Chapter - 3

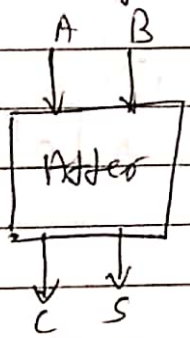
Design examples.
1) Half adder and full adder
2) Half subtractor and full substractor.

1) Adder :-
Addition of two binary digits is most basic operation performed by the digital computers.

Rule of binary addition.

A	B	sum(s)	carry(c)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Block diagram of adder.



Types of binary adder :-

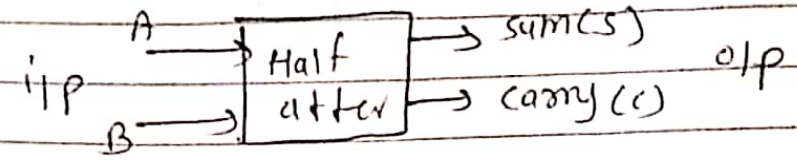
1) Half adder & 2) full adder.

1) Half adder :-
Def^n :- Half adder is a combinational logic circuit with two input and two outputs.

(2)

classmate Date Page

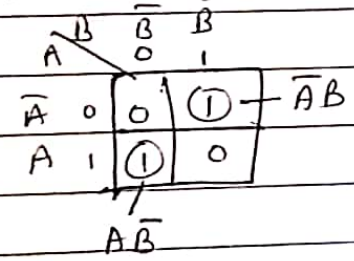
block diagram :-



Truth Table :-

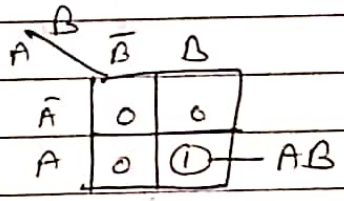
Inputs		outputs	
A	B	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Design using k-maps :-



for sum

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

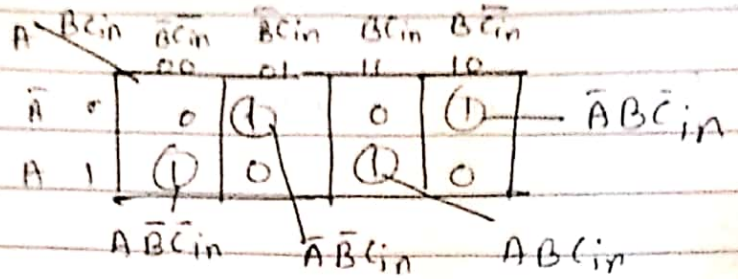


for carry.

$C = AB$

Design using K-Map:

for sum (S):



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

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

Ex-NOR Ex-OR

$$S = C_{in}\overline{(\bar{A}B + A\bar{B})} + \bar{C}_{in}(\bar{A}B + A\bar{B})$$

Let $x = \bar{A}B + A\bar{B}$

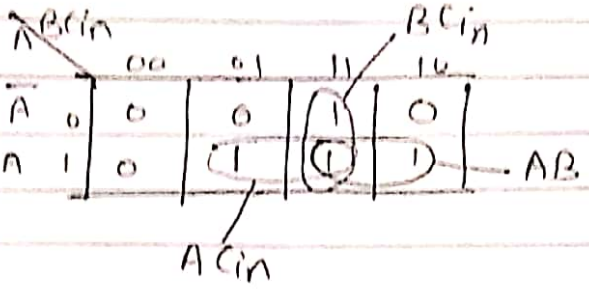
$$S = C_{in}\bar{x} + \bar{C}_{in}x = C_{in} \oplus x$$

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

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

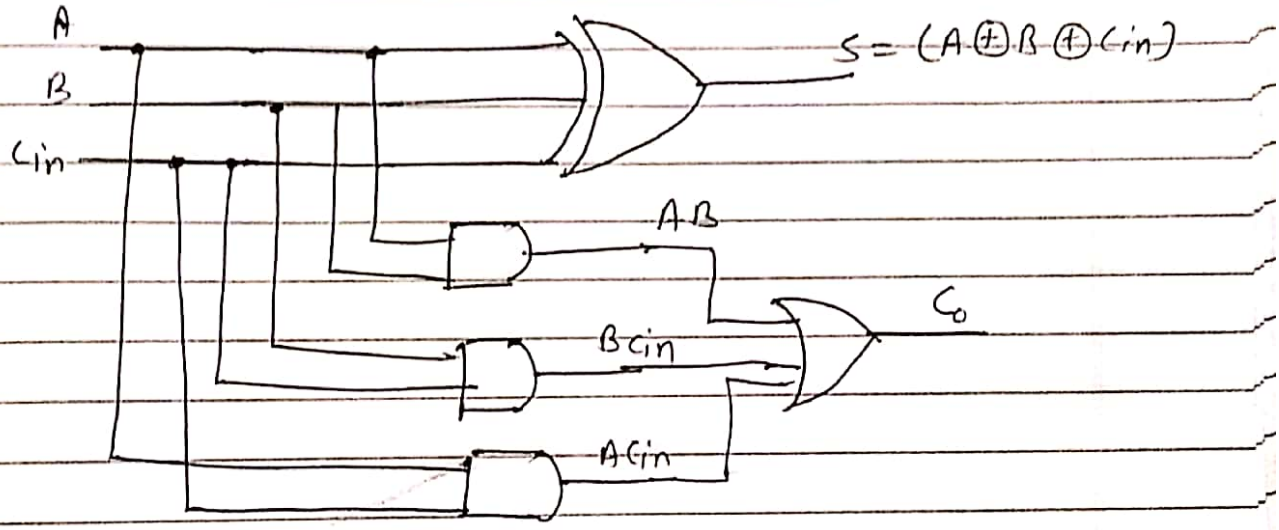
$$\boxed{S = C_{in} \oplus A \oplus B}$$

For carry :-

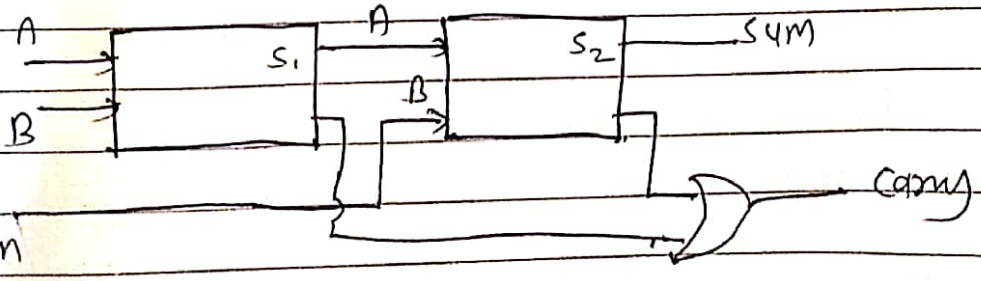


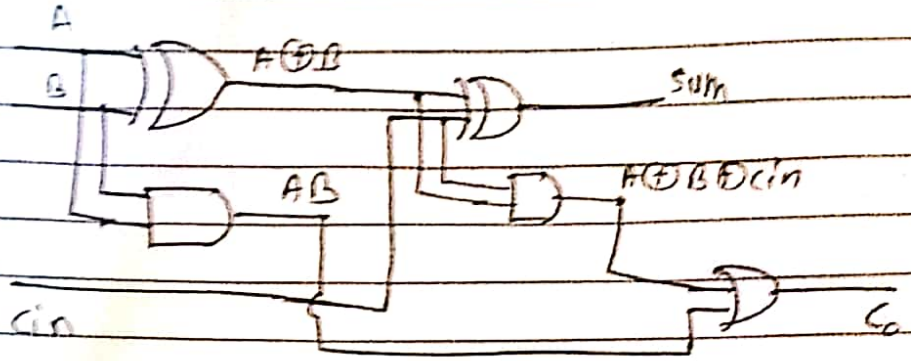
$C_0 = AB + ACin + BCin$

logic diagram.



* full adder using Half adder:-





$S = (A \oplus B) \oplus C_{in} = A \oplus B \oplus C_{in}$

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

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

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

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

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

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

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

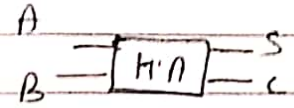
$= BC_{in} + AB + AC_{in}(\bar{B} + B)$

$C_o = BC_{in} + AB + AC_{in}.$

Application of full adder:

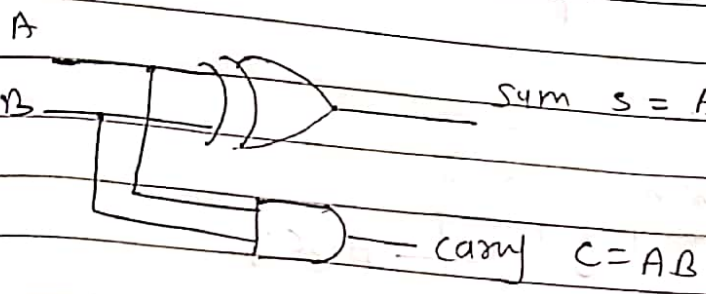
1) It is used in the digital diary
2) It is used in digital computers
3) It is used in arithmatic logic unit (ALU).
4) full adder acts as basic building block of 4 bit / 8 bit binary / BCD adder etc such as 7483.

comparision of half adder & full adder.

	Half adder	full adder.
1)		1) 
2)	It adds two bits	2) It add three bits.
3)	used as building block of full adder	3) basic building block of n-bit parallel adder
4)	It has no practical application	4) Practicaly used in n-bit adder, ALUs etc.

Date
Page

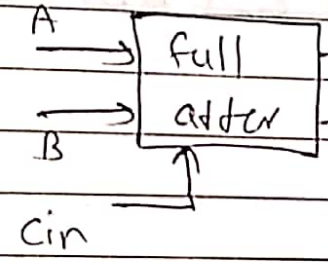
ckt diagram



2) full adder :-

Defⁿ :- full adder is a three input two output combinational logic circuit which can add three single bits applied at its input to produce sum & carry outputs.

block diagram :-



Truth Table :-

A	B	Cin	sum	carry (C_0)
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

* Binary subtractor :-

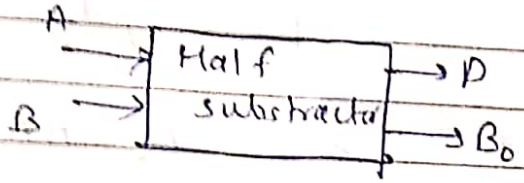
The types of binary substractor.

1) Half substractor 2) Full substractor.

1) Half substractor :-

Def^n :- Half substractor is a combinational circuit with two inputs and two outputs.

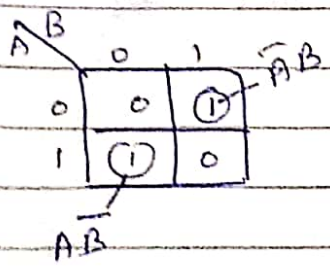
block diagram :-

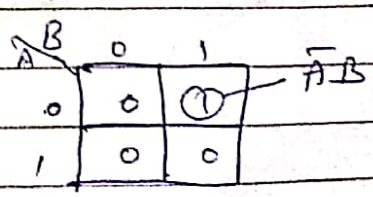


Truth Table :-

A	B	Difference (D) (A - B)	Borrow B_o
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Design using k-map :-

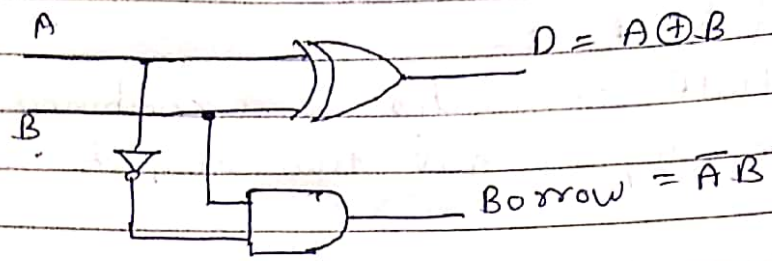




Difference $D = \bar{A}B + A\bar{B} = A \oplus B$

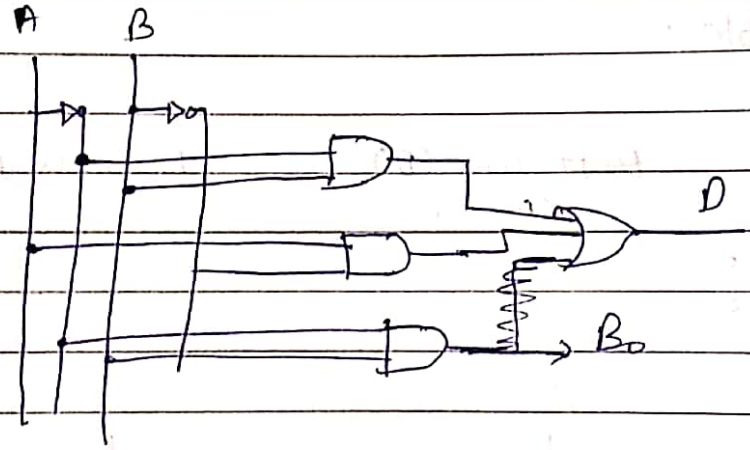
Borrow $B_0 = \bar{A}B$.

logic diagram :-



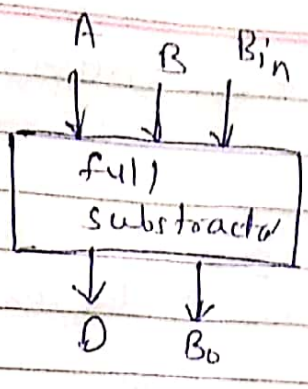
* Half substractor using Basic gates :-

$D = \bar{A}B + A\bar{B}$. and $B_0 = \bar{A}B$



* full substractor :-

Def^n :- The full substractor is a combinational circuit with three inputs A, B, B_{in} and two outputs D and B_0.



Truth Table :-

A	B	B_{in}	D $(A-B-B_{in})$	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

Design using k-map :-

For difference.

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

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

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

Ex-NOR — Ex-OR

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

Let $A \oplus B = C$

$D = B_{in}\bar{C} + \overline{B_{in}}C = B_{in} \oplus C$

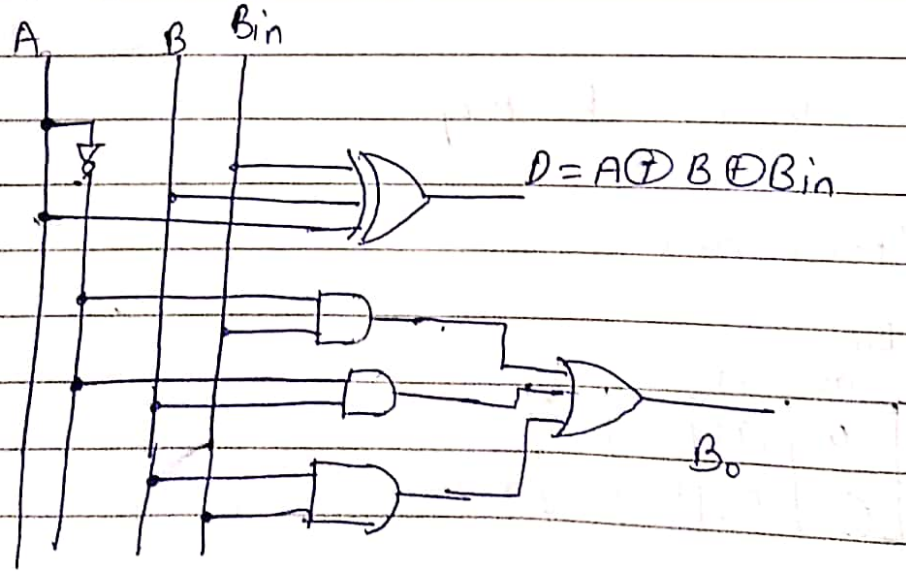
$D = B_{in} \oplus A \oplus B$

For Borrow output :-

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

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

logic diagram :-



* code converters:-

1) Binary to gray code converter:-

Truth Table:-

Decimal	Binary B_3 B_2 B_1 B_0	Gray: 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
10	1 0 1 0	1 1 1 1
11	1 0 1 1	1 1 1 0
12	1 1 0 0	1 0 1 0
13	1 1 0 1	1 0 1 1
14	1 1 1 0	1 0 0 1
15	1 1 1 1	1 0 0 0

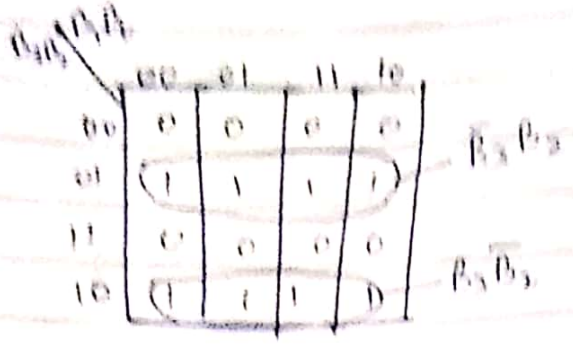
Design using K-Map.

For output G_3.

B_3B_2 \ B_1B_0	$\bar{B}_1\bar{B}_0$ 00	$\bar{B}_1B_0$ 01	B_1B_0 11	$B_1\bar{B}_0$ 10
$\bar{B}_3\bar{B}_2$ 00	0	0	0	0
$\bar{B}_3B_2$ 01	0	0	0	0
B_3B_2 11	1	1	1	1
$B_3\bar{B}_2$ 10	1	1	1	1

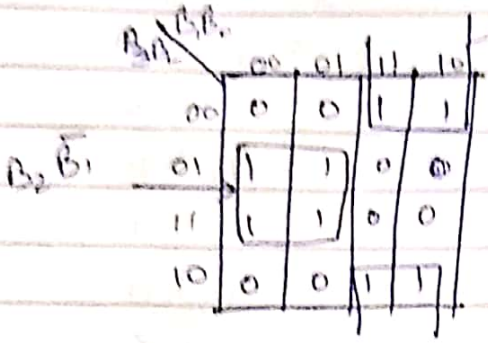
$G_3 = B_3$

For output G_2



$G_2 = \bar{B}_3B_2 + B_3\bar{B}_2 = B_3 \oplus B_2$

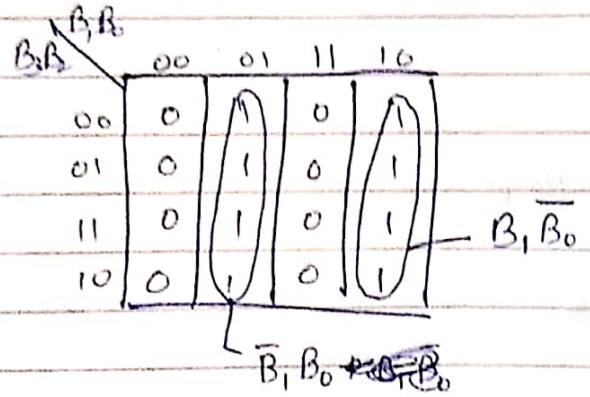
For output G_1



$G_1 = B_2\bar{B}_1 + \bar{B}_2B_1$

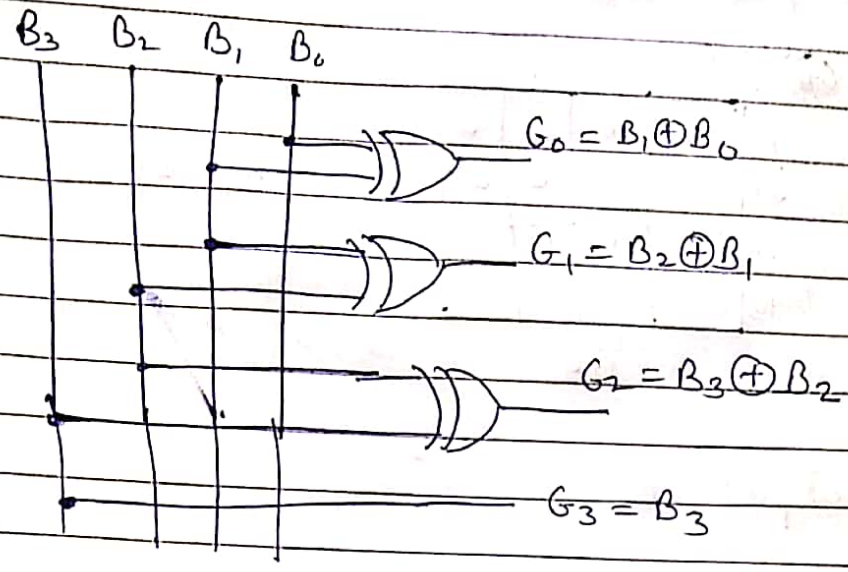
$G_1 = B_2 \oplus B_1$

For output $G_0 \rightarrow$



$G_0 = \bar{B}_1B_0 + B_1\bar{B}_0 = B_1 \oplus B_0$

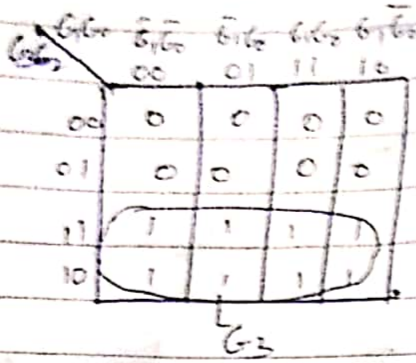
Logic diagram :-



2) Gray to binary code conversion.

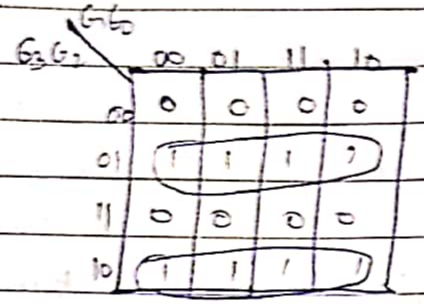
Decimal	Gray input G_3 G_2 G_1 G_0	Binary output B_3 B_2 B_1 B_0
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
3	0 0 1 1	0 0 1 0
2	0 0 1 0	0 0 1 1
6	0 1 1 0	0 1 0 0
7	0 1 1 1	0 1 0 0
5	0 1 0 1	0 1 0 1
4	0 1 0 0	0 1 1 0
12	1 1 0 0	0 1 1 1
13	1 1 0 1	1 0 0 0
15	1 1 1 1	1 0 0 1
14	1 1 1 0	1 0 1 0
10	1 0 1 0	1 0 1 1
11	1 0 1 1	1 1 0 0
9	1 0 0 1	1 1 0 1
8	1 0 0 0	1 1 1 1

For output B_3



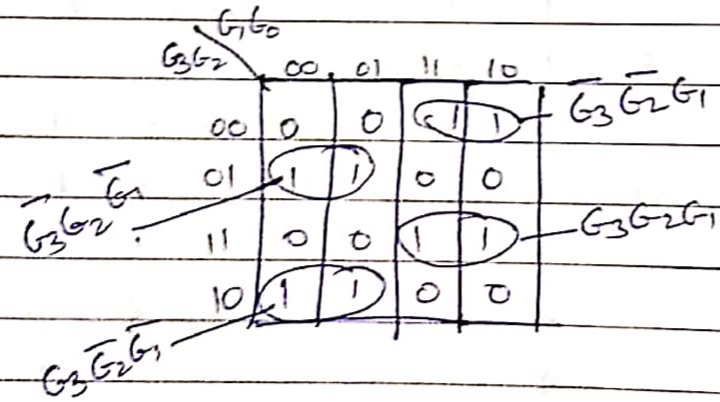
$B_3 = G_3$

For output B_2



$B_2 = G_3\bar{G}_2 + \bar{G}_3 G_2 = G_2 \oplus G_3$

For output B_1



$B_1 = \bar{G}_3\bar{G}_2G_1 + \bar{G}_3G_2\bar{G}_1 + G_3G_2G_1 + G_3\bar{G}_2\bar{G}_1$

$B_1 = \bar{G}_1(\bar{G}_3G_2 + G_3\bar{G}_2) + G_1(\bar{G}_3\bar{G}_2 + G_3G_2)$

EX-OR

EX-NOR

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

$B_1 = \bar{G_1}X + G_1\bar{X}$ where $X = G_3 \oplus G_2$

$B_1 = G_1 \oplus X = G_1 \oplus (G_3 \oplus G_2)$

$B_1 = G_1 \oplus G_2 \oplus G_3$

For output B_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

$B_0 = \bar{G_3}G_2\bar{G_1}G_0 + \bar{G_3}G_2\bar{G_1}\bar{G_0} + \bar{G_3}\bar{G_2}G_1\bar{G_0} + \bar{G_3}G_2G_1G_0$
$+ G_3G_2\bar{G_1}G_0 + G_3G_2G_1\bar{G_0} + G_3\bar{G_2}\bar{G_1}G_0 + G_3G_2G_1G_0$

$B_0 = \bar{G_1}G_0(\underbrace{\bar{G_3}G_2 + G_3\bar{G_2}}_{\text{Ex-OR}}) + \bar{G_1}G_0(\underbrace{\bar{G_3}\bar{G_2} + G_3G_2}_{\text{Ex-NOR}}) +$

$G_1G_0(\underbrace{\bar{G_3}G_2 + G_3\bar{G_2}}_{\text{Ex-OR}}) + G_1\bar{G_0}(\underbrace{\bar{G_3}\bar{G_2} + G_3G_2}_{\text{Ex-NOR}})$

$B_0 = \bar{G_1}\bar{G_0}(G_3 \oplus G_2) + \bar{G_1}G_0(\overline{G_3 \oplus G_2}) + G_1G_0(G_3 \oplus G_2)$
$+ G_1\bar{G_0}(\overline{G_2 \oplus G_2})$

$B_0 = (G_3 \oplus G_2)(\bar{G_1}\bar{G_0} + G_1G_0) + (\overline{G_3 \oplus G_2})(\bar{G_1}G_0 + G_1\bar{G_0})$

$B_0 = (G_3 \oplus G_2)\overline{(G_1 \oplus G_0)} + \overline{(G_3 \oplus G_2)}(G_1 \oplus G_0)$

let $x = G_3 \oplus G_2$ and $Y = (G_1 \oplus G_0)$

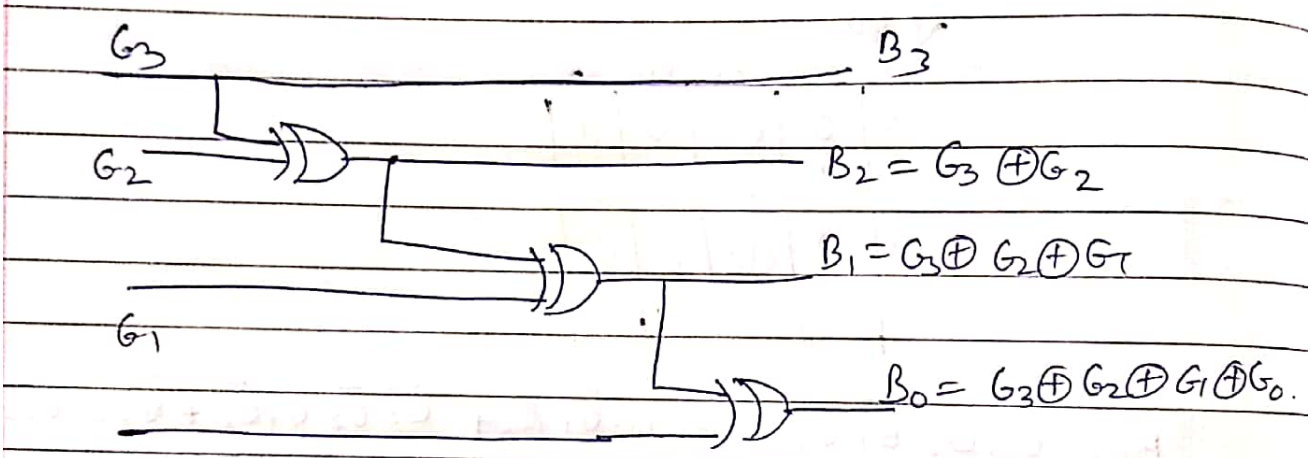
$$B_0 = x\bar{Y} + \bar{x}Y = x \oplus Y$$

substituting for x and y we get,

$$B_0 = (G_3 \oplus G_2) \oplus (G_1 \oplus G_0)$$

$$B_0 = G_3 \oplus G_2 \oplus G_1 \oplus G_0.$$

Logic diagram :-



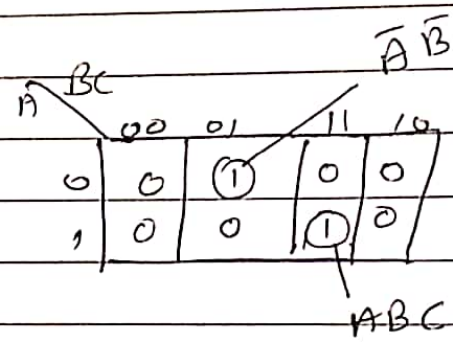
1) write the Boolean equation and draw logic diagram for the logic that will have output as Y and inputs A, B, C the logic performs the following operation.

1) Y = 1 when A = B = C = 1
2) Y = 1 when A = B = 0 & C = 1
3) Y = 0 for all other cases.

→ Truth Table:-

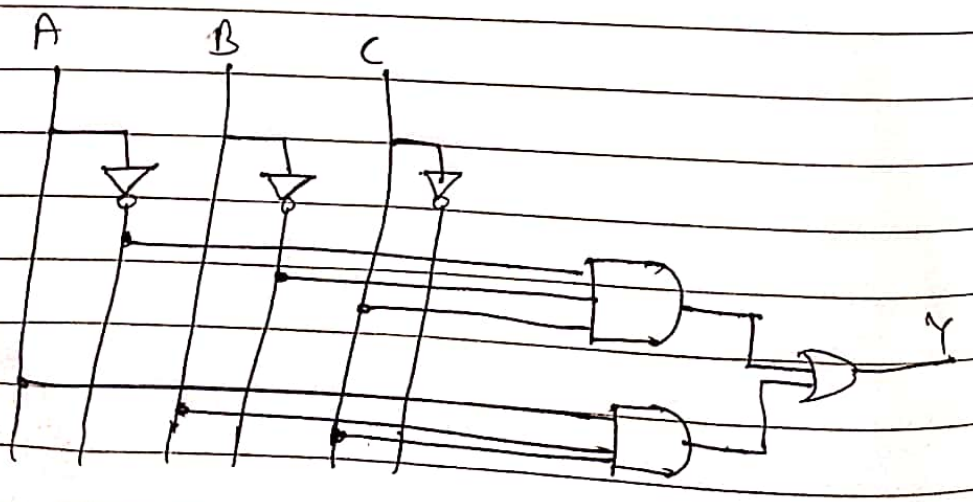
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

K- map:



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

logic diagram:-



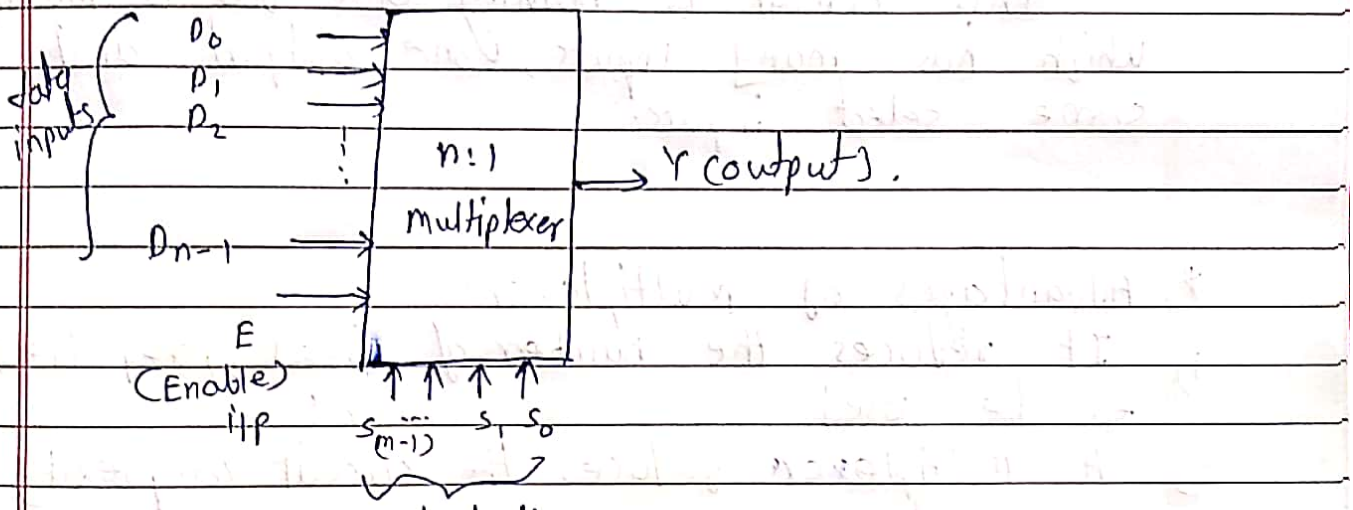
* Multiplexer :- (Data selector) :-

• Defⁿ :- A multiplexer is a combinational circuit with n data inputs one output and m select lines/ inputs which select one of the n data inputs and connect/ routes it to the output.

The selection of one of the n inputs is done with the help of the select inputs.

Multiplexer is special type of combinational circuit.

• Block diagram :- n to 1 multiplexer.



$D_0, D_1 \text{ --- } D_{n-1}$ → are the data inputs.

~~$S_{m-1} \ldots S_1$~~

$S_0, S_1 \text{ --- } S_{m-1}$ → are select lines.

Y → output.

E → Enable inputs. which is useful for cascating. it is generaly active low terminal, that means it will perform the required operation when it is low

• ~~The select line~~

To select n inputs we need m select lines such that $2^m = n$.

* need of multiplexers :-

1) In most of the electronic system, the digital data is available from more than one source. It is necessary to route this data over single line.

under such circumstances we require a circuit which selects one of the many sources at a time.

This circuit is nothing else but a multiplexer which has many inputs, one output and some select inputs.

* Advantages of multiplexer :-

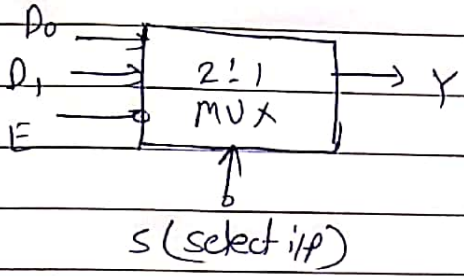
1) It reduces the number of wires, required to be used.
2) A multiplexer reduces the circuit complexity and cost.
3) we can implement many combinational circuit using mux
4) It simplifies the logic design
5) It does not need the k maps for simplification.

* Types of multiplexers :-

1) 2:1 multiplexer / MUX
2) 4:1 MUX
3) 8:1 MUX
4) 16:1 MUX
5) 32:1 MUX.

1) 2:1 multiplexers :-

- block diagram :-



- Truth Table :-

Enables	select i/p s	output Y.
0	x	0
1	0	D_0
1	1	D_1

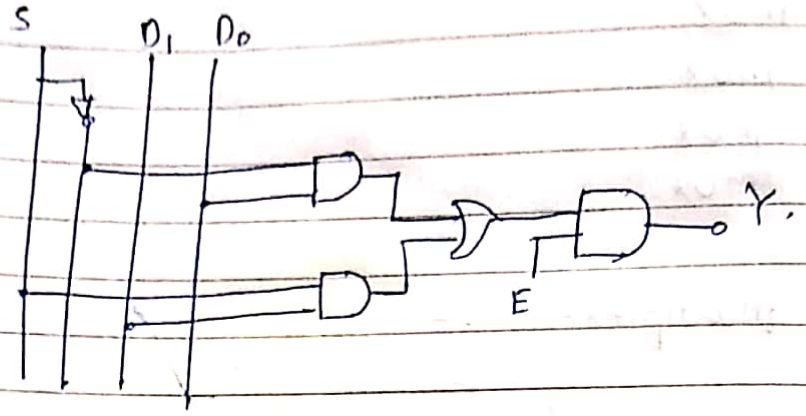
x → don't care.

- equation :-

$Y = E\bar{S}D_0 + ESD_1$

$Y = E(\bar{S}D_0 + SD_1)$

logic diagram:-



2) 4:1 MUX :-

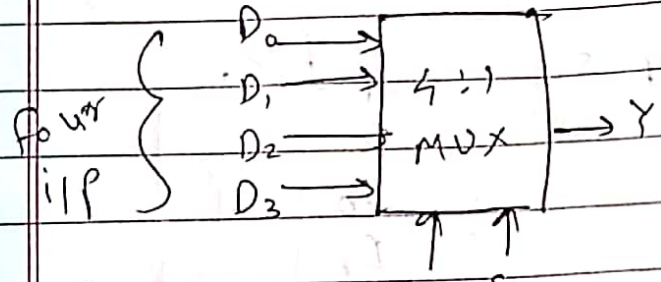
4:1 MUX →

4 → ~~inp~~ input

1 → output.

$n = 4$ hence number of select lines $m = 2$

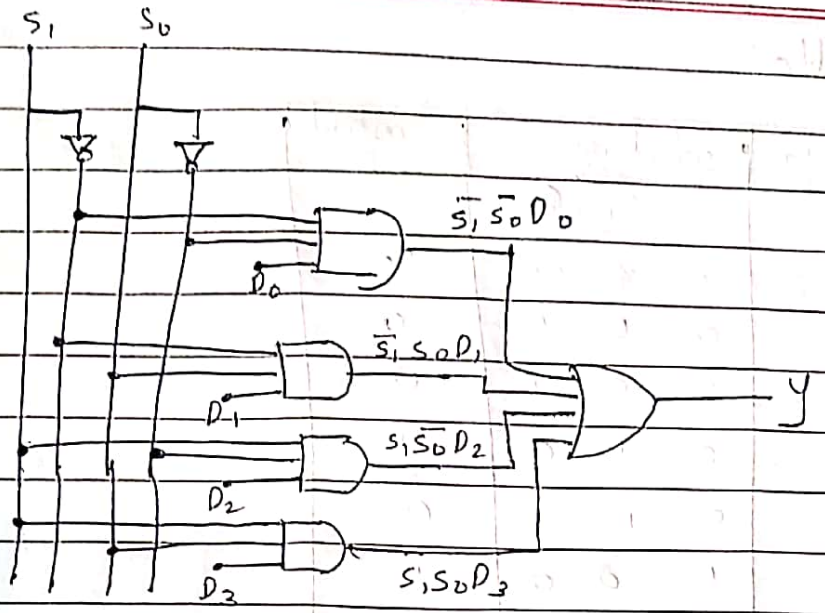
so that $2^m = n$



S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

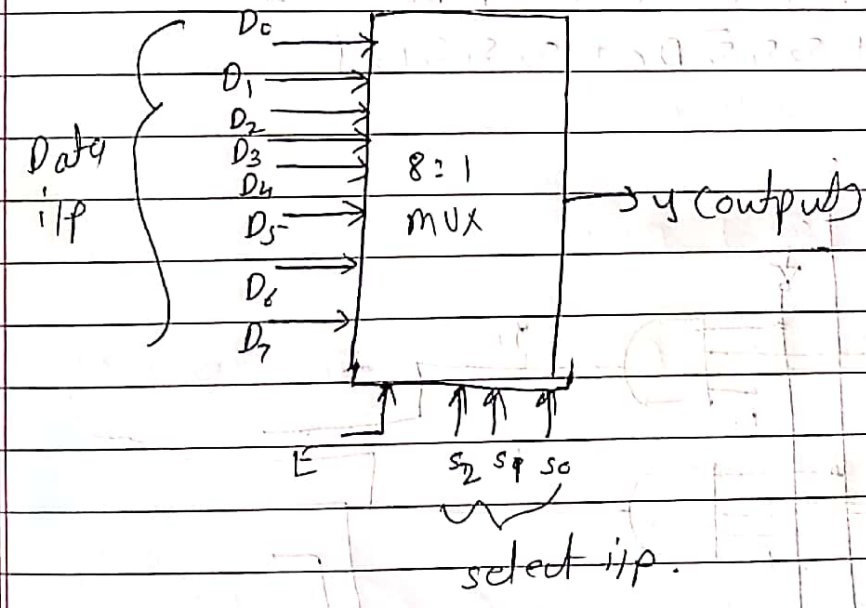
$$Y = \bar{S_1}\bar{S_0} D_0 + \bar{S_1} S_0 D_1 + S_1 \bar{S_0} D_2 + S_1 S_0 D_3$$

~~89[illegible]~~



$$Y = \bar{S}_1\bar{S}_0D_0 + \bar{S}_1S_0D_1 + S_1\bar{S}_0D_2 + S_1S_0D_3$$

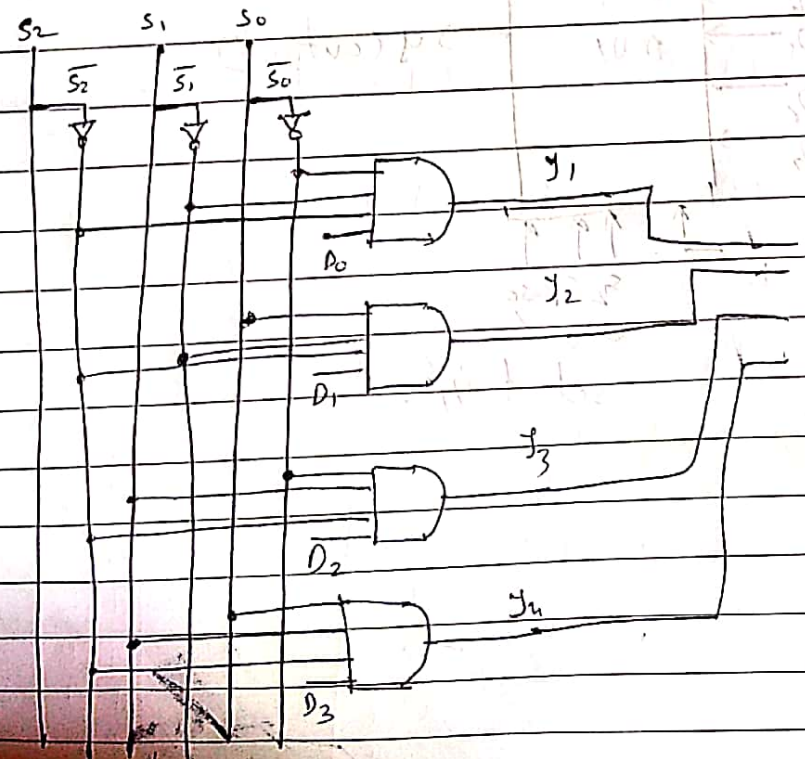
3) 8:1 multiplexer -

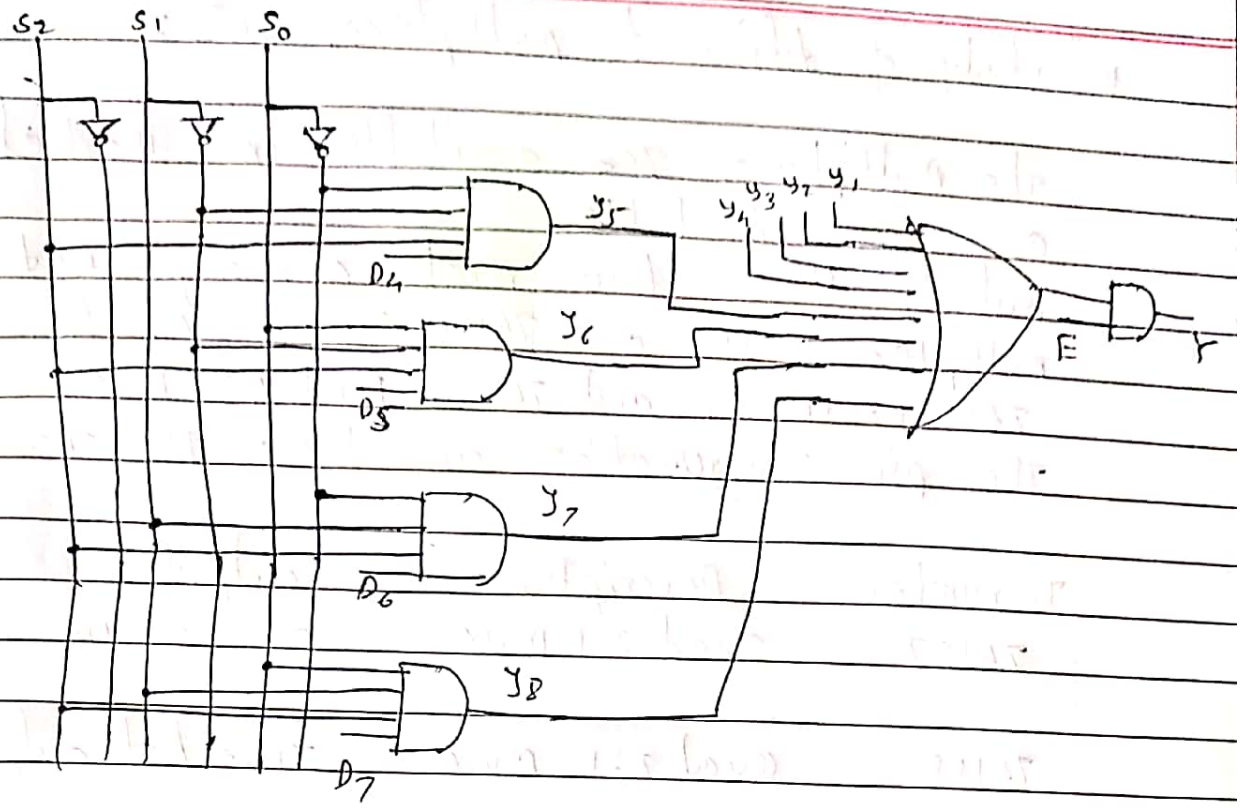


Truth Table:

Enable	select i/p			output
E	S_2	S_1	S_0	Y
0	x	x	0	x
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

$$Y = E \cdot [\bar{S_2}\bar{S_1}\bar{S_0}D_0 + \bar{S_2}\bar{S_1}S_0D_1 + \bar{S_2}S_1\bar{S_0}D_2 + \bar{S_2}S_1S_0D_3 + S_2\bar{S_1}\bar{S_0}D_4 + S_2\bar{S_1}S_0D_5 + S_2S_1\bar{S_0}D_6 + S_2S_1S_0D_7]$$





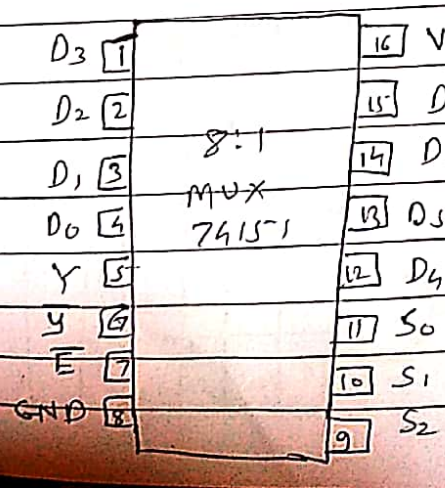
* Application of multiplexer:-

1) It is used as a data selector to select one out of many data inputs.
2) It is used for simplification of logic design.
3) In the data acquisition system.
4) In designing the combinational circuits.
5) In the D/A converters.
6) To minimize the number of connections in a logic circuit.

* study of different multiplexer IC's :=

The multiplexer ICs available in market are given in table.
out of these two most commonly used multiplexer ICs are 74150 (16:1 mux) & 74151 (8:1 mux) and 74153 (Dual 4:1 mux)
The pin configuration of these ICs shown in fig.

Ic number	Description	output.
74157	Quad 2:1 mux	same as i/p
74158	Quad 2:1 mux	Inverted output
74153	Dual 4:1 mux	same as i/p
74352	Dual 4:1 mux	Inverted output
74151A	8:1 mux	complementary o/p
74152	8:1 mux	Inverted o/p
74150	16:1 mux	Inverted o/p



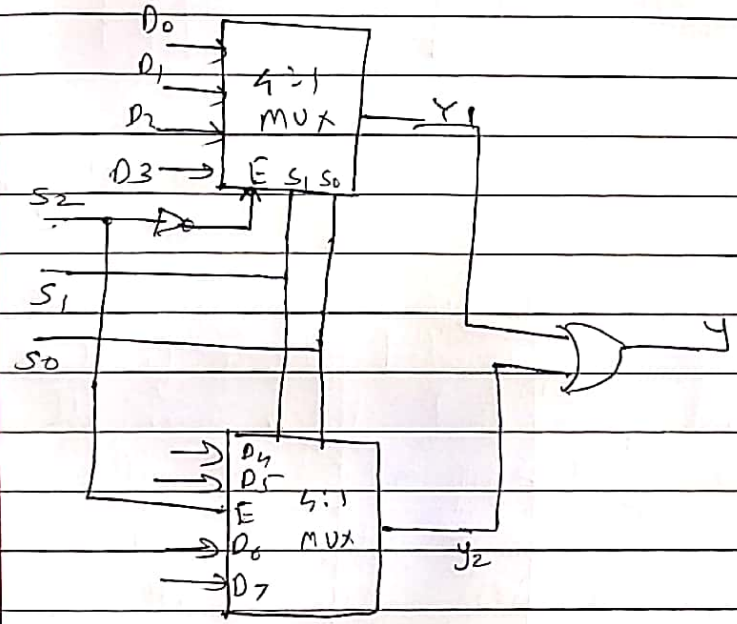
$D_0 - D_{15}$ → data i/p.
$S_0 - S_2$ → select i/p
$\bar{E}$ → enable i/p
$\bar{Y}$ → complemented o/p.

* pin configuration of 74151 *

* Multiplexer Tree:-
multiplexer having more number of i/p can be obtained by cascading two or more multiplexer with less number of inputs.
This called as multiplexer tree.

example:-
1) obtain an 8:1 multiplexer using two 4:1 multiplexers.

→



select i/p			output
S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

* Use of multiplexers in combinational logic design:

1) Advantage:-
   1) logic design is simplified
   2) It is not necessary to simplify the logic expression.
   3) It minimizes the number of ICs required to be used.

uses of MUX for combinational ckt. design:-

1) A truth table or logic expression in standard SOP or POS form is given to us.
2) We have to follow the design procedure given below to use MUX to for implementing the given logical expression.

example:-

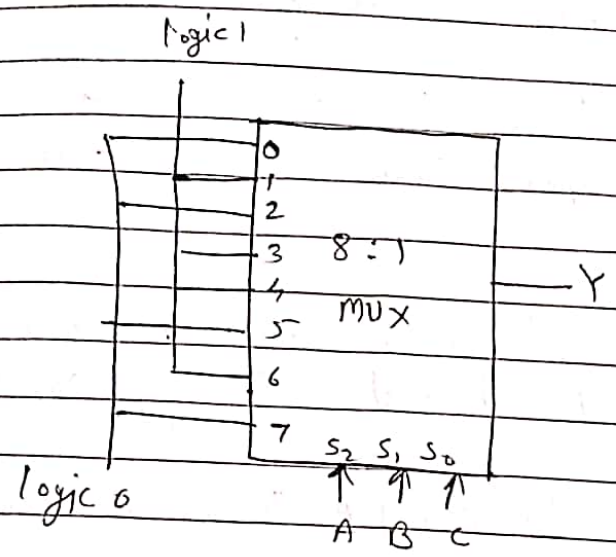
1) Realize the logic function of the truth table given in Table. using a MUX.

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

⟹ $Y = \text{output} = \Sigma m(1,3,4,6)$

2) connect the data i/p 1, 3, 4, 6 to logic 1 and the remaining to logic 0.

3) connect the i/p A, B, C, to select line S_2, S_1, S_0.



Home work :-

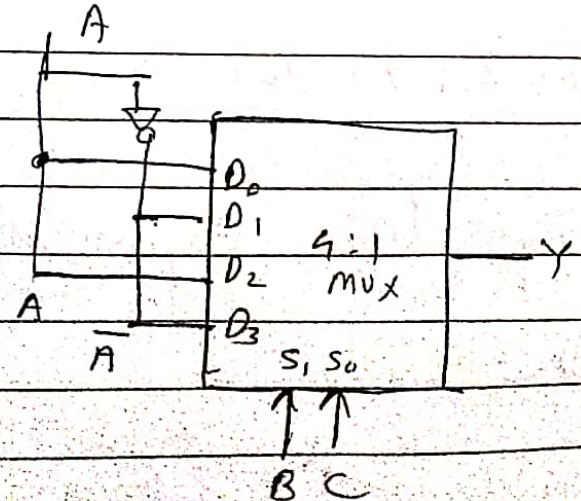
example :-

1) Implement the following expression using a multiplexer :-

$$f(A,B,C) = \Sigma m(0,2,4,6)$$

2) Implement the logic expression function $f(A,B,C) = \Sigma m(1,3,4,6)$ using 4:1 MUX.

→ $f(A,B,C) = \Sigma m(1,3,4,6)$



	D_0	D_1	D_2	D_3
$\bar{A}$	0	(1)	2	(3)
A	(4)	5	(6)	7

check each colounm in design table.

1) If both the minterm in colounm are not circled, then apply logic '0' to corresponding data i/p.

2) If only the minterm in second row is encircled then A should be applied to that data i/p.
[hence we should apply A to D_0, D_2 i/p]

3) If only minterm in first row is ~~circled~~ encircled then $\bar{A}$ should be connected to that data i/p.
[hence $\bar{A}$ to D_1 & D_3 i/p]

4) If both minterm colounm are ~~circled~~ encircled then apply a logic 1 to the corresponding data i/p. Note that there is no such column in our implementation table.

1) <u>Example 1</u> - Assignment:

$f(A, B, C, D) = \sum m(2, 4, 5, 7, 10, 14)$.

2) Implement following logic expression using 16:1 mux

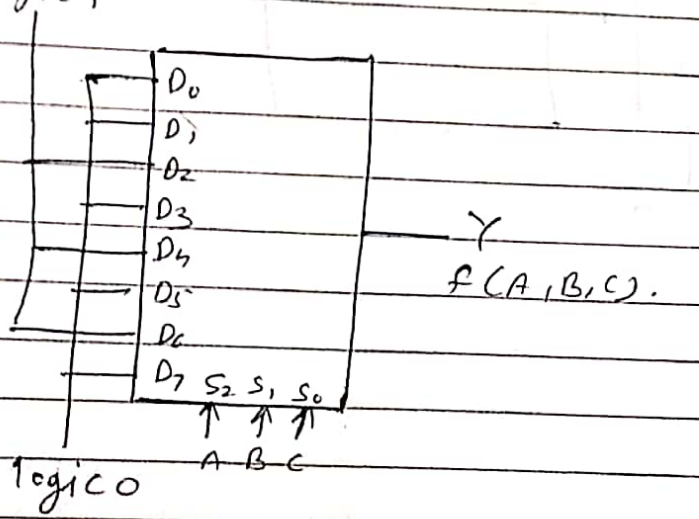
$Y = \sum m(0, 3, 5, 7, 8, 9, 12, 13)$

3) Implement following function 4:1 MUX
$Y = \sum m(0,1,3)$

4) Implementing a standard POS expression using multiplexer.

5) Implement the following logic function using the 8:1 mux.
$f(A,B,C) = \Pi M(0,1,3,5,7)$.

→ connect 0,1,3,5,7 to logic 0



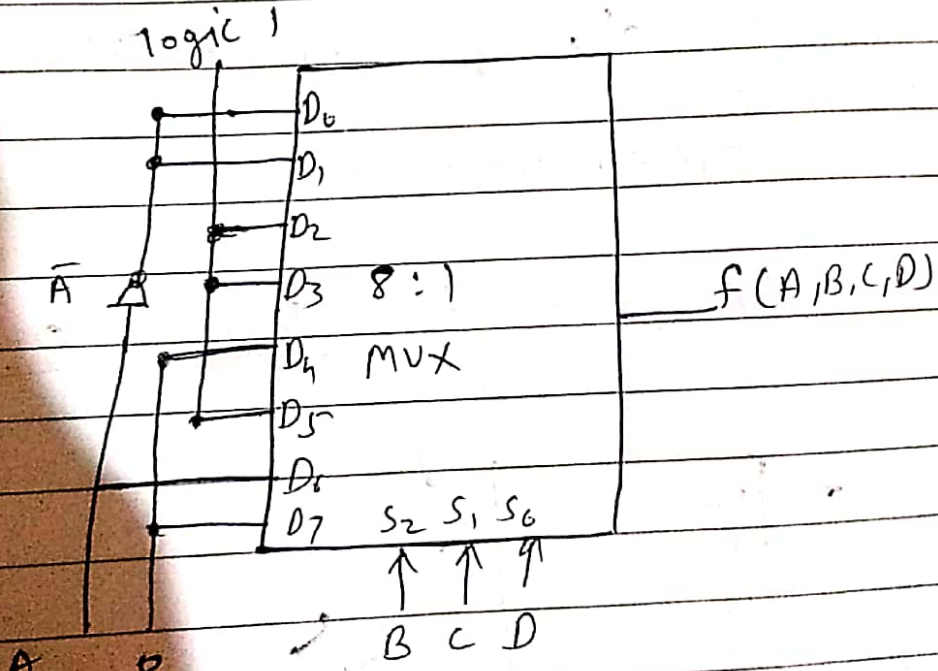
6) Implementation of Boolean SOP expression with Don't care condition.

1) Implement the following Boolean expression using 8:1 MUX

$f(A,B,C,D) = \sum m(1,3,5,10,11,13,14) + d(0,2)$

The don't care condition are assumed logic 1

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	(0)	(1)	(2)	(3)	4	(5)	6	7
A	8	9	(10)	(11)	12	(13)	(14)	15
	$\bar{A}$	$\bar{A}$	1	1	0	1	A	0



- It can be used as Boolean function generator.
- Typical average propagation delay time data input to W output 12.5 nS.
- Typical power dissipation 30 mW.

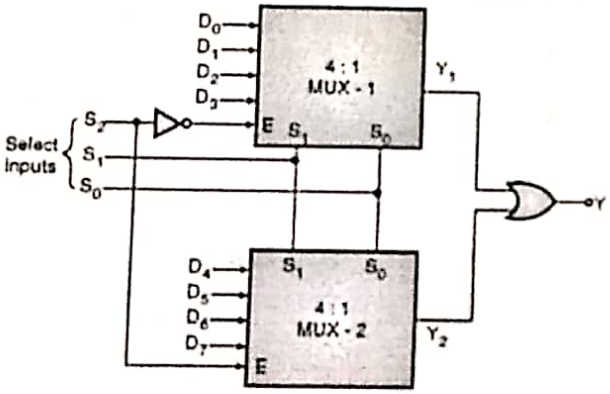
5.6 Multiplexer Tree :

- The multiplexers having more number of inputs can be obtained by cascading two or more multiplexers with less number of inputs.
- This is called as a multiplexer tree. This concept will be clear after solving the following examples.

Ex. 5.6.1 : Obtain an 8 : 1 multiplexer using two 4 : 1 multiplexers. **S-03, W-03, W-09, 4 Marks**

Soln. :

- The cascading of two 4 : 1 multiplexer results in 8 : 1 multiplexer as shown in Fig. P. 5.6.1.
- There are in all eight data inputs (D_0 through D_7).
- The select lines S_1 and S_0 of both 4 : 1 multiplexers are connected in parallel whereas a third select input S_2 is used for enabling one multiplexer at a time.
- S_2 is connected directly to the enable (E) terminal of MUX-1 whereas $\overline{S}_2$ is connected to the enable terminal of MUX-2.



(C-424) **Fig. P. 5.6.1 : 8 : 1 multiplexer by cascading two 4 : 1 multiplexers**

(C-6186) **Table P. 5.6.1 : Truth table**

Select Inputs			Output	
S_2	S_1	S_0	Y	
0	0	0	D_0	MUX 1 is enable
0	0	1	D_1	
0	1	0	D_2	
0	1	1	D_3	
1	0	0	D_4	MUX 2 is enable
1	0	1	D_5	
1	1	0	D_6	
1	1	1	D_7	

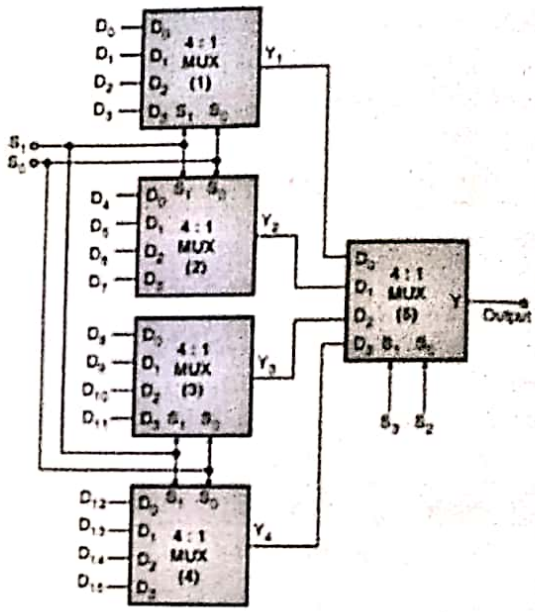
- The outputs of the two multiplexers are ORed to obtain the final output Y.

Ex. 5.6.2 : Implement a 16 : 1 multiplexer using 4 : 1 multiplexers. **S-14, W-15, I-Scheme: W-18, 4 Marks**

Soln. : Refer Fig. P. 5.6.2 for implementation.

Description :

- The select inputs S_1 and S_0 of the multiplexers 1, 2, 3 and 4 are connected together.
- The select inputs S_3 and S_2 are applied to the select inputs S_1 and S_0 of MUX-5.
- The outputs Y_1, Y_2, Y_3, Y_4 are applied to the data inputs D_0, D_1, D_2 and D_3 of MUX-5 as shown in Fig. P. 5.6.2.



(C-425) **Fig. P. 5.6.2 : 16 : 1 multiplexer using 4 : 1 multiplexers**

- The operation can be summarized using Table P. 5.6.2.

(C-6187) **Table P. 5.6.2 : Summary of operation**

Select Inputs				Mux. Outputs				Final Output	
S_3	S_2	S_1	S_0	Y_1	Y_2	Y_3	Y_4	Y	
0	0	0	0	D_0	D_4	D_8	D_{12}	D_0	$S_3S_2 = 00$ ∴ MUX-5 selects Y_1
0	0	0	1	D_1	D_5	D_9	D_{13}	D_1	
0	0	1	0	D_2	D_6	D_{10}	D_{14}	D_2	
0	0	1	1	D_3	D_7	D_{11}	D_{15}	D_3	
0	1	0	0	D_0	D_4	D_8	D_{12}	D_4	$S_3S_2 = 01$ ∴ MUX-5 selects Y_2
0	1	0	1	D_1	D_5	D_9	D_{13}	D_5	
0	1	1	0	D_2	D_6	D_{10}	D_{14}	D_6	
0	1	1	1	D_3	D_7	D_{11}	D_{15}	D_7	
1	0	0	0	D_0	D_4	D_8	D_{12}	D_8	$S_3S_2 = 10$ ∴ MUX-5 selects Y_3
1	0	0	1	D_1	D_5	D_9	D_{13}	D_9	
1	0	1	0	D_2	D_6	D_{10}	D_{14}	D_{10}	
1	0	1	1	D_3	D_7	D_{11}	D_{15}	D_{11}	
1	1	0	0	D_0	D_4	D_8	D_{12}	D_{12}	$S_3S_2 = 11$ ∴ MUX-5 selects Y_4
1	1	0	1	D_1	D_5	D_9	D_{13}	D_{13}	
1	1	1	0	D_2	D_6	D_{10}	D_{14}	D_{14}	
1	1	1	1	D_3	D_7	D_{11}	D_{15}	D_{15}	

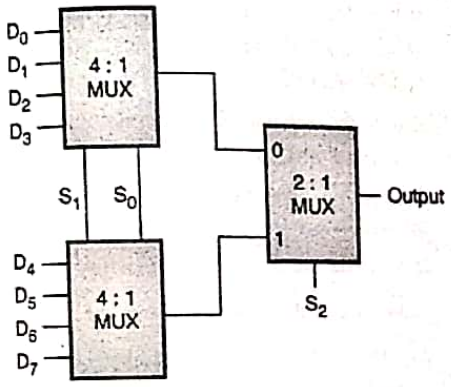
Ex. 5.6.3 : Design 8 : 1 MUX using 2 : 1 MUX and 4 : 1 MUX. **S-15, 4 Marks**

Soln. :

8 : 1 MUX using 2 : 1 MUX and 4 : 1 MUX :

Truth table for 8 : 1 MUX

Enable	Select inputs			Output
E	S_2	S_1	S_0	Y
0	X	X	X	0
1	0	0	0	D_0
1	0	0	1	D_1
1	0	1	0	D_2
1	0	1	1	D_3
1	1	0	0	D_4
1	1	0	1	D_5
1	1	1	0	D_6
1	1	1	1	D_7



(C-5094) **Fig. P. 5.6.3**

5.7 Use of Multiplexers in Combinational Logic Design :

Multiplexers ICs for 2 : 1 , 4 : 1, 8 : 1 and 16 : 1 multiplexers are available. We can use them to implement the given Boolean expressions representing the combinational circuits. Thus it is possible to design and implement many combinational circuits by using multiplexer and a few logic gates.

Advantages :

Advantages of using multiplexers for logic design are as follows :

1. Logic design is simplified.
2. It is not necessary to simplify the logic expression.
3. It minimizes the number of ICs required to be used.

Use of MUX for combinational circuit design :

1. A truth table or logic expression in standard SOP or POS form is given to us.
2. We have to follow the design procedure given below to use MUX for implementing the given logical expression.

Design procedure :

Step 1 : Identify the decimal number corresponding to each minterm in the given expression as illustrated below.

If $Y = \underbrace{\bar{A}\bar{B}\bar{C}}_{0} + \underbrace{\bar{A}\bar{B}C}_{5} + \underbrace{\bar{A}BC}_{7}$ (C-6264)

Step 2 : The input data lines of the multiplexer, corresponding to these numbers (0, 5 and 7) are connected to logic 1 level.

Step 3 : All the other input data lines of multiplexer are connected to logic 0 level.

Step 4 : The inputs (A, B, C) are to be connected to the select inputs.

The following example will make this concept clear.

Ex. 5.7.1 : Realize the logic function of the truth table given in Table P. 5.7.1 using a multiplexer.

(C-7617) **Table P. 5.7.1**

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

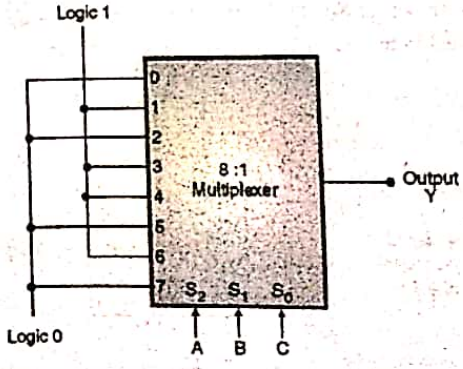
Soln. :

Step 1 : Express the output Y in the standard SOP form $Y = \Sigma m\ (1, 3, 4, 6)$

Step 2 : Connect the data inputs 1, 3, 4 and 6 to logic 1 and the remaining to logic 0.

Step 3 : Connect the inputs A, B, C to the select lines S_2 S_1 and S_0 respectively.

Step 4 : Draw the logic diagram as shown in Fig. P. 5.7.1.



(C-429) **Fig. P. 5.7.1 : Logic diagram for Ex. 5.7.1**

Ex. 5.7.2 : Implement the following expression using a multiplexer :

$f(A, B, C) = \Sigma m\ (0, 2, 4, 6)$

Soln. :

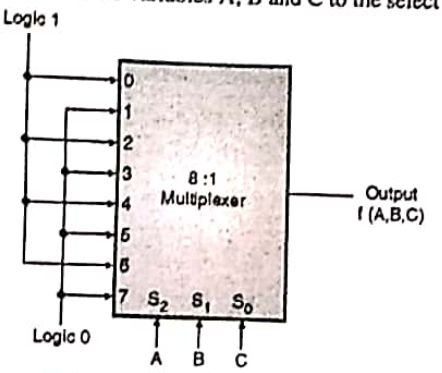
Since there are three variables, the multiplexer having three select inputs should be used. Hence an 8 : 1 multiplexer as shown in Fig. P. 5.7.2 should be used.

Step 1 : Identify the decimal number corresponding to each minterm.

The decimal numbers corresponding the minterms are 0, 2, 4 and 6.

Step 2 : Connect the data input lines 0, 2, 4 and 6 to logic 1 and the remaining lines (1, 3, 5, 7) to logic 0 as shown in Fig. P. 5.7.2.

Step 3 : Connect the variables A, B and C to the select inputs.



(C-430) **Fig. P. 5.7.2 : Implementation of a logic expression using a multiplexer**

Note : We can use IC 74151 which is an 8 : 1 Multiplexer to implement Boolean equations using 8 : 1 MUX.

Ex. 5.7.3 : Implement the logic function f (A, B, C) = $\sum$m (1, 3, 4, 6) using a 4 : 1 multiplexer.

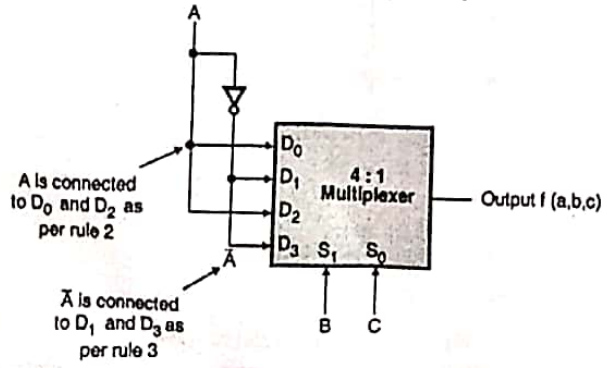
Soln. :

The logic function to be implemented is

$$f(A, B, C) = \sum m(1, 3, 4, 6)$$

Step 1 : Apply two variables B and C to the select inputs :

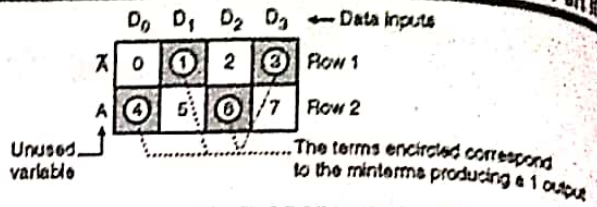
As shown in Fig. P. 5.7.3(a), the two input variables B and C are applied to the select lines S_1 and S_0 respectively.



(C-431) **Fig. P. 5.7.3(a) : Logic diagram**

Step 2 : Write the design table :

The design table is shown in Fig. P. 5.7.3(b).



(C-432) **Fig. P. 5.7.3(b) : Design table**

- The data inputs D_0 to D_3 have been written at the top of the table and the two possible values A and $\bar{A}$ of the unused variable A have been written.
- In the eight boxes we enter the decimal numbers corresponding to the minterms (0 to 7) serially.
- Encircle those minterms corresponding to which the output is 1 (minterms 1, 3, 4, 6).

Step 3 : Check each column in the design table :

The columns of design table are inspected using the following rules :

Rule 1 : If both the minterms in a column are not circled, then apply logic 0 to the corresponding data input. Note that there is no such column in our implementation table.

Rule 2 : If only the minterm in the second row is encircled (see columns 1 and 3) then "A" should be applied to that data input. **Hence we should apply A to the D_0 and D_2 inputs.**

Rule 3 : If only the minterm in the first row is encircled, (see columns 2 and 4), then $\bar{A}$ should be connected to that data input. **Hence we should apply $\bar{A}$ to the D_1 and D_3 inputs.**

Rule 4 : If both the minterms in a column are encircled, then apply a logic 1 to the corresponding data input. Note that there is no such column in our implementation table.

Step 4 : Draw the logic diagram :

The logic diagram is as shown in Fig. P. 5.7.3(a).

Note : We can use IC 74153 which is a dual 4 : 1 Multiplexer in order to implement the Boolean expressions using 4 : 1 MUX.

Ex. 5.7.4 : Implement the following Boolean function using 8 : 1 multiplexer :

$f(A, B, C, D) = \sum m(2, 4, 5, 7, 10, 14)$

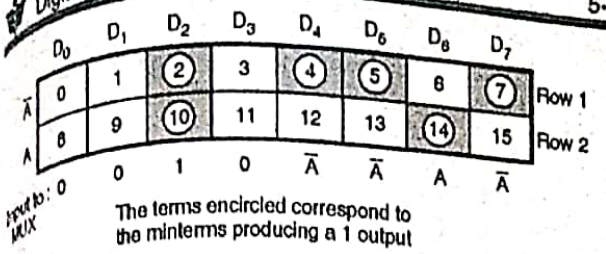
Soln. :

Step 1 : Apply the variables B, C and D to the select inputs :

As shown in Fig. P. 5.7.4(b), the three variables B, C and D are connected to the select lines S_2, S_1 and S_0 respectively.

Step 2 : Write the design table :

The design table is as shown in Fig. P. 5.7.4(a).



(C-434) **Fig. P. 5.7.4(a) : Design table**

- In the sixteen boxes we have entered the decimal numbers corresponding to the minterms 0 to 15 serially.
- Encircle those minterms which correspond to an output = 1 (minterms 2, 4, 5, 7, 10, 14).

Step 3 : Inspect each column in the design table :

Rule 1 : If both the minterms in a column are not circled, then apply logic 0 to the corresponding data input. Note that there is no such column in our implementation table.

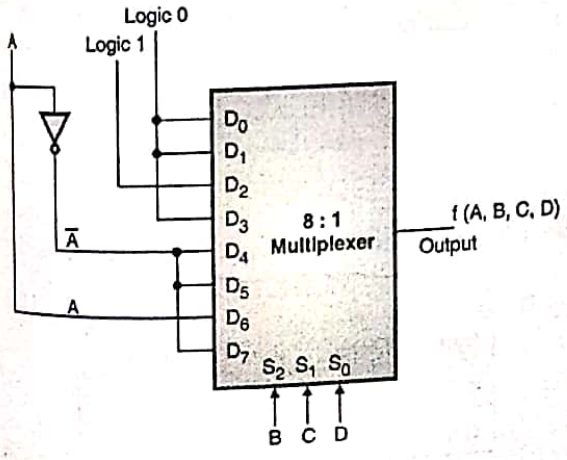
Rule 2 : If only the minterm in the second row is encircled then "A" should be applied to that data input. **Hence we should apply A to the D_6 input.**

Rule 3 : If only the minterm in the first row is encircled, then $\overline{A}$ should be connected to that data input. **Hence we should apply $\overline{A}$ to the D_4, D_5 and D_7 inputs.**

Rule 4 : If both the minterms in a column are encircled, then apply a logic 1 to the corresponding data 1 input. Note that there is no such column in our implementation table.

Step 4 : Draw the logic diagram :

The logic diagram is as shown in Fig. P. 5.7.4(b).



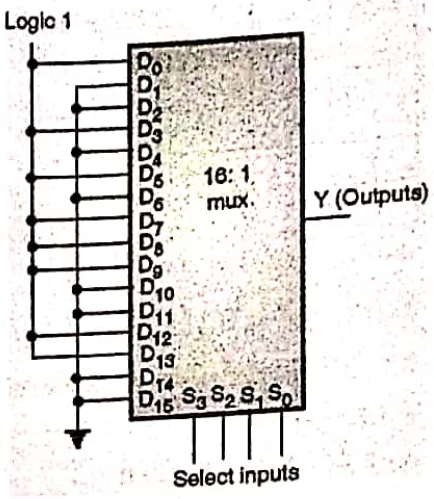
(C-435) **Fig. P. 5.7.4(b) : Logic diagram**

Ex. 5.7.5 : Implement the following logic expression using 16 : 1 mux :
$Y = \Sigma m (0, 3, 5, 7, 8, 9, 12, 13)$
S-06, 4 Marks

Soln. :

$$Y = m_0 + m_3 + m_5 + m_7 + m_8 + m_9 + m_{12} + m_{13}$$

- Connect the data inputs corresponding to these minterms to logic 1. That means connect D_0, D_3, D_5, D_7, D_8, D_9, D_{12} and D_{13} to logic 1 and connect remaining data inputs to 0.
- The required logic diagram is shown in Fig. P. 5.7.5.



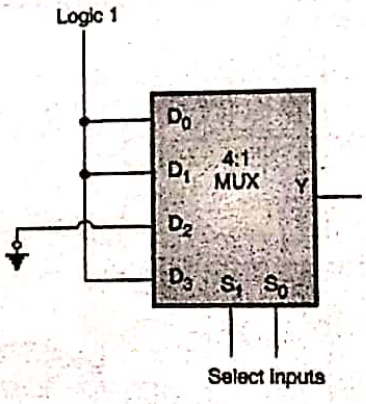
(C-2612) **Fig. P. 5.7.5**

Ex. 5.7.6 : Implement the following function using 4 : 1 Mux : $Y = \Sigma m (0, 1, 3)$
W-07, W-11, 4 Marks

Soln. :

Given : $Y = \Sigma m (0, 1, 3) = m_0 + m_1 + m_3$

The implementation of this equation using a 4 : 1 MUX is shown in Fig. P. 5.7.6.



(C-2615) **Fig. P. 5.7.6**

Ex. 5.7.7 : Design an 8 : 1 MUX using NAND gates.
W-05, 4 Marks

Soln. :

- The truth table of 8 : 1 multiplexer is as given in Table P. 5.7.7.

Table P. 5.7.7 : Truth table of 8 : 1 multiplexer

Select inputs			Output
S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

- The Boolean expression for the output is given by,

$$Y = \overline{S}_2 \overline{S}_1 \overline{S}_0 D_0 + \overline{S}_2 \overline{S}_1 S_0 D_1 + \overline{S}_2 S_1 \overline{S}_0 D_2 + \overline{S}_2 S_1 S_0 D_3 + S_2 \overline{S}_1 \overline{S}_0 D_4 + S_2 \overline{S}_1 S_0 D_5 + S_2 S_1 \overline{S}_0 D_6 + S_2 S_1 S_0 D_7$$

Let $Y = A + B + C + D + E + F + G + H$

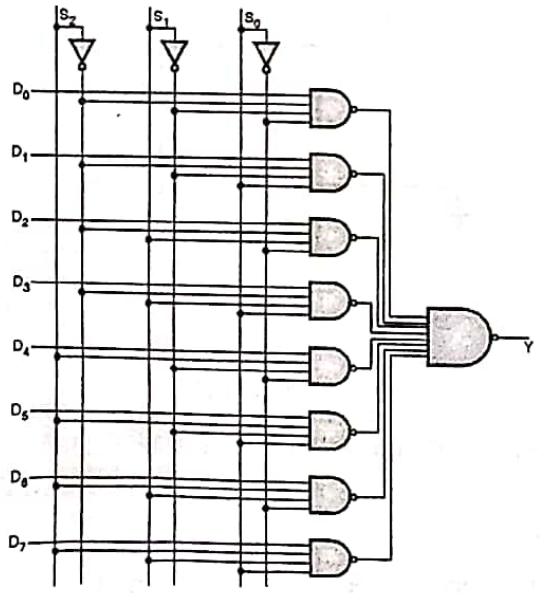
$$Y = \overline{A \cdot B \cdot C \cdot D \cdot E \cdot F \cdot G \cdot H}$$

... De Morgan's theorem.

Substituting the values of A,B,C,D,E,F,G,H we get,

$$Y = \overline{S_2 S_1 S_0 D_0} \cdot \overline{S_2 S_1 S_0 D_1} \cdot \overline{S_2 S_1 S_0 D_2} \cdot \overline{S_2 S_1 S_0 D_3} \cdot \overline{S_2 S_1 S_0 D_4} \cdot \overline{S_2 S_1 S_0 D_5} \cdot \overline{S_2 S_1 S_0 D_6} \cdot \overline{S_2 S_1 S_0 D_7}$$

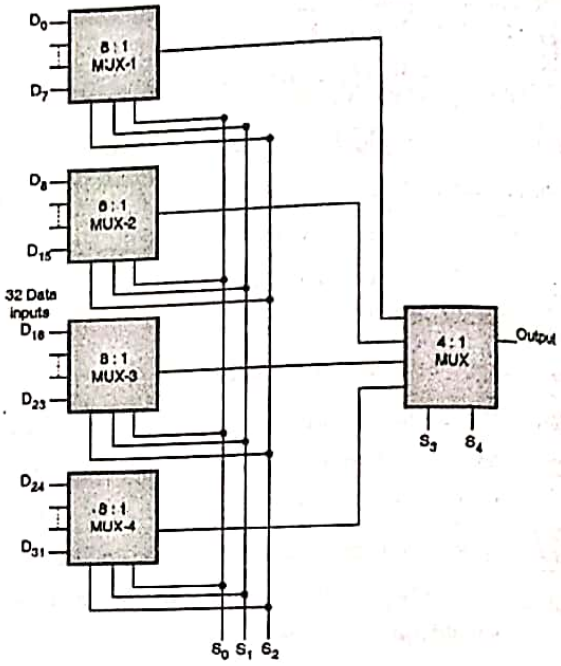
- The above expression is implemented using NAND gates as shown in Fig. P. 5.7.7.



(C-2611) **Fig. P. 5.7.7 : 8 : 1 multiplexer using NAND gates**

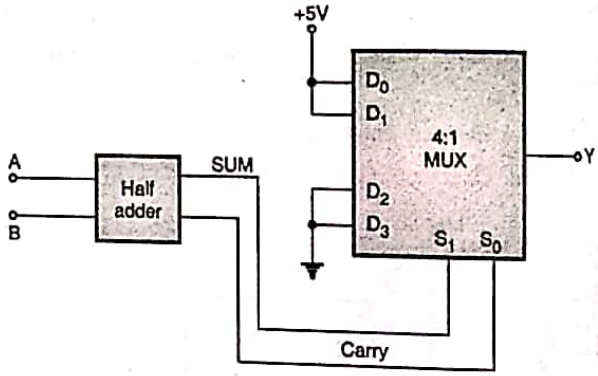
Ex. 5.7.8 : Design a 32 : 1 MUX using 8 : 1 multiplexers. **W-05, W-08, 4 Marks**

Soln. : Please Refer Fig. P. 5.7.8 for solution.



(C-2614) **Fig. P. 5.7.8 : 32 : 1 MUX using 8 : 1 MUX**

Ex. 5.7.9 : In the given Fig. P. 5.7.9 signals AB change from 00 to 11. Write the truth table for the system. **W-07, 4 Marks**



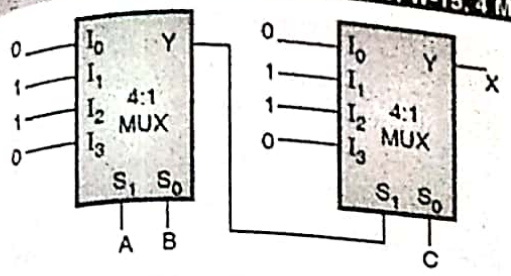
(C-2616) **Fig. P. 5.7.9**

Soln. : The required truth table is as shown in Table P. 5.7.9.

Table P. 5.7.9

Inputs		Output of H.A.		MUX output
A	B	S	C	Y
0	0	0	0	1
0	1	1	0	0
1	0	1	0	0
1	1	0	1	1

Ex. 5.7.10 : In the circuit shown in Fig. P. 5.7.10, what will be the output X ? **S-09, W-15, 4 Marks**



(C-2617) **Fig. P. 5.7.10**

Soln. :

Select inputs			Output of MUX 1	Output of MUX 2
A	B	C	Y	X
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	0	1

Ex. 5.7.11 : Give the truth table of the full adder and implement it using multiplexer **W-12, 4 Marks**

Soln. :

Step 1 : Write the truth table of full adder :

Table P. 5.7.11 : Truth table of a full adder

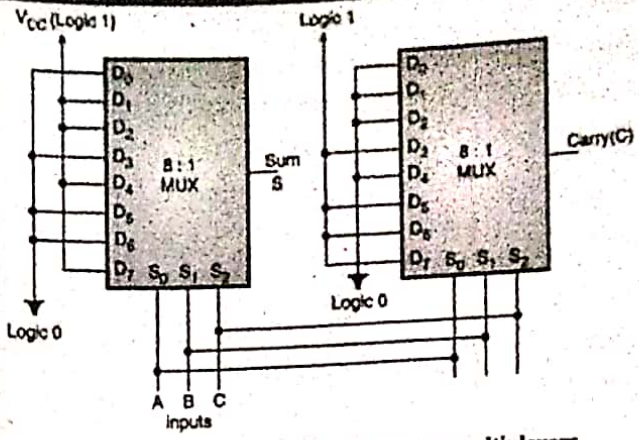
Inputs			Output	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Step 2 : Implementation :

From the truth table we get,

Sum (S) = Σm (1, 2, 4, 7) and Carry (C) = Σm (3, 5, 6, 7).

We need two 8:1 multiplexers. Fig. P. 5.7.11 shows the implementation.

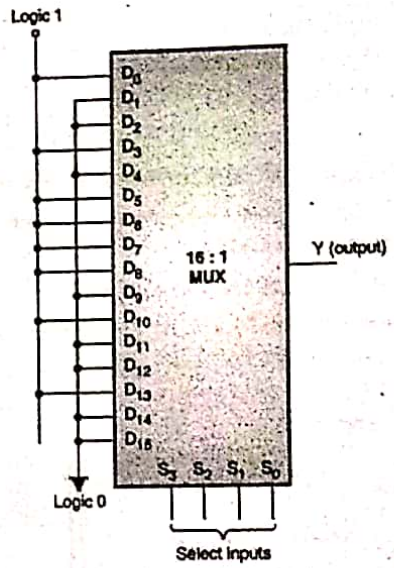


(C-3254) **Fig. P. 5.7.11 : Full adder using multiplexers**

Ex. 5.7.12 : List any four applications of multiplexer and implement the following logic expression using 16 : 1 MUX.
$Y = \Sigma m$ (0, 3, 5, 6, 7, 10, 13). **S-15, 6 Marks**

Soln. : Refer section 5.4 for applications of multiplexer.

$Y = \Sigma m$ (0, 3, 5, 6, 7, 10, 13)



(C-5086) **Fig. P. 5.7.12**

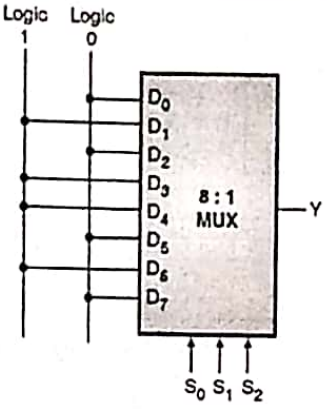
Ex. 5.7.13 : Realize the logic function of the truth table given below using a multiplexer. **S-16, 6 Marks**

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Soln. :

Step 1 : Implementation using 8 : 1 MUX :

Fig. P. 5.7.13 shows implementation using 8 : 1 MUX.



(C-5492) **Fig. P. 5.7.13**

5.7.1 Implementing a Standard POS Expression using Multiplexer :

Let us consider the Boolean expression in the standard POS form and its implementation using a multiplexer.

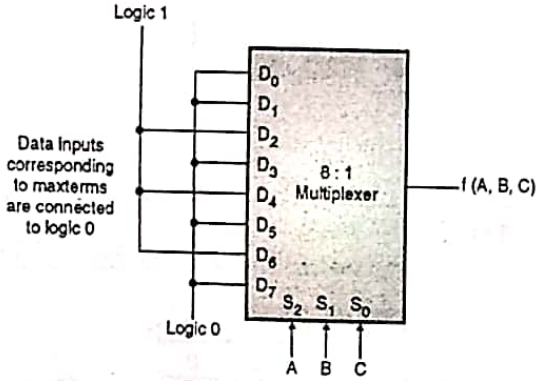
Ex. 5.7.14 : Implement the following logic function using the 8 : 1 multiplexer.

$f(A, B, C) = \Pi M (0, 1, 3, 5, 7)$

Soln. :

In the given expression, the maxterms have been specified.

- Hence we should connect the corresponding data inputs (D_0, D_1, D_3, D_5 and D_7) to logic 0.
- And connect the remaining data inputs to logic 1.
- Connect the input variables A, B, C to the select inputs S_2, S_1 and S_0 respectively. The logic diagram is as shown in Fig. P. 5.7.14.



(C-439) **Fig. P. 5.7.14 : Implementation of standard POS expression**

5.7.2 Implementation of Boolean SOP Expression with Don't Care Conditions :

- Sometimes the Boolean expressions are given with don't care conditions.
- We know that the don't care conditions can be treated as logic 1's or 0's.
- Ex. 5.7.15 illustrates the use of multiplexer to implement such equations.

Ex. 5.7.15 : Implement the following Boolean expression using an 8 : 1 multiplexer.

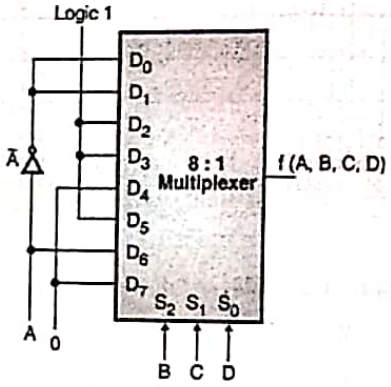
$f(A, B, C, D) = \Sigma m (1, 3, 5, 10, 11, 13, 14) + d(0, 2)$.

Soln. :

The don't care conditions are assumed to be logic 1's. The design table is shown in Fig. P. 5.7.15(a) and the corresponding logic diagram is shown in Fig. P. 5.7.15(b).

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	(0)	(1)	(2)	(3)	4	(5)	6	7
A	8	9	(10)	(11)	12	(13)	(14)	15
Input to Mux	$\bar{A}$	$\bar{A}$	1	1	0	1	A	0

(a) Design table



(b) Implementation

(C-445) **Fig. P. 5.7.15**

5.8 Demultiplexers :

5.8.1 Demultiplexer Principle : S-16

MSBTE Questions

Q. 1 Define and draw the logical symbol of a demultiplexer. **(S-16, 2 Marks)**

Definition :

A de-multiplexer is a combinational circuit with one data input, n outputs and m select inputs, which selects one of the n data outputs and routes (connects) it to the input. The selection of one of the n outputs is done with the help of the select inputs.

Block diagram :

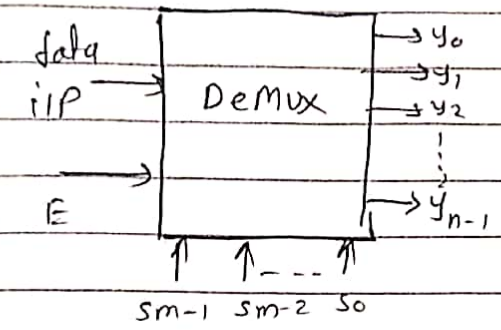
- The block diagram of a 1 to n demultiplexer is shown in Fig. 5.8.1(a) and its equivalent circuit is as shown in Fig. 5.8.1(b). It has one data input, n outputs , m select inputs and one enable input as shown.

* Demultiplexers:-

Definition:- A demultiplexer is a combinational ckt with one data i/p & n output and m select lines / inputs.

which selects one of the n data output and routes (connects) it to the input. The selection of one of the n outputs is done with the help of select i/p.

block diagram :-



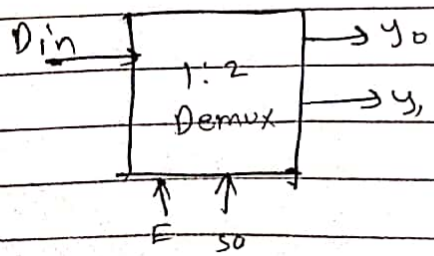
Need of demultiplexers :-

1) In most of electronic systems the digital data from more than one sources is added together and this common signal is transmitted over a commas communication channel.
2) under such circumstances we require a circuit which separate the multiple signals from the common one.
3) This ckt nothing else but a demultiplexer which has one i/p, many output & some select inputs.

* Types of demultiplexers :-

1) 1:2 demultiplexer
2) 1:4 demultiplexer
3) 1:8 demultiplexer
4) 1:16 demultiplexer

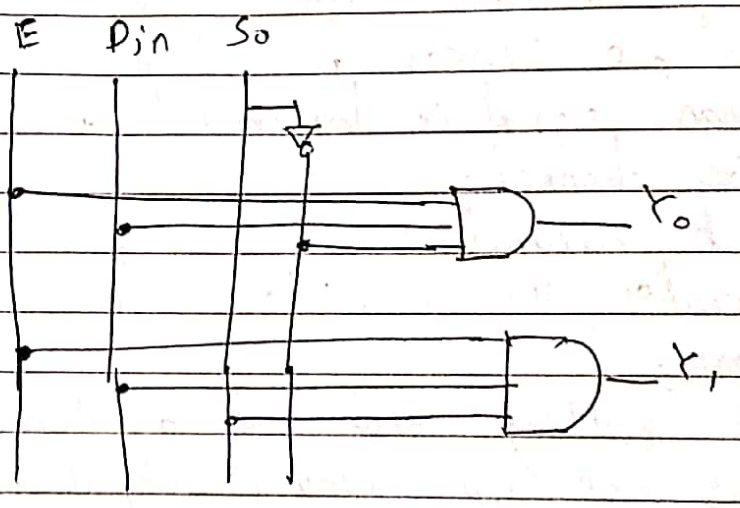
1) 1:2 demux :-



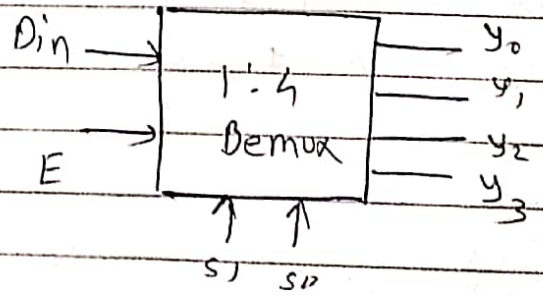
Enable	select	outputs	
E	S_0	Y_1	Y_0
0	X	0	0
1	0	0	D_{in}
1	1	D_{in}	0

$Y_0 = E D_{in} \bar{S_0}$

$Y_1 = E D_{in} S_0$



2) 1:4 Demultiplexer :-

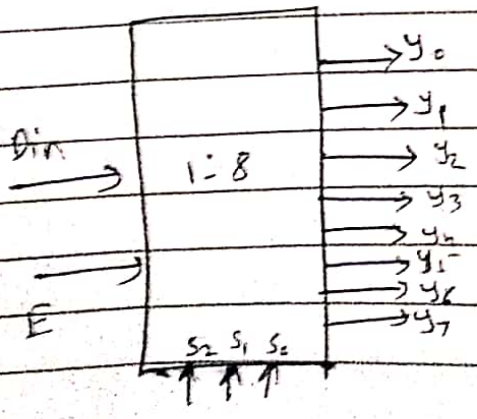


E	D_{in}	S_1	S_0	Y_0	Y_1	Y_2	Y_3	
0	0	x	x	0	0	0	0	
1	0	0	0	0	0	0	0	Y_0 connected to D_{in}
1	1	0	0	1	0	0	0	
1	0	0	1	0	0	0	0	Y_1 connected to D_{in}
1	1	0	1	0	1	0	0	
1	0	1	0	0	0	0	0	Y_2
1	1	1	0	0	0	1	0	
1	0	1	1	0	0	0	0	Y_3
1	1	1	1	0	0	0	1	

OR

E	D_{in}	S_1	S_0	Y_0	Y_1	Y_2	Y_3
0	0	x	x	0	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	0	1	0	0
1	1	1	0	0	0	1	0
1	1	1	1	0	0	0	1

3) ~~8 mux~~ 1:8 Demux

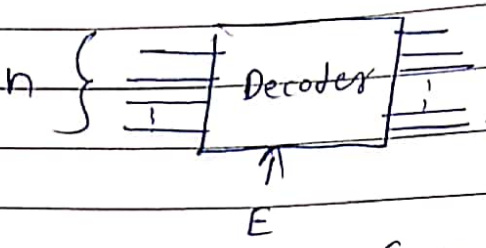


E	S_2	S_1	S_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	Din
1	0	0	1	0	0	0	0	0	0	Din	0
1	0	1	0	0	0	0	0	0	Din	0	0
1	0	1	1	0	0	0	0	Din	0	0	0
1	1	0	0	0	0	0	Din	0	0	0	0
1	1	0	1	0	0	Din	0	0	0	0	0
1	1	1	0	0	Din	0	0	0	0	0	0
1	1	1	1	Din	0	0	0	0	0	0	0

Home work :-
implement or design 1:16 demultiplexer.

* Decoder :-

Decoder is multi-input & multi output logic circuit which decodes n inputs into 2^n possible outputs.

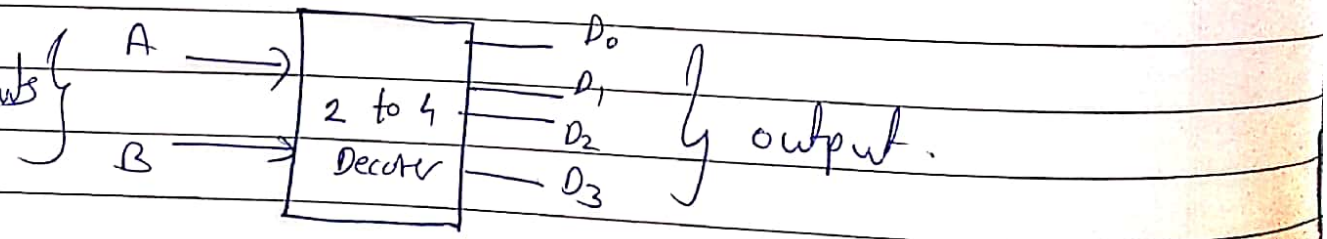


when E (Enable) < E = 0 decoder is disabled
E = 1 decoder is enabled.

* Application :-

1) code converters.
2) BCD to seven segment decoders
3) Nixie tube decoders
4) Relay actuators.

1) 2 to 4 decoder.

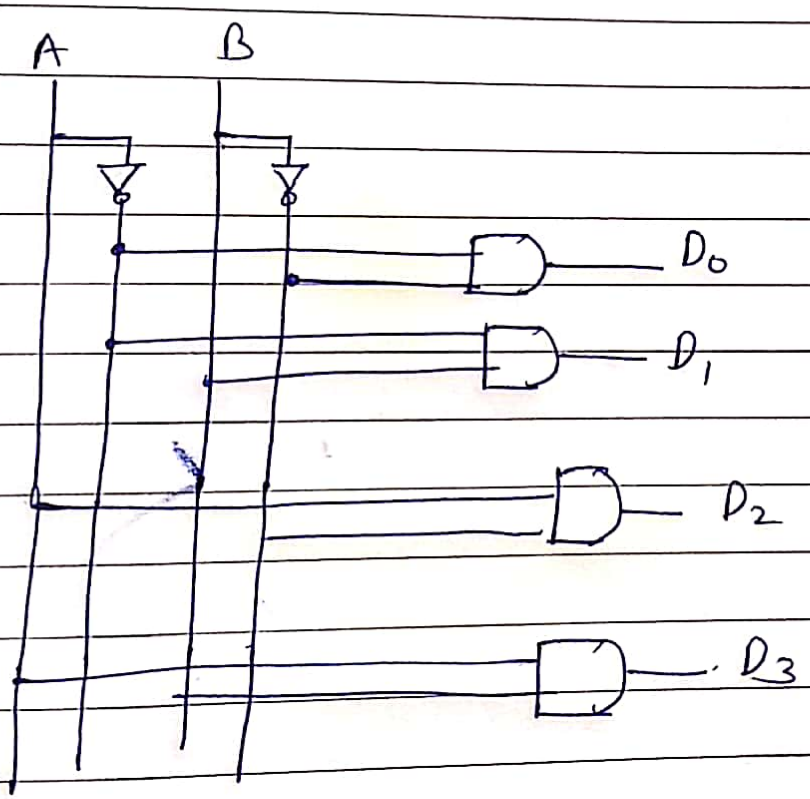


* Block diagram *

E	A	B	D_0	D_1	D_2	D_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

$D_0 = \bar{A}\bar{B}$ $D_1 = \bar{A}B$ $D_2 = A\bar{B}$

$D_3 = AB$.



5.14.5 3 to 8 Line Decoder : S-06

MSBTE Questions

Q. 1 Draw the circuit diagram of 3 : 8 decoder. **(S-06, 4 Marks)**

- The truth table of 3 to 8 line decoder is shown in Table 5.14.3.

Table 5.14.3 : Truth table for 3 to 8 line decoder

Inputs			Outputs								
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	$D_0 = \bar{A}\bar{B}\bar{C}$
0	0	1	0	**1**	0	0	0	0	0	0	$D_1 = \bar{A}\bar{B}C$
0	1	0	0	0	**1**	0	0	0	0	0	$D_2 = \bar{A}B\bar{C}$
0	1	1	0	0	0	**1**	0	0	0	0	$D_3 = \bar{A}BC$
1	0	0	0	0	0	0	**1**	0	0	0	$D_4 = A\bar{B}\bar{C}$
1	0	1	0	0	0	0	0	**1**	0	0	$D_5 = A\bar{B}C$
1	1	0	0	0	0	0	0	0	**1**	0	$D_6 = AB\bar{C}$
1	1	1	0	0	0	0	0	0	0	**1**	$D_7 = ABC$

Logic diagram :

- The logic diagram for 3 to 8 line decoder is shown in Fig. 5.14.5.

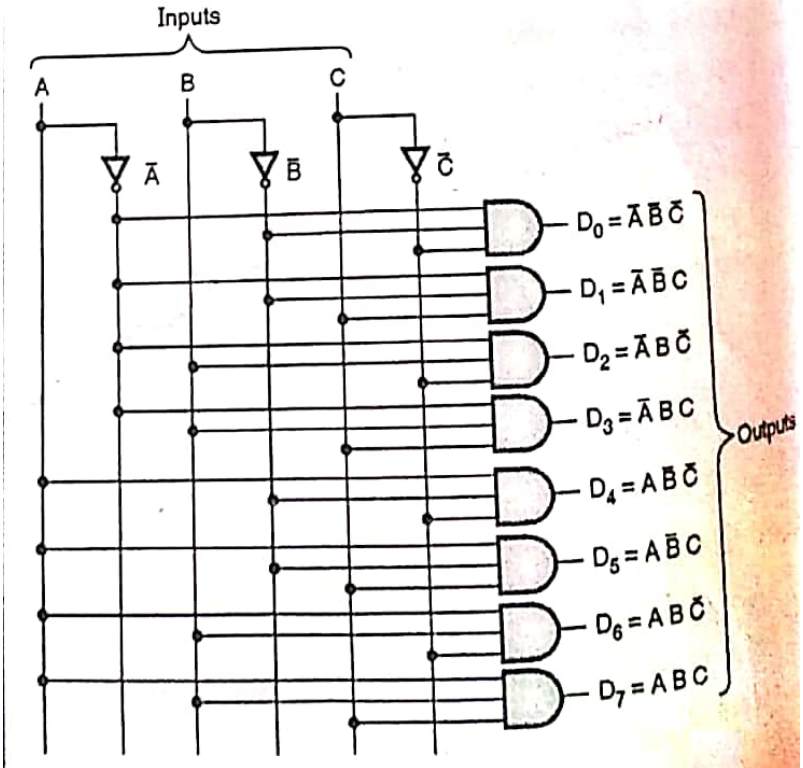
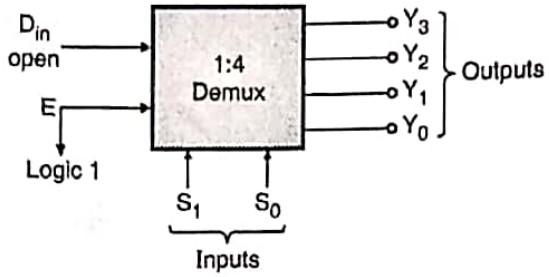


Fig. A (C-482) Fig. B

5.14.3 Demultiplexer as Decoder :

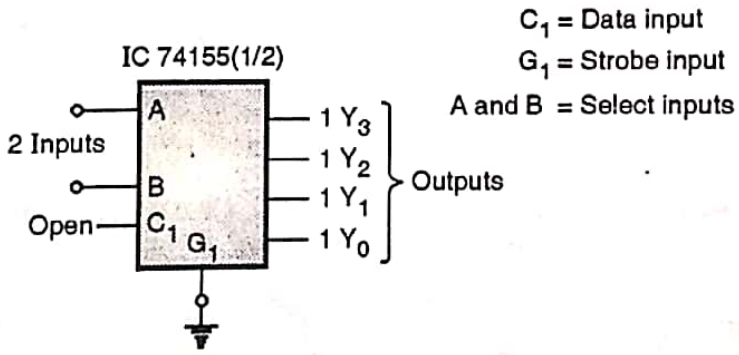
- We can use a demultiplexer as a decoder.
- Let us see how to operate a 1 : 4 demux as 2 : 4 decoder.
- Consider Fig. 5.14.3 which shows a 1 : 4 demultiplexer.
- D_{in} is the data input, S_1 S_0 are the select lines and Y_3 through Y_0 are the outputs.
- In order to operate it as 2 : 4 decoder, we have to use S_1 S_0 as inputs, keep D_{in} open and use Y_3 to Y_0 as outputs as shown in Fig. 5.14.3.



(C-483) **Fig. 5.14.3 : 1 : 4 demux as 2 : 4 decoder**

5.14.4 IC74155 as 2 : 4 Decoder :

- Fig. 5.14.4 shows the connection diagram to use IC 74155 as 2 : 4 decoder.
- The select inputs A and B are used as the 2 inputs to the decoder. The data input C_1 is left open (logic 1) and the strobe input G_1 is connected to ground, so as to make it active.
- The operation of this circuit is summarized in Table 5.14.2.



(C-7056) **Fig. 5.14.4 : Use of IC 74155 as 2 : 4 decoder**

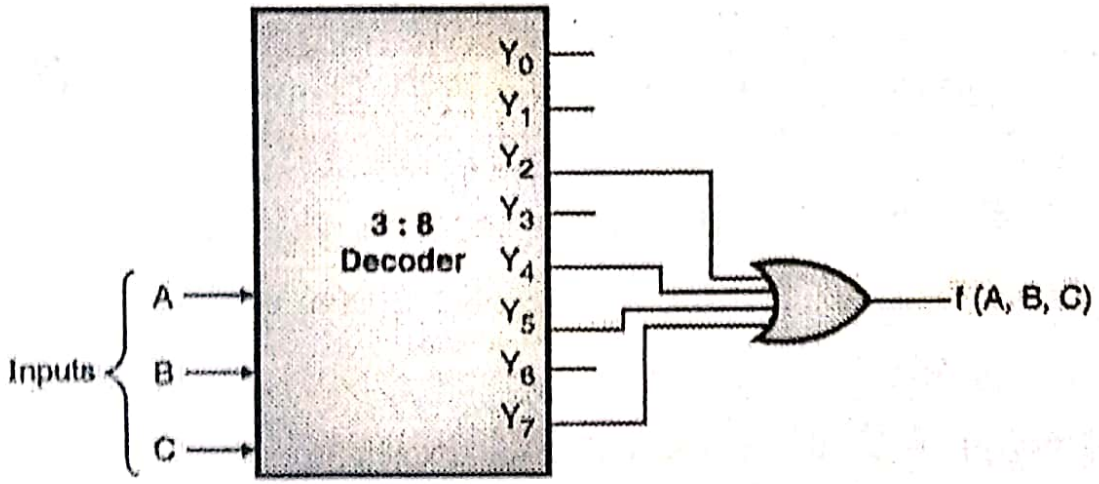
(C-7057) **Table 5.14.2 : Truth table of IC 74155 working as 2 : 4 decoder**

Inputs				Outputs			
A	B	C_1	G_1	$1Y_0$	$1Y_1$	$1Y_2$	$1Y_3$
0	0	1	0	0	1	1	1
0	1	1	0	1	0	1	1
1	0	1	0	1	1	0	1
1	1	1	0	1	1	1	0

Ex. 5.14.1 : Implement the following Boolean function using a 3 : 8 decoder and external gates : $f(A, B, C) = \Sigma (2, 4, 5, 7)$

Soln. :

- The decoder produces minterms. The outputs Y_2 , Y_4 , Y_5 and Y_7 are ORed to produce the required output.
- The logic implementation of the given logic function is shown in Fig. P. 5.14.1.

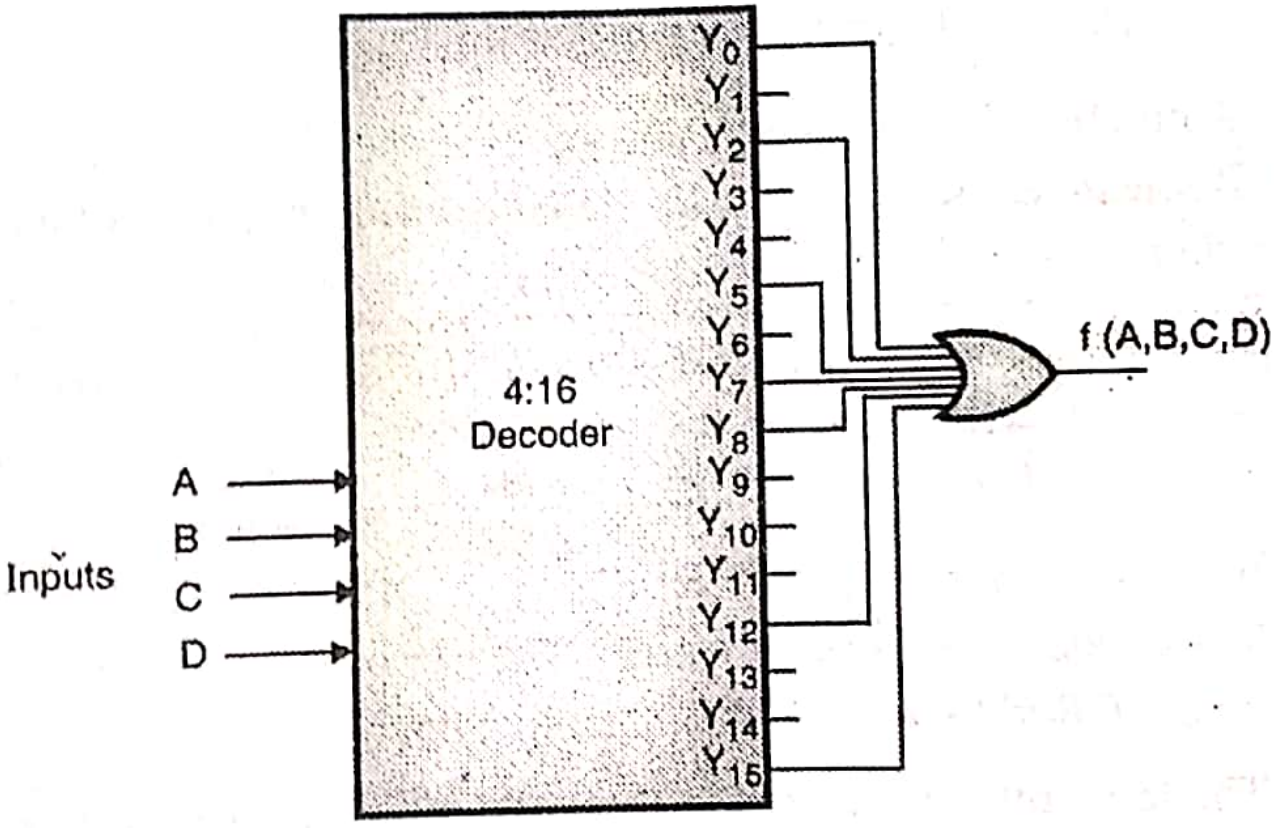


(C-486) **Fig. P. 5.14.1 : Implementation of Boolean equation using decoder and gate**

Ex. 5.14.2 : Realize using demultiplexer / decoder :
$f = \Sigma (0, 2, 5, 7, 8, 12, 15)$. **W-08, 3 Marks**

Soln. :

- Fig. P. 5.14.2 shows the implementation of 4 : 16 decoder.
- The decoder produces minterms. The outputs Y_0, Y_2, Y_5, Y_7, Y_8, Y_{12} and Y_{15} are ORed to produce the required output.



(C-2632) **Fig. P. 5.14.2**