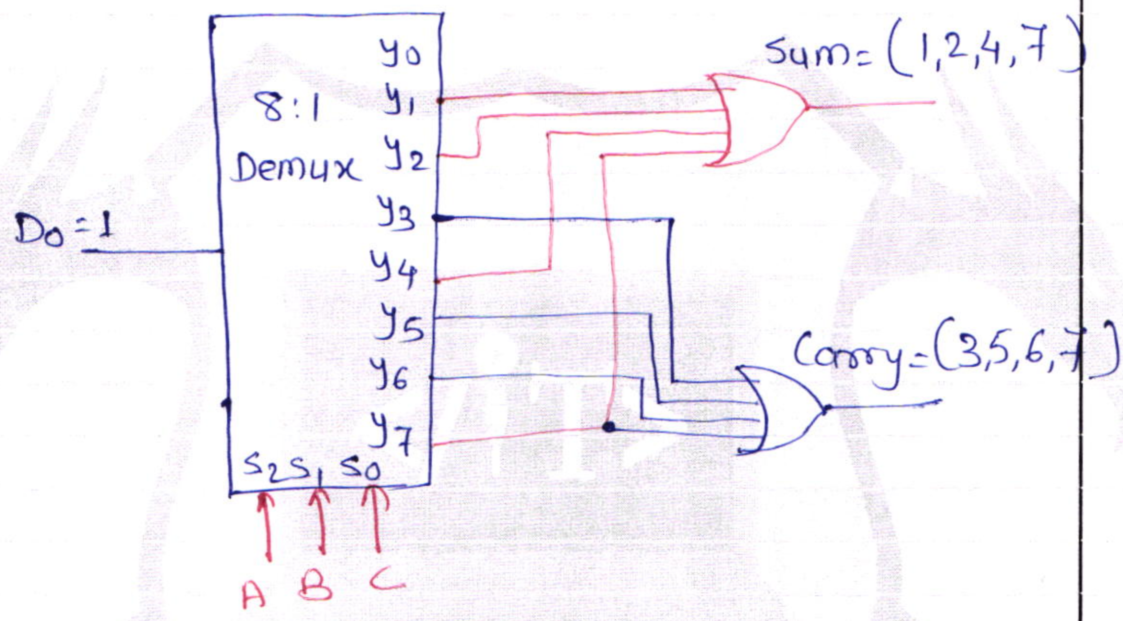
I/P's			O/P's		
A	B	C	Sum	Carry	Decimal
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	2
0	1	1	0	1	3
1	0	0	1	0	4
1	0	1	0	1	5
1	1	0	0	1	6
1	1	1	1	1	7

Bollean expression [Minterms] in SOP Form For Sum & Carry

1] Sum = $\sum m(1,2,4,7)$

2] Carry = $\sum m(3,5,6,7)$

Design of Full Adder using 1:8 demux



Ex 2] Implement full subtractor using 1:8 Demultiplexer

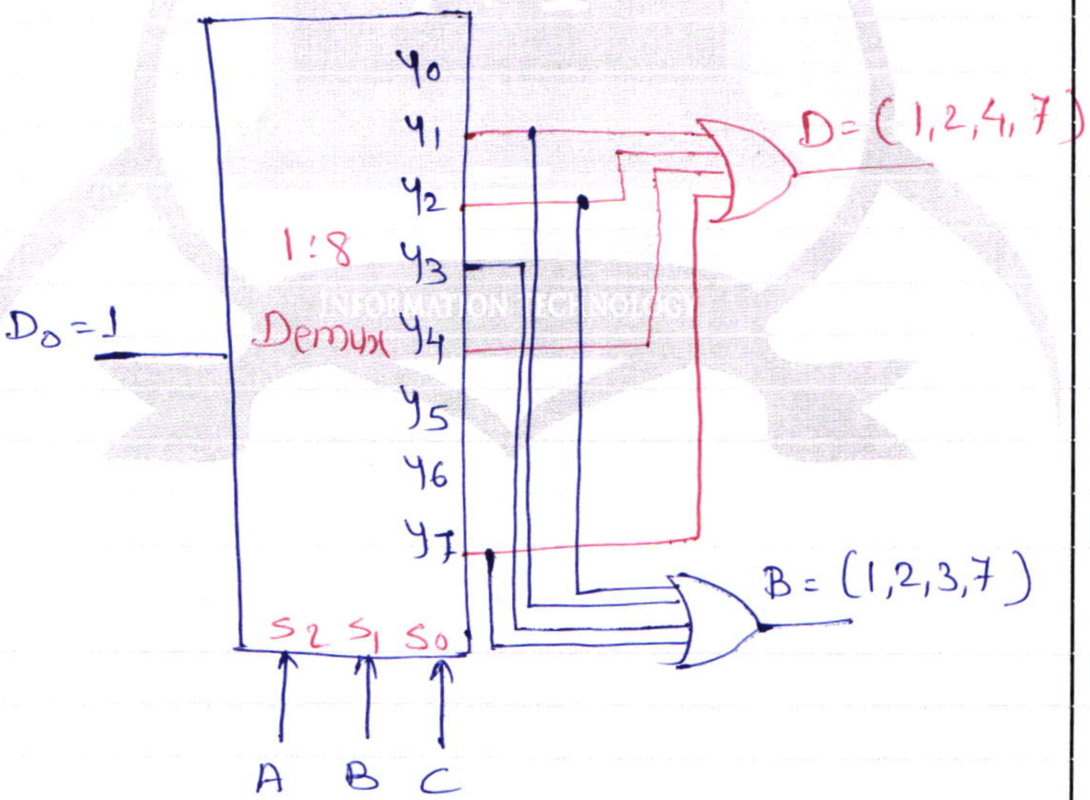
I/P's			O/P's		
A	B	C	Difference	Borrow	Decimal
0	0	0	0	0	0
0	0	1	1	1	1
0	1	0	1	1	2
0	1	1	0	1	3
1	0	0	1	0	4
1	0	1	0	0	5
1	1	0	0	0	6
1	1	1	1	1	7

From the truth table we can express the difference & Borrow ~~out~~ O/P's in the SOP form.

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

Borrow (B) = $\Sigma m(1,2,3,7)$

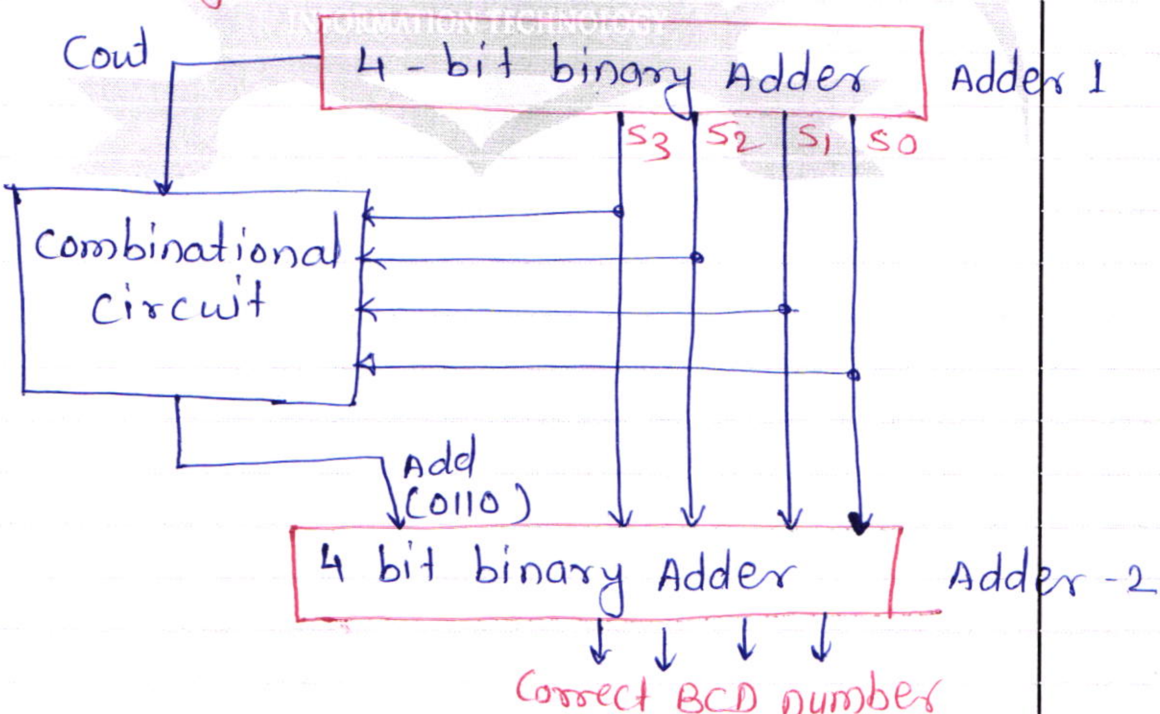
Design Full subtractor using 1:8 demux.



* BCD [Binary Coded Decimal] Adder:

- Using IC 74283
- It can't be greater than 9
  Range [0-9]
- If sum <= 9 & carry = 0,
  then result is correct & in the true BCD Form
- If sum > 9 or carry = 1,
  then result is wrong & not in true BCD Form
- Wrong result can be corrected by adding six (0110)

Block Diagram - Two-4-bit BCD numbers



ex1] Design BCD Adder using 4-bit binary Adders.

→ Truth table for 4-bit binary Adders.

Decimal	S_3	S_2	S_1	S_0	y
	I/P's				O/P
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

Rows 0–9: sum is valid BCD numbers, $y = 0$, Range [0-9]

Rows 10–15: sum is invalid BCD numbers, $y = 1$

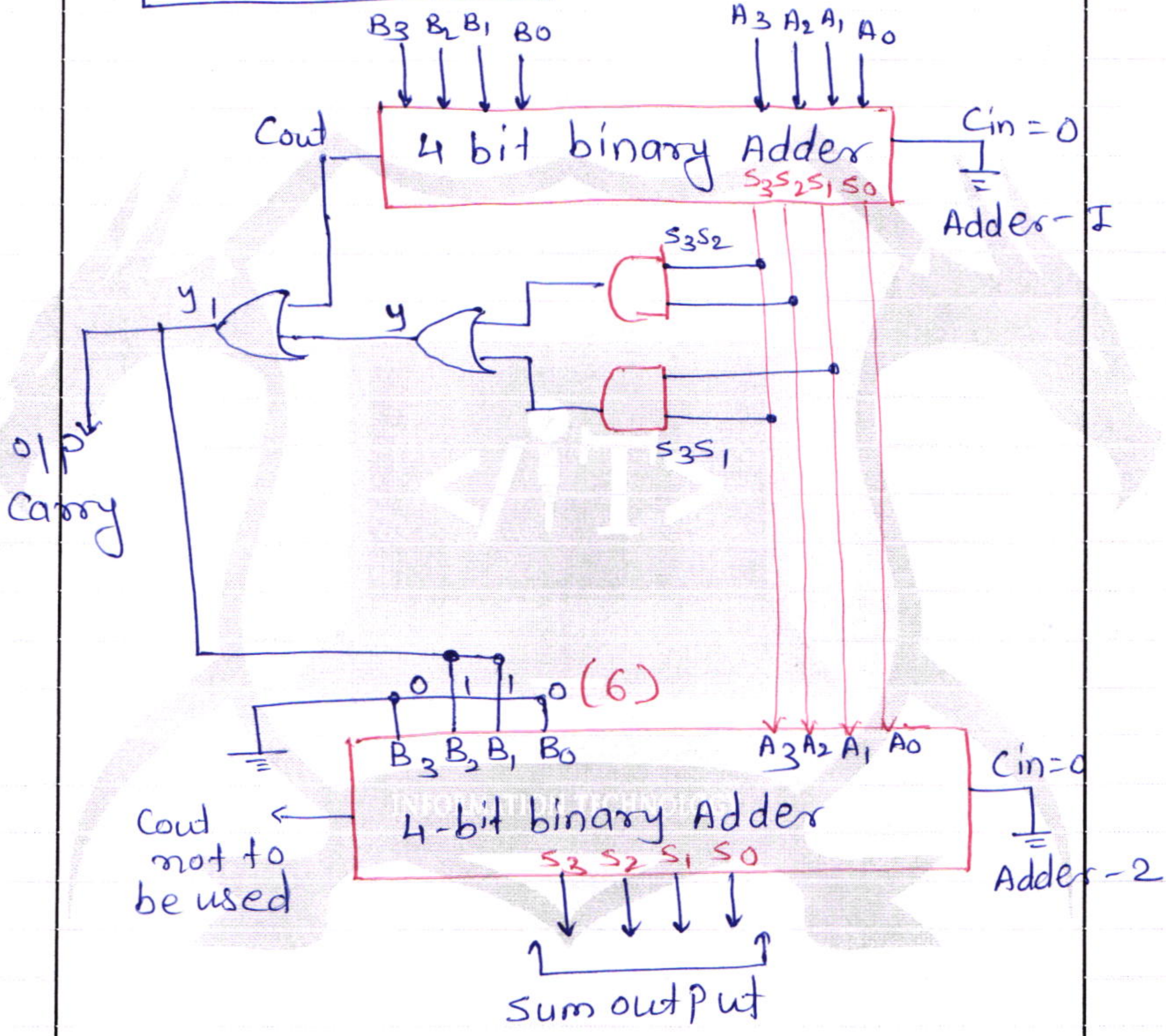
K-map for O/P y.

S_3S_2 \ S_1S_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

S_3S_2

S_1S_3

$$\boxed{y = S_3S_2 + S_1S_3}$$



Case I : Sum ≤ 9 & Carry $= 0$

- O/P of combinational ckt $y_1 = 0$
- $\therefore B_3B_2B_1B_0 = 0000$ for adder 2
- O/P of adder 2 is same as that of adder - 1

Case - II :- sum > 9 & Carry = 0

- o/p of adder is greater than 9, then o/p of y1 of combinational ckt becomes 1

  ∴ $B_3 B_2 B_1 B_0 = 0110$ of adder 2

- ∴ six (0110) will be added to the sum o/p of adder-1.

- we get corrected result of BCD at the sum o/p of adder-2

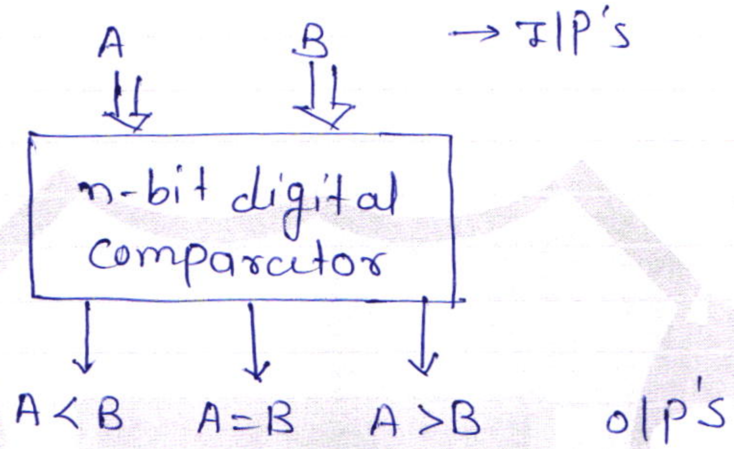
Case - III : sum ≤ 9 but Carry = 1

- As Carry o/p of adder 1 is high, $y_1 = 1$

  ∴ $B_3 B_2 B_1 B_0 = 0110$ will be added to the sum o/p of adder 1

- we get the corrected BCD result at the sum o/p of adder-2.

* Magnitude Comparators :—



- It is a Combinational ckt.
- It has three outputs, those are A>B, A=B & A<B.
- It is depending on the result of comparison, one of these O/P's will go high.
- It has Four types

1] 1-Bit Binary comparator
2] 2 - " " "
3] 4 - " " "
4] 8 - " " "

I] 1-Bit Binary comparator :-

- It has two i/p's A & B & three outputs 1] $A<B$ 2] $A=B$ 3] $A>B$
- Truth Table

I/P's		O/P's		
A	B	$y_1 = A<B$	$y_2 = A=B$	$y_3 = A>B$
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

K-map

For y_1

A \ B	0	1
0	0	1
1	0	0

$$\boxed{y_1 = \bar{A}B}$$

For y_2

A \ B	0	1
0	1	0
1	0	1

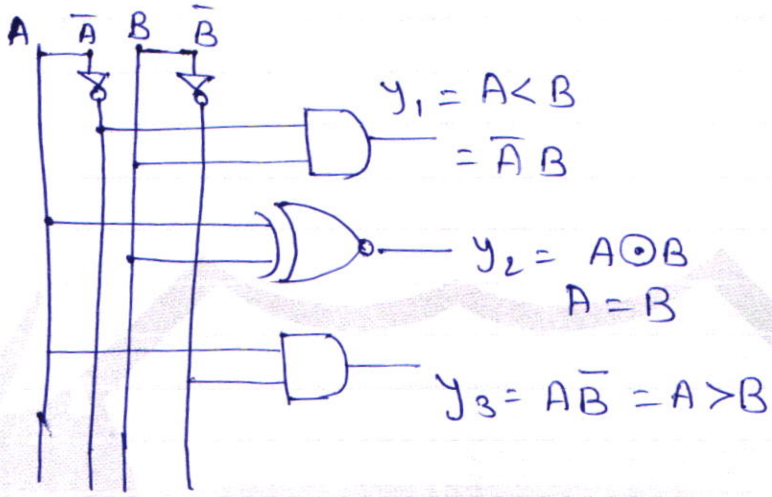
$$\boxed{y_2 = \bar{A}\bar{B} + AB}$$

$$\boxed{y_2 = A \odot B}$$

For y_3

A \ B	0	1
0	0	0
1	1	0

$$\boxed{y_3 = A\bar{B}}$$



Realization of one bit comparator

II] 2-bit magnitude comparator :-

Truth Table

I/P's				O/P's		
A_1	A_0	B_1	B_0	A<B	A=B	A>B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

k-maps

For $A<B$

$A_1A_0 \backslash B_1B_0$	00	01	11	10
00		1	1	1
01			1	1
11				
10			1	

$\rightarrow \bar{A}_1 B_1$

$\bar{A}_1\bar{A}_0 B_0$ $\quad$ $\bar{A}_0 B_1 B_0$

$$\boxed{A<B = \bar{A}_1\bar{A}_0 B_0 + \bar{A}_1 B_1 + \bar{A}_0 B_1 B_0}$$

For $A>B$:

$A_1A_0 \backslash B_1B_0$	00	01	11	10
00				
01	1			
11	1	1		1
10	1	1		

$A_0\bar{B}_1\bar{B}_0$ $\quad$ $\rightarrow A_1A_0\bar{B}_0$ $\quad$ $A_1\bar{B}_1$

$$\boxed{A>B = A_0\bar{B}_1\bar{B}_0 + A_1A_0\bar{B}_0 + A_1\bar{B}_1}$$

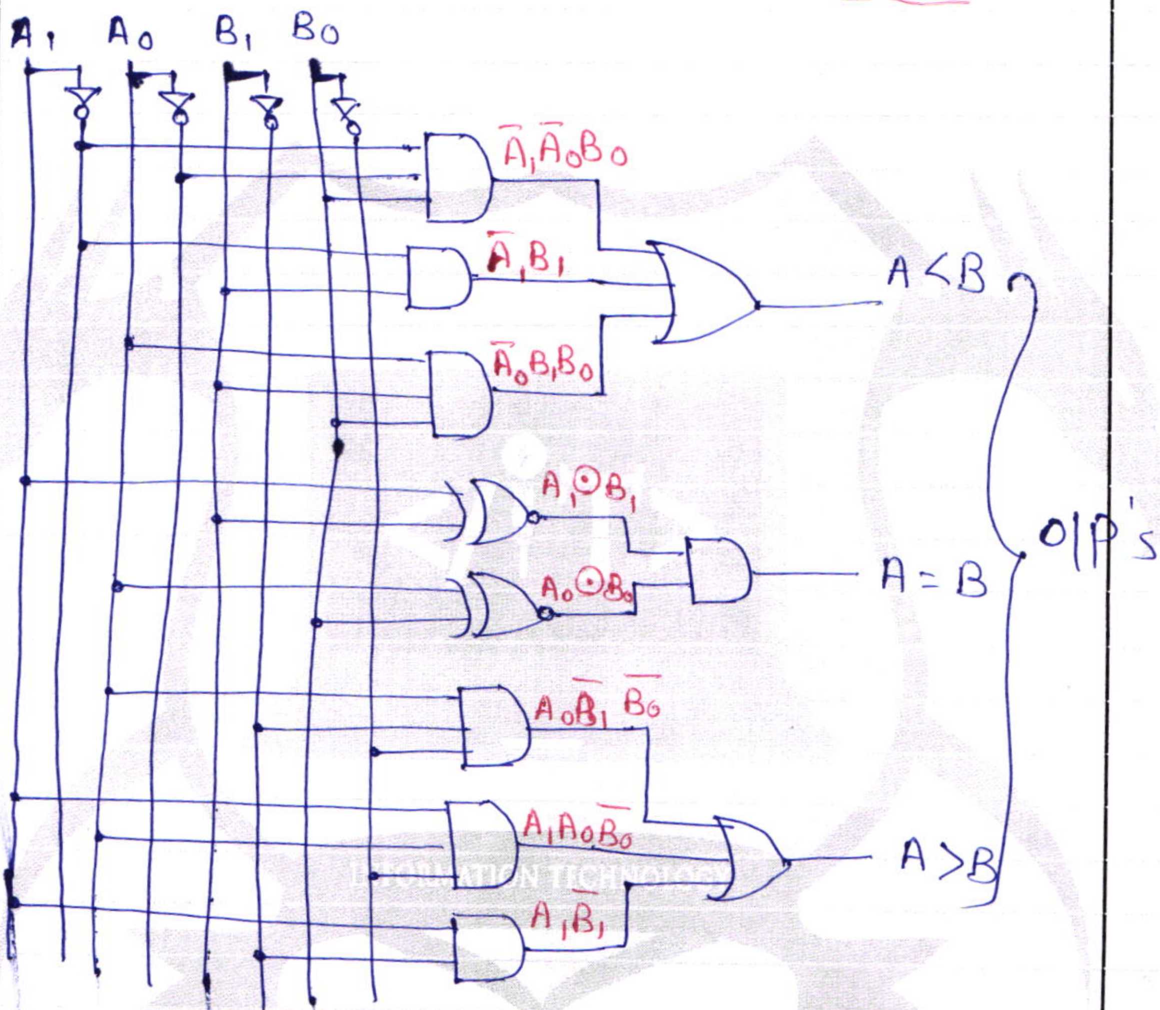
For $A=B$

$A_1A_0 \backslash B_1B_0$	00	01	11	10
00	1			
01		1		
11			1	
10				1

$$\boxed{A=B = (A_1 \odot B_1)(A_0 \odot B_0)}$$

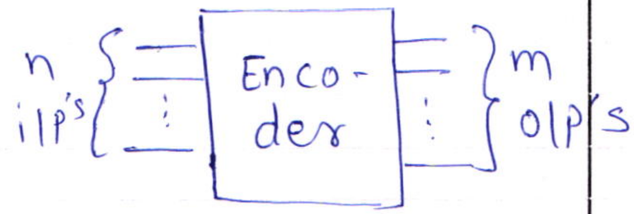
$\rightarrow$ EX-NOR

Logic Diagram for 2-bit Comparator



* Encoders

Block Diagram



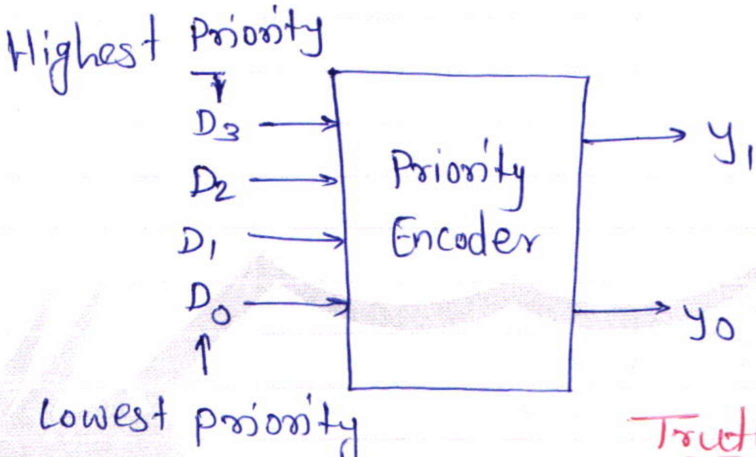
- Combinational ckt
- Inverse oper^n of the decoder
- n → number of i/p lines
- m → number of o/p lines

Types

1] Priority Encoder
2] Decimal to BCD
3] Octal to Binary (8:3)
4] Hexadecimal to Binary

I] Priority Encoder :-

- This is special type of Encoder
- Priorities are given to the i/p lines.
- There are four i/p's. D_0 to D_3 & two o/p's y_1 & y_0.
- As shown in block diagram. D_3 has the highest priority
  D_0 " " lowest "



Truth Table

Highest	I/P's		Lowest	O/P's	
D_3	D_2	D_1	D_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

K-map

For y_1

D_3D_2 \ D_1D_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

$\rightarrow D_2$

$\rightarrow D_3$

$$\boxed{y_1 = D_3 + D_2}$$

For y_0

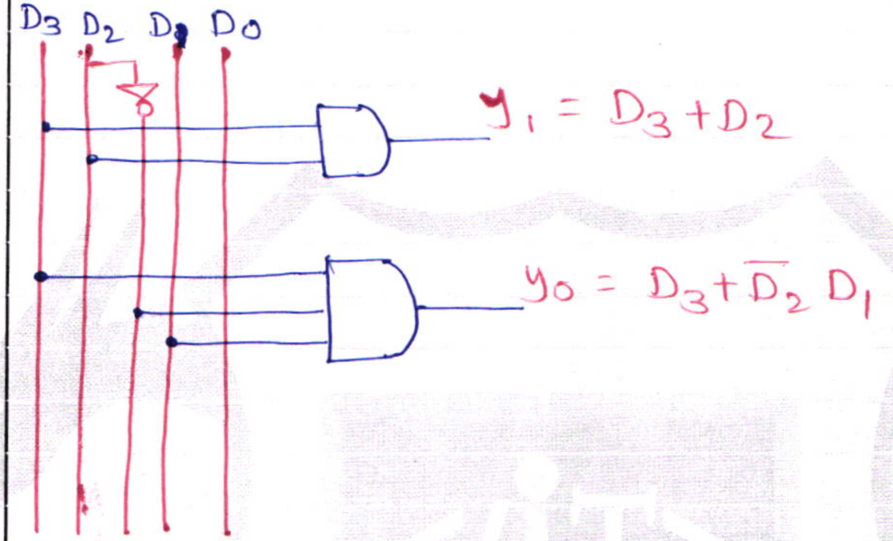
D_3D_2 \ D_1D_0	00	01	11	10
00	X	0	1	1
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$\rightarrow \bar{D}_2 D_1$

$\rightarrow D_3$

$$\boxed{y_0 = D_3 + \bar{D}_2 D_1}$$

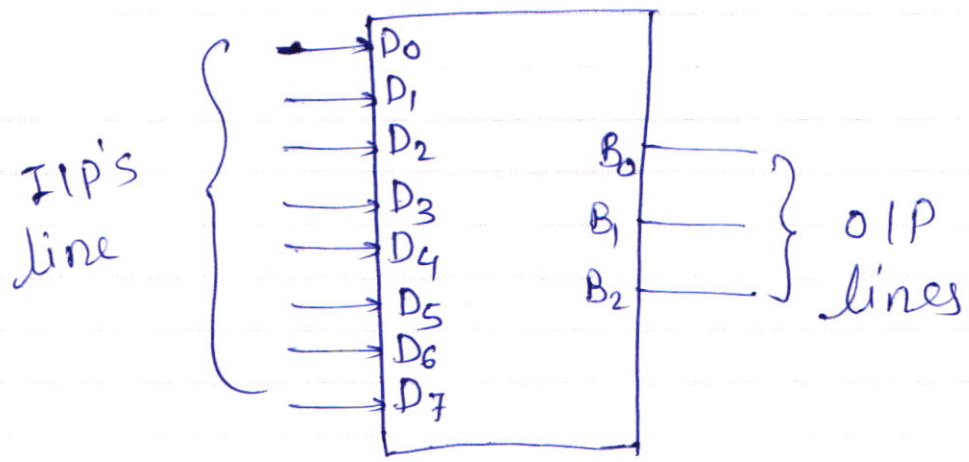
Implementation - Logic Diagram



Logic Diagram for Priority Encoder

II] Octal to Binary Encoder :-

- It has 8 i/p lines & 3 o/p lines.
- It is also called as 8:3 Encoder
- Block Diagram



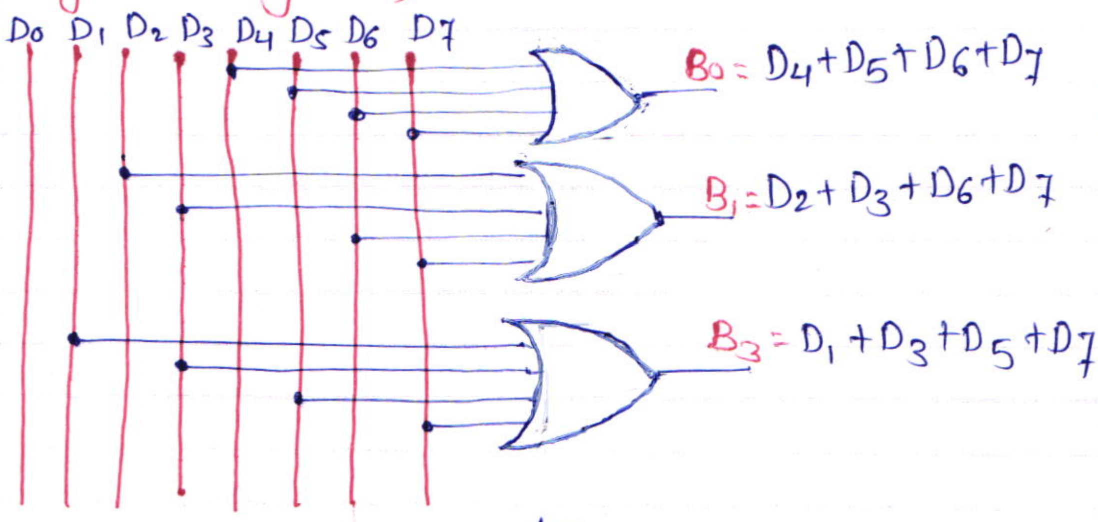
Truth Table

I/P's								O/P's		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	B_0	B_1	B_2
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

Boolean expression

$$B_0 = D_4 + D_5 + D_6 + D_7$$
$$B_1 = D_2 + D_3 + D_6 + D_7$$
$$B_2 = D_1 + D_3 + D_5 + D_7$$

Logic Diagram



* Decoder :-

- combinational ckt
- It has n i/p & to a maximum 2^n o/p
- It is identical to a demultiplexer without any data i/p.
- It performs the operation which are exactly opposite to those of an encoder.

Appl^n

1] code converters
2] BCD to seven segements decoders

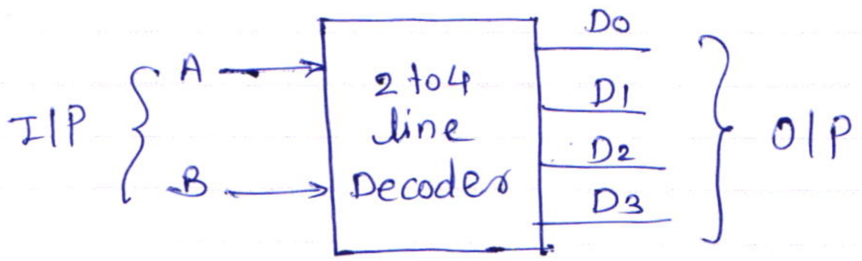
Types:

1] 2 to 4 line decoder
2] 3 to 8 line decoder
3] BCD to seven segment decoder.

I] 2 to 4 line Decoder :-

- It has two i/p's & Four o/p's

Block diagram



Truth Table

It shows that o/p is 1 for only a specific combination of i/p's.

I/p's		o/p's			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

logical expression

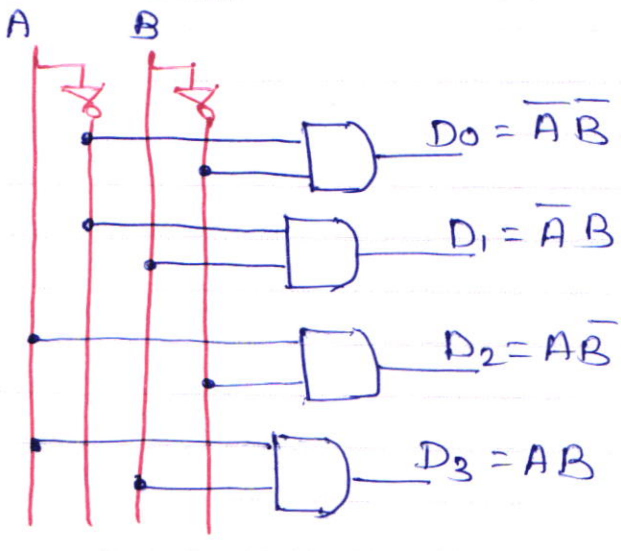
$D_0 = \bar{A}\bar{B}$

$D_1 = \bar{A}B$

$D_2 = A\bar{B}$

$D_3 = AB$

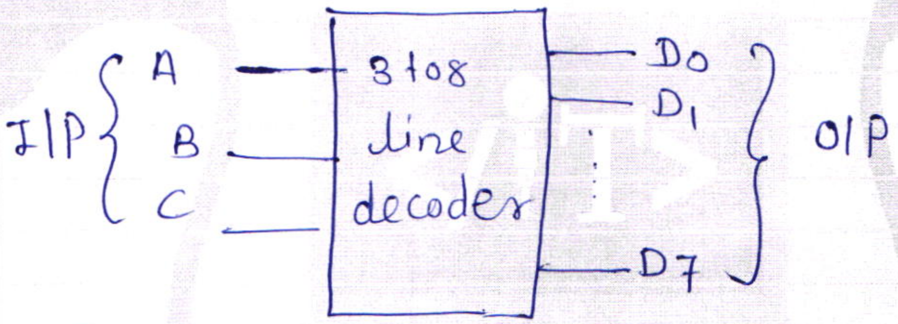
combinational logic ckt



II] 3 to 8 line Decoder :-

- It has three i/p's & 8 o/p's
- I/p's are A, B & C
- o/p's are D_0 to D_7

Block diagram



Truth Table

I/P's			O/P's							
A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
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

Boolean Expression

$D_0 = \bar{A}\bar{B}\bar{C}$ $D_1 = \bar{A}\bar{B}C$

$D_2 = \bar{A}B\bar{C}$ $D_3 = \bar{A}BC$

$D_4 = A\bar{B}\bar{C}$ $D_5 = A\bar{B}C$

$D_6 = AB\bar{C}$ $D_7 = ABC$

Combinational logic ckt - 3 to 8 line decoder