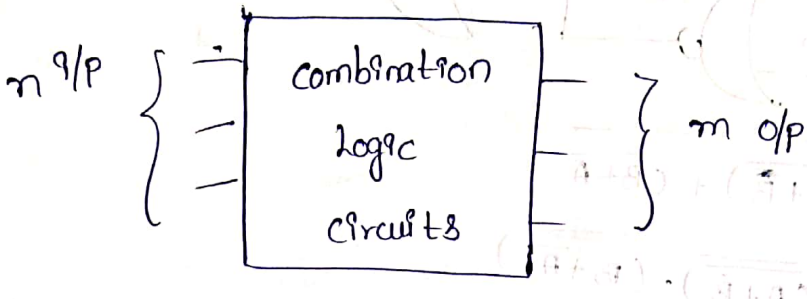
UNIT-4

Combinational Logic (or) Circuits

Block diagram of combinational circuits :-



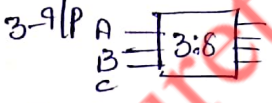
Topics :-

1. Half-Adder — 2-I/P — 2 O/P
2. Half-Sub — 2-I/P — 2 O/P
3. Full-adder — 3-I/P — 2 O/P
4. Full-Sub — 3-I/P — 2 O/P
5. Decoder — n-I/P — 2^n O/P

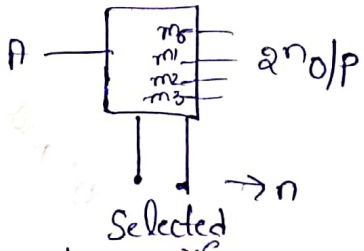
Ex :- 2 -I/P - 2^2 = 4 O/P

Ex :- 3 -I/P - 2^3 = 8 O/P

3:8 Decoder



6. Encode — 2^n I/P - n O/P



7. multiplex — MUX - n selection lines — $n : 2^n$ - O/P

8. De-multiplex — De-MUX — $n : 2^n$ - I/P

9. priority Encode

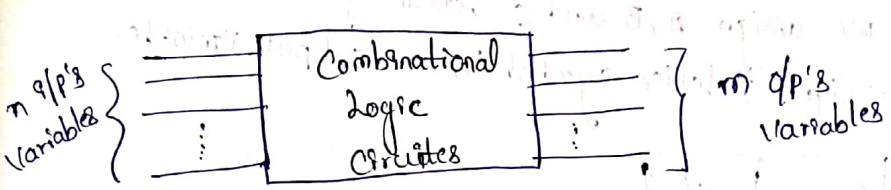
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Combinational Circuits

→ When logic gates are connected together to produce a specified output for certain specified combinations of input variables, with no storage involved. The resulting circuit is called as combinational logic circuit.

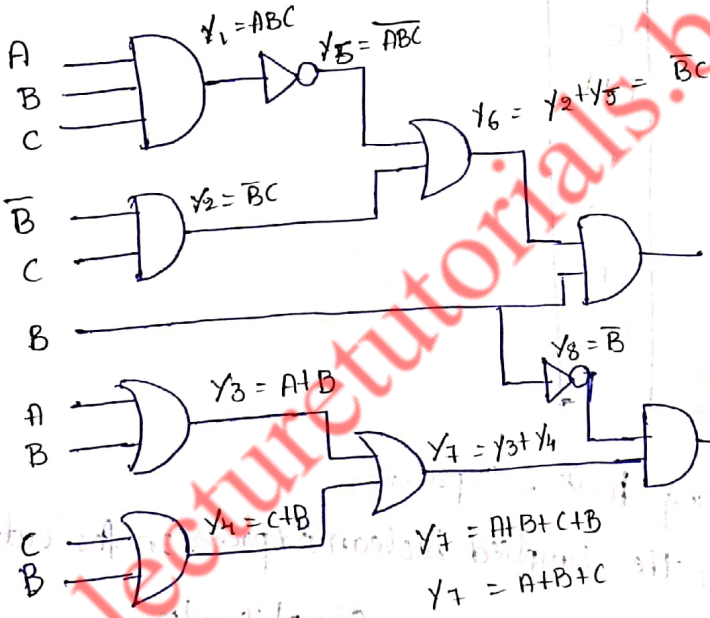
In combinational logic the output variables are at all times dependce on the combination of input variables.

→ The Block diagram of Input Vs combination circuit is given below



Analysis Processur :-

:- To obtain the boolean function for output of the given circuit



$F_1 = Y_6 \cdot B = (\overline{B}C + \overline{ABC})B$

$= B\overline{B}C + B(\overline{ABC})$

$= 0 + BC(\overline{A} + \overline{B} + \overline{C})$

$= \overline{A}BC + B\overline{B}C$

$F_2 = Y_8 \cdot Y_7$

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

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

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

$Y_7 = A + B + C + B$

$Y_7 = A + B + C$

$\therefore F_1 = B(\overline{A} + \overline{C})$, $F_2 = \overline{B}(A + C)$

A B C	$\overline{B}$	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	F_1	F_2
0 0 0	1	0	0	0	0	1	1	0	1	0	0
0 0 1	1	0	1	0	1	1	1	1	1	0	1
0 1 0	0	0	0	1	1	1	1	1	0	1	0
0 1 1	0	0	0	1	1	1	1	1	0	1	0
1 0 0	1	0	0	0	1	0	1	1	1	0	1
1 0 1	1	0	1	0	1	1	1	1	1	0	1
1 1 0	0	0	0	1	1	1	1	1	0	0	0
1 1 1	0	1	0	1	1	0	0	1	0	0	0

Design procedure :-

Example:

* Design a combinational logic circuit with three input variables that will produce a logic one output when more than one input variables are logic one.

Step 1 :- Derive the truth table for given statement

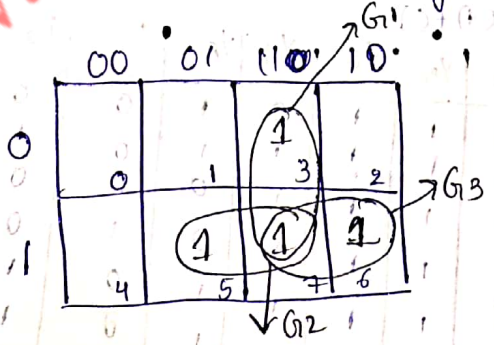
The given problem specifies that there are three input variables and only one output variable. we assign A, B and C letter symbols to three input variables and assign "Y" letter symbol to one output variable.

Ste

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

Step 2 :- obtain simplifying boolean expression

Now we can obtain the simplied boolean expression for output for out variable Y for using K-map simplification



G1

A	B	C
0	1	1
1	1	1

BC

G2

A	B	C
1	0	1
1	1	1

AC

G3

A	B	C
1	1	1
1	1	0

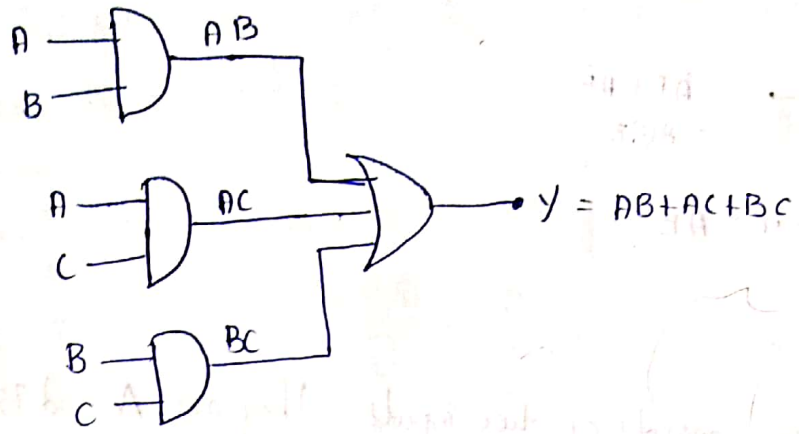
AB

$Y = BC + AC + AB$

Step :- 3 Draw the logic diagram

we are going various combinational circuits to the above boolean expression

Logic Diagram



Design of Adders

→ Digital computers perform various Arthamatic operation as the addition of two binary digits. This simple addition consists of four elemer tary operations, namely

$0+0=0$
$0+1=1$
$1+0=1$
$1+1=(10)_2$

) Half Adder

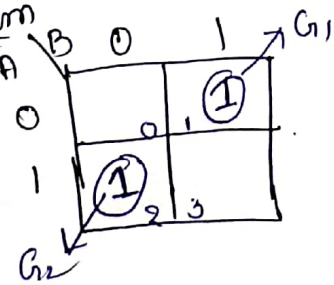
→ The Half Adder consists of two input variables and two output variables we assign A and B are input variables and Sum (S) and carry (C).

A, B = i/p variables
S, C = o/p variables

Truth table

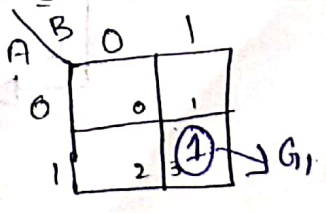
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

k-map for Sum



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

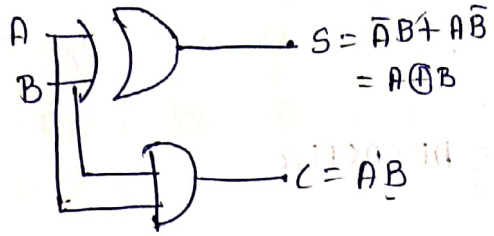
k-map for carry



$C = AB$

logic Diagram

for S



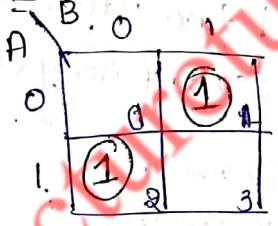
(ii) Half - subtraction :-

→ The Half subtraction consists of two inputs they are. A and B and two outputs they are difference (D) and barrow (B).

→ The truth table

A	B	D (Difference)	B (barrow)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

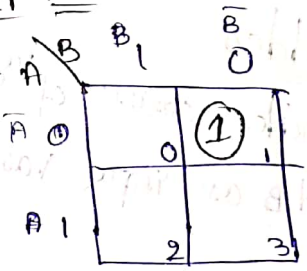
K-map for D



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

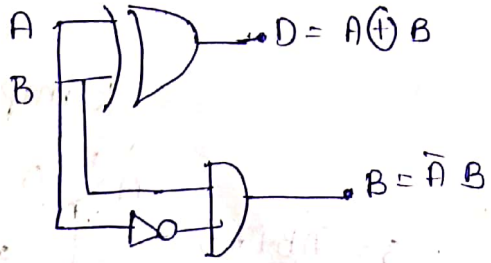
$= A \oplus B$

K-map for B



$B = \bar{A}B$

Logic Diagram



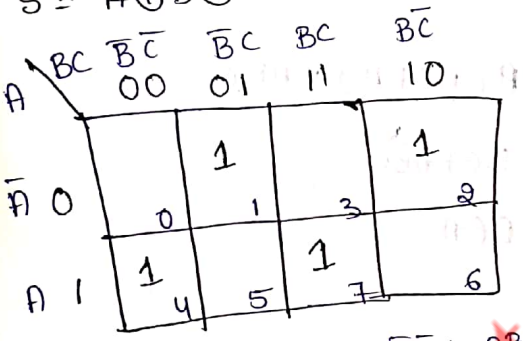
Full-Adder :-

→ In full-Adder consists of three inputs they are A, B, C ~~and D~~ and outputs are Sum and Carry

A	B	C	S (Sum)	C (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 S

$S = A \oplus B \oplus C$



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

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

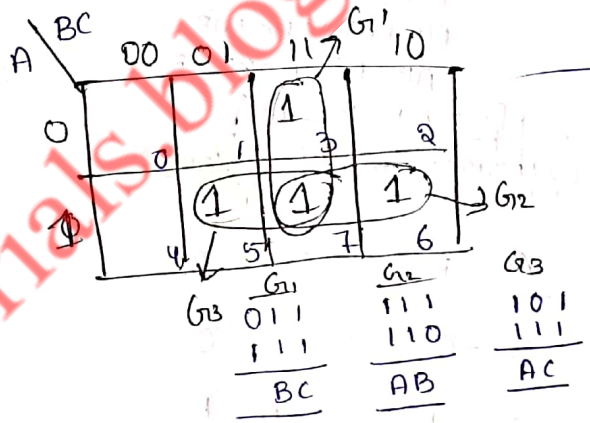
$S = C(A \odot B) + \bar{C}(A \oplus B)$

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

$S = A \oplus B \oplus C$

K-map for C

$C = AB + BC + CA$



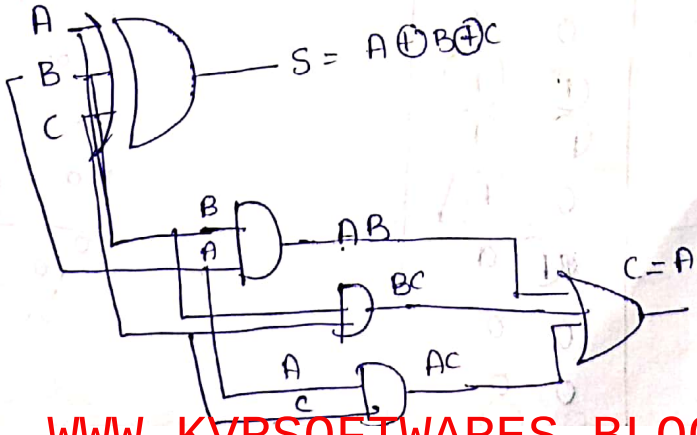
$C = AB + BC + CA$

(or)

$= \cancel{\bar{A}BC + A\bar{B}C + ABC + AB\bar{C}}$

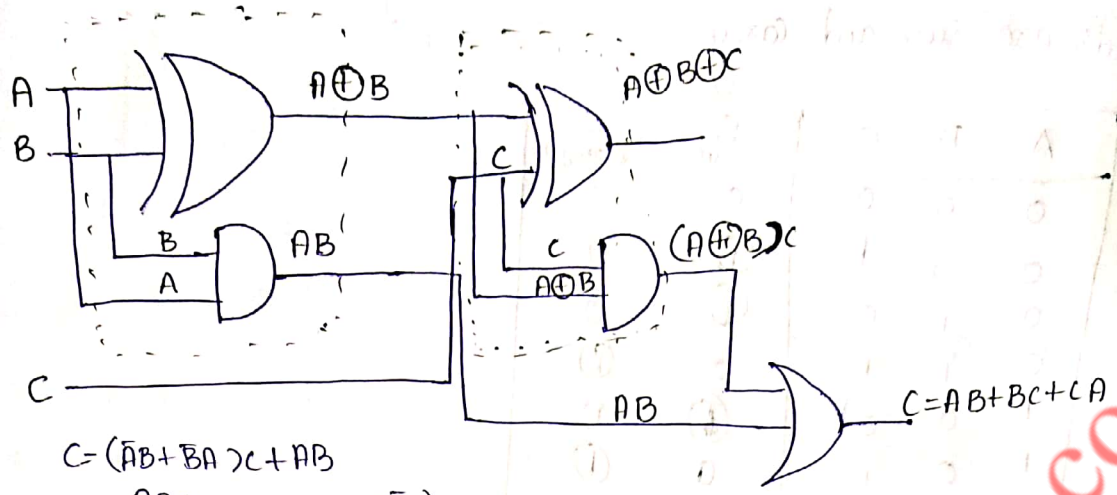
$= \cancel{C(\bar{A}B + A\bar{B}) + \ldots}$

Logic Diagram :-



(9m)

** To Design full Adder by using two to half adders and OR gate :-



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

$= AB(1+C) + C(AB + A\bar{B})$

$= AB + ABC + \bar{A}BC + A\bar{B}C$

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

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

$= AB(1+C) + BC + A\bar{B}C$

$= AB + ABC + BC + A\bar{B}C$

$= AB + BC + AC(B+\bar{B})$

$C = AB + BC + AC$

~~$C = (A \oplus B)C + AB$~~

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

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

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

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

$= C + A$

Subtraction operations :-

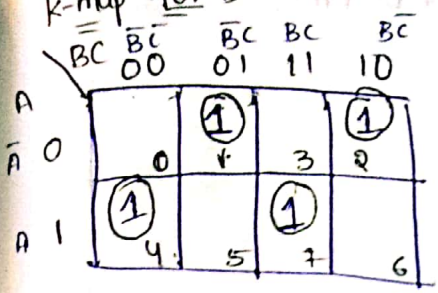
A	B	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Full - Subtraction

→ It consists of three inputs two outputs they are A, B, C as the inputs D, B as the out puts

A	B	C	D	B
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 D



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

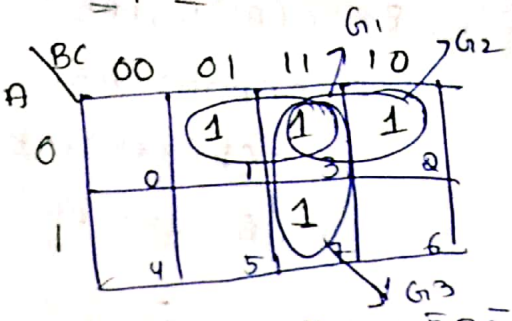
$= ((\bar{A}\bar{B} + AB) + \bar{C}(\bar{A}B + A\bar{B})$

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

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

$= A \oplus B \oplus C$

k-map for B



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

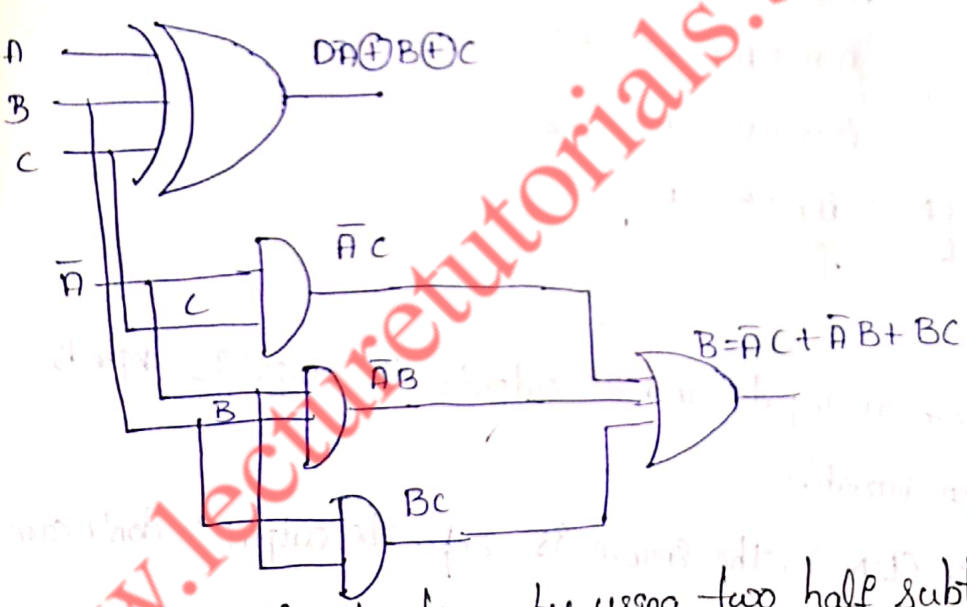
$= \underset{A\ BC}{\underline{G_1}}\ \underset{0\ 1\ 1}{\underline{G_2}}\ \underset{0\ 1\ 1}{\underline{G_3}}$

$0\ 0\ 1 \quad 0\ 1\ 0 \quad 1\ 1\ 1$

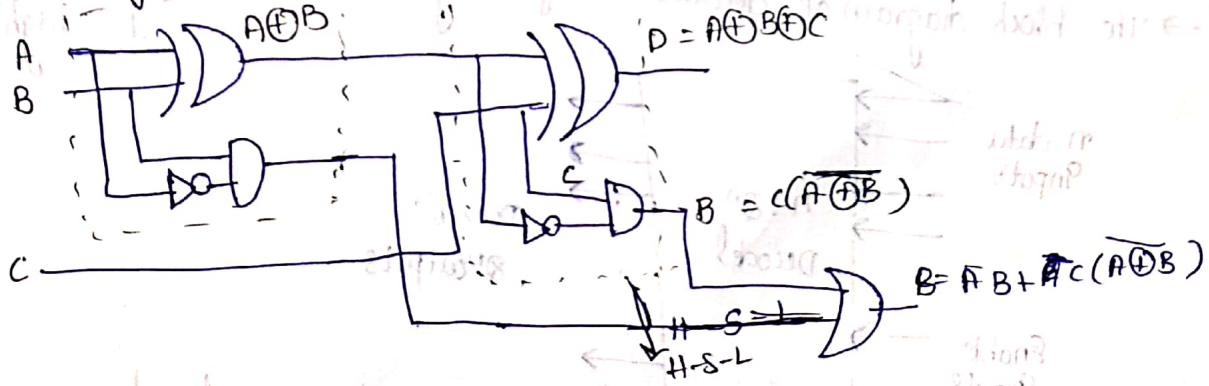
$\bar{A}C \qquad \bar{A}B \qquad BC$

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

Logic diagram



To Design full subtraction by using two half subtraction and one OR gate



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

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

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

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

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

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

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

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

$= \bar{A}B + C\bar{A}\bar{B} + ABC$ $[\because 1+C=1]$

$= \bar{A}BC(1+C) + \bar{A}\bar{B}C + ABC$

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

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

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

$= \bar{A}BC(1+C) + BC + \bar{A}\bar{B}C$

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

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

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

Decoder

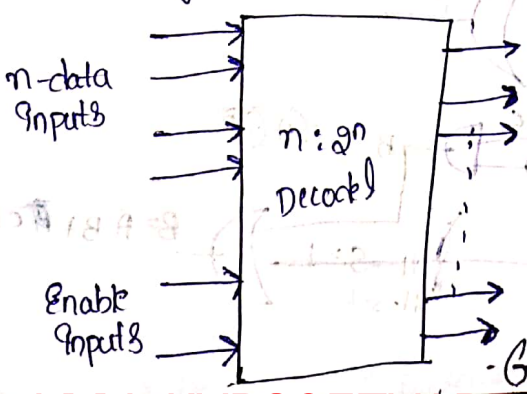
→ The Decoder n inputs and 2^n outputs the output is depends upon Enable input variable.

→ The Enable is (EN), the Enable is zero the output is don't care condition

→ the Enable is '1' the output is on state

→ The block diagram of decoder is given by

[0 - Low
1 - high]



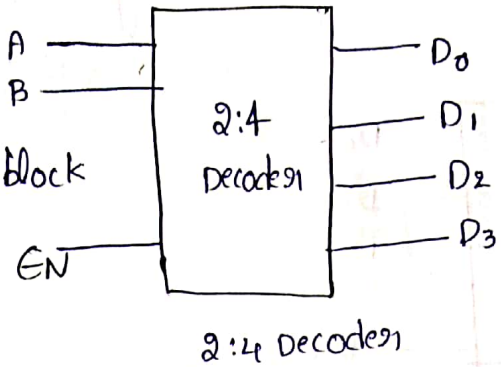
General structure of decoder

Binary Decoder :-

2 : 4 Decoder :-

Truth table :-

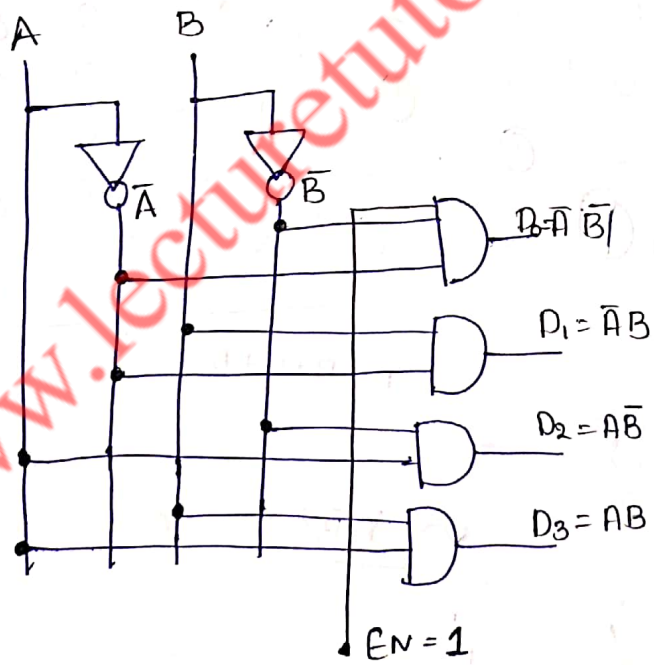
The 2 : 4 Decoder diagram is given by



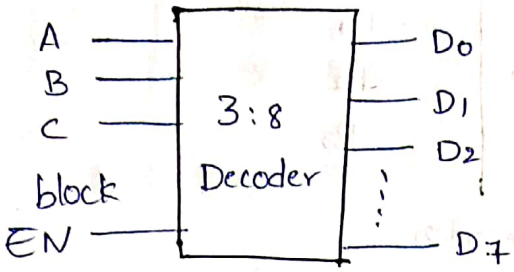
2 : 4 Decoder

Truth table :-

Inputs			outputs			
EN	A	B	D_3	D_2	D_1	D_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

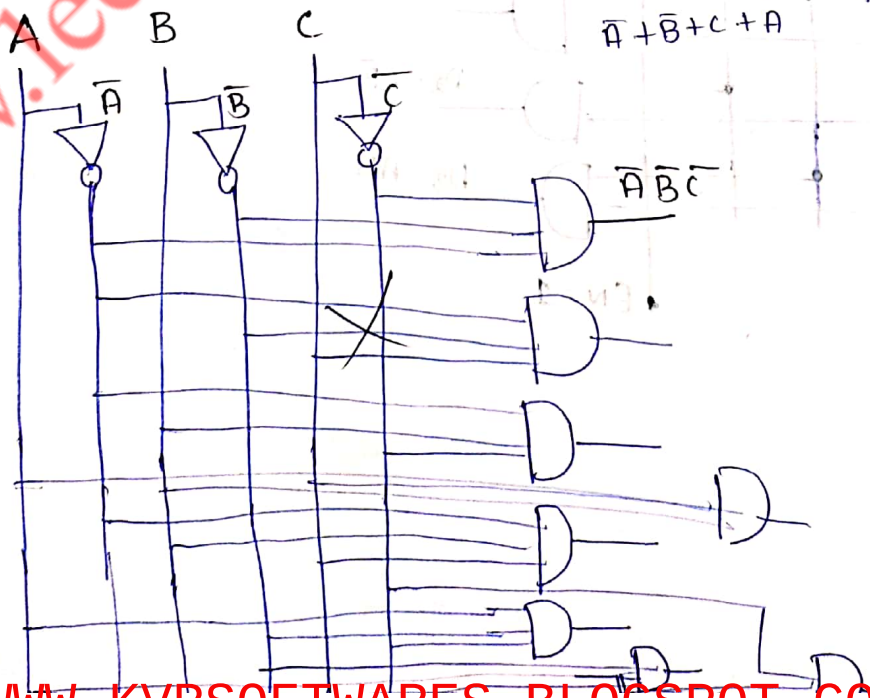


3:8 Decoder :- The 3:8 Decoder is given by

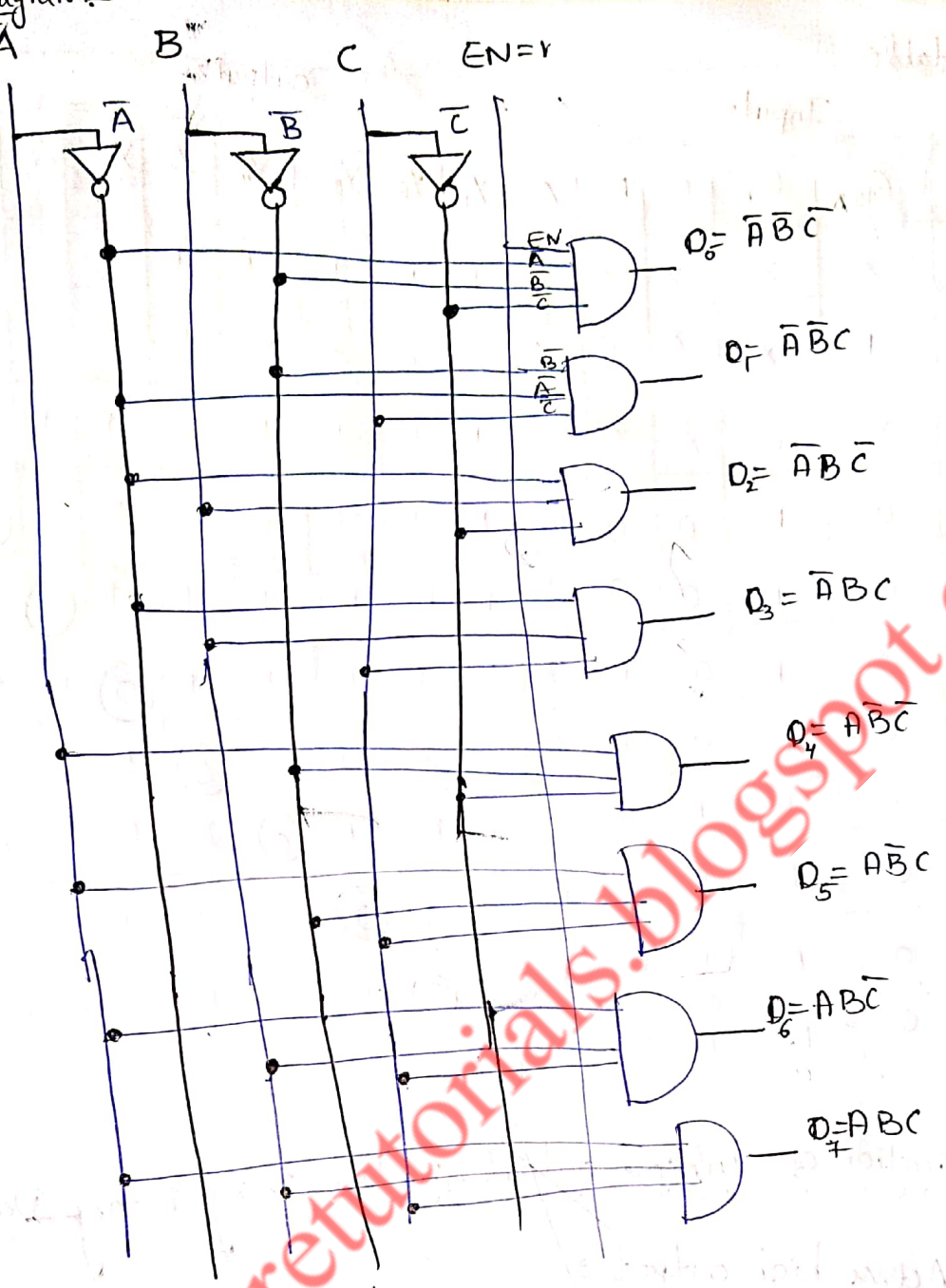


Truth table

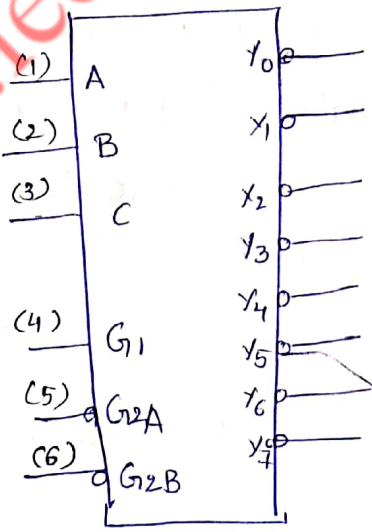
Input				output							
EN	A	B	C	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0



Logic diagram:



The 74 x 138 3 to 8 Decoder



Logic Symbol

Truth table

Inputs						outputs							
G_{2B}	G_{2A}	G_1	C	B	A	$\overline{Y_7}$	$\overline{Y_6}$	$\overline{Y_5}$	$\overline{Y_4}$	$\overline{Y_3}$	$\overline{Y_2}$	$\overline{Y_1}$	Y_0
1	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	0	x	x	x	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	0
0	0	1	0	0	1	1	1	1	1	1	1	0	1
0	0	1	0	1	0	1	1	1	1	1	0	1	1
0	0	1	0	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	0	1	1	1	0	1	1	1	1
0	0	1	1	0	1	1	1	0	1	1	1	1	1
0	0	1	1	1	0	1	0	1	1	1	1	1	1
0	0	1	1	1	1	0	1	1	1	1	1	1	1

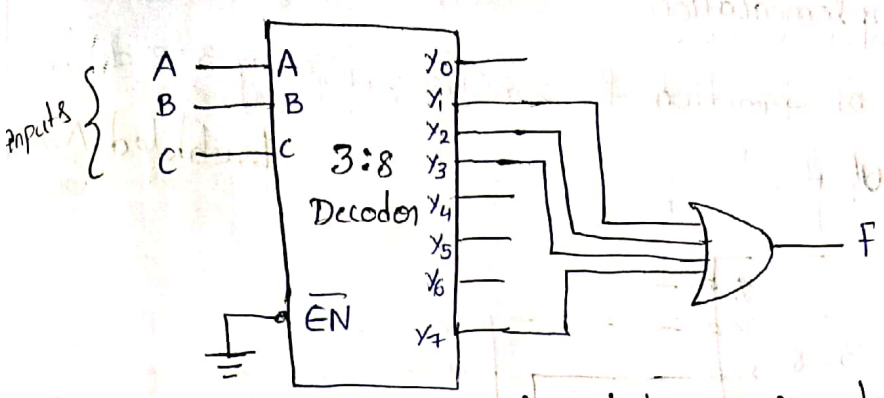
Realization of Multiple output function using Binary Decoders

For Active high output :-

Sop function implementation

When decoder output is active high, it generates minterms (product terms) for input variables i.e it makes selected output logic 1. In such case to implement Sop function we have to take sum of selected product terms genrated by decoder. This can be implemented by ORing the selected decoder outputs, as shown in the fig 5.62 The fig 5.62 shows the implementation of function $f = \Sigma M(1,2,3,7)$ using 3:8 decoders with active high outputs

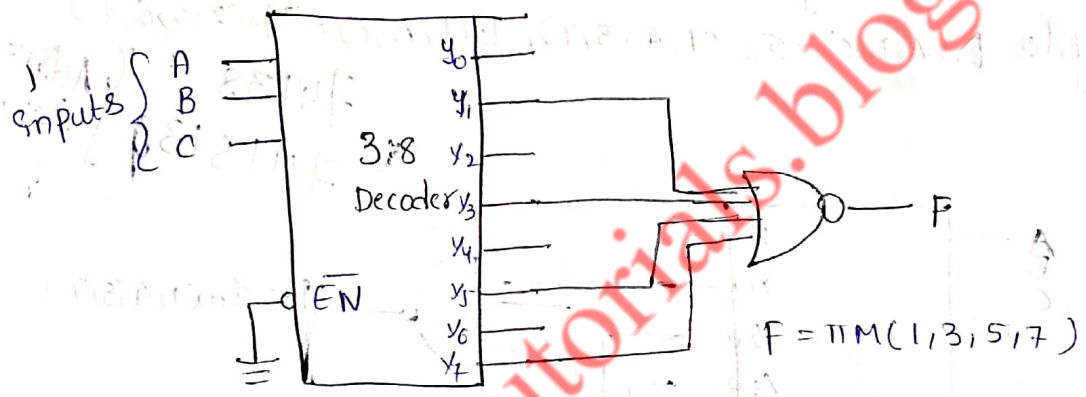
0 – low – off state
1 – high – on



Single output function implementation using decoder and gate

POS Function Implementation

The implementation of a function $f = \Pi M(1,3,5,7)$ using 3:8 decoder with active high outputs.



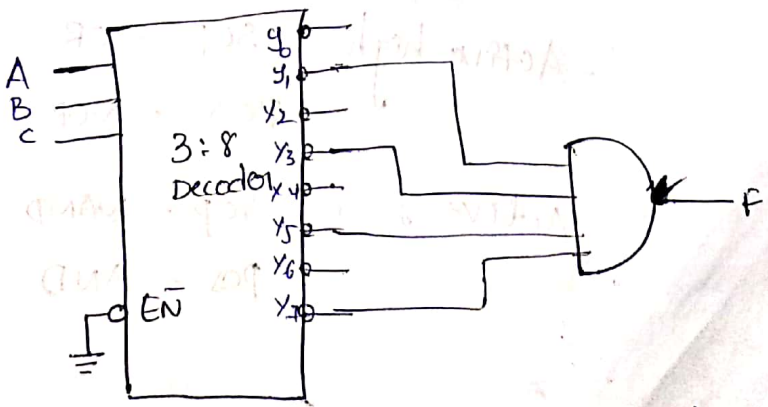
Implementation of POS function using decoder

For Active low Output

POS function implementation

The implementation of function $f = \Pi M(1,3,5,7)$ using 3:8 decoder with active low outputs

Bubbled AND

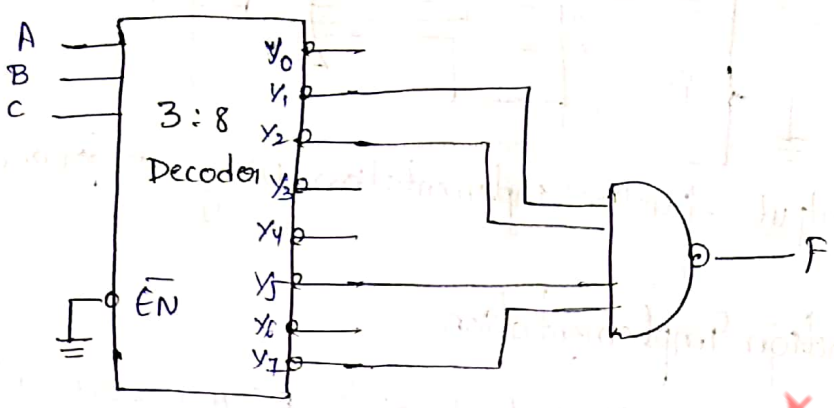


Implementation of POS function using decoder

Sop function Implementation

The implementation of function $f = \Sigma m(1,2,5,7)$ using 3:8 decoder with active low outputs

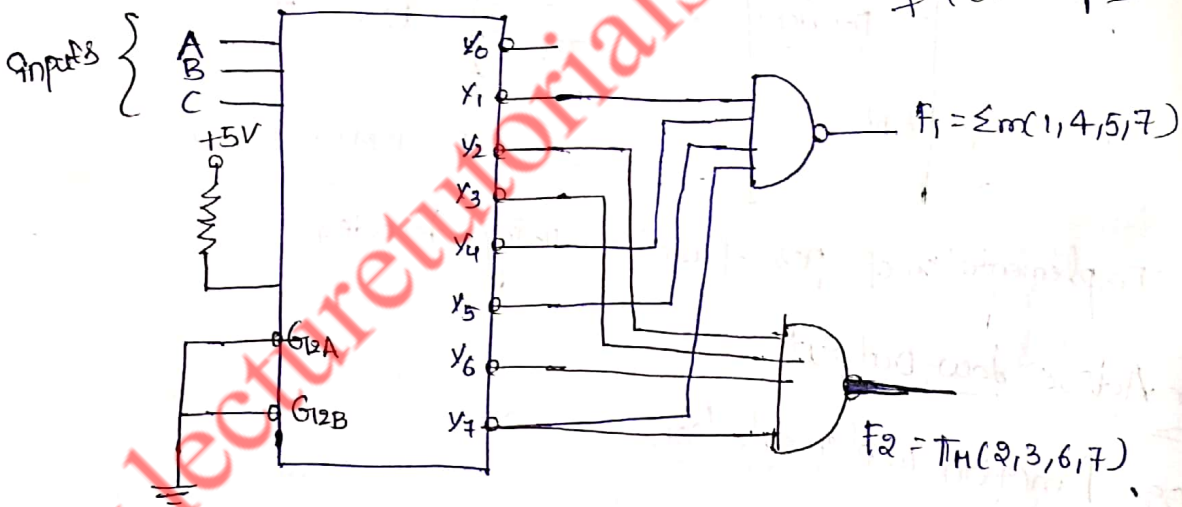
Bubbled NAND Gate



Implementation of sop function using decoder

Ex:- Implement following multiple output function using 74LS138 and External gates. $F_1(A,B,C) = \Sigma m(1,4,5,7)$, $F_2(A,B,C) = \Pi M(2,3,6,7)$

74138
74LS138
} Active low



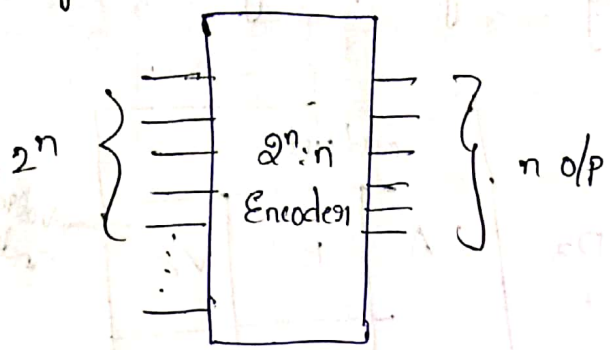
Active high — SOP — OR
POS — NOR

Active low — SOP — NAND
POS — AND

Encoder :- (2^n i/p – n o/p)

→ The Encoder it consists of 2^n inputs and n outputs

Block diagram :-



Ex :- 4

Octal – Binary Encoder

truth-table .

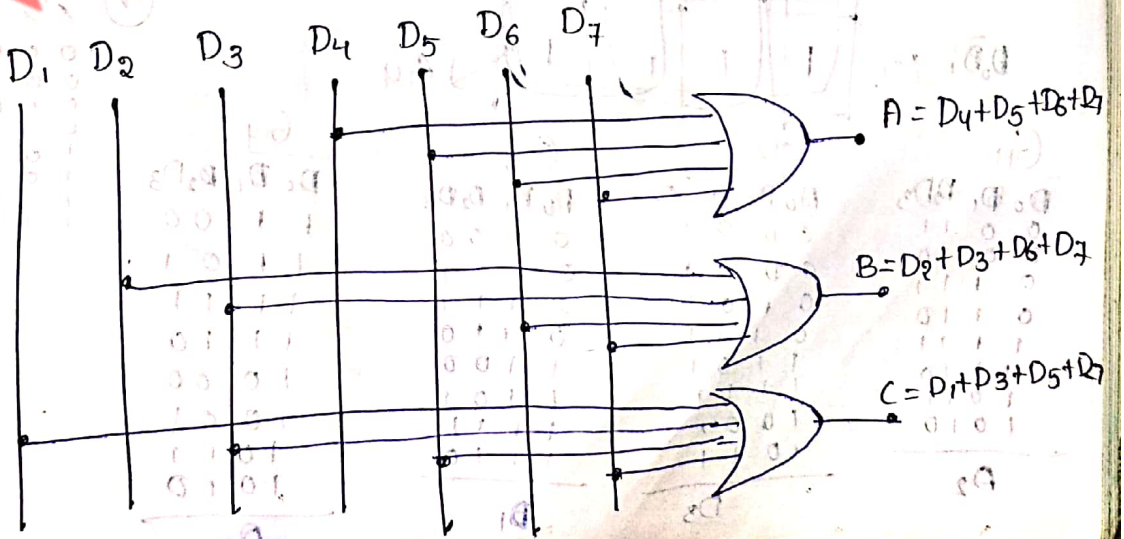
→ The truth table of octal to binary convertor

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

$A = D_4 + D_5 + D_6 + D_7$

$B = D_2 + D_3 + D_6 + D_7$

$C = D_1 + D_3 + D_5 + D_7$



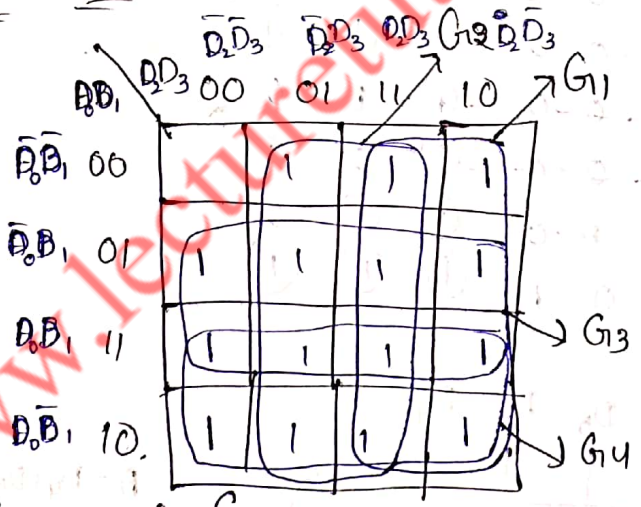
Priority Encoder :-

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 the same time, the input having the highest priority will take precedence

Truth table :-

Inputs				outputs		
D_0	D_1	D_2	D_3	A	B	V (valid indicator)
0	0	0	0	x	x	0
1	0	0	0	0	0	1
x	1	0	0	0	1	1
x	x	1	0	1	0	1
x	x	x	1	1	1	1

four variable k-map - "V"



(i) 1 0 0 0
(ii) x 1 0 0
0 1 0 0
1 0 0 0
(iii) x x 1 0
0 0 1 0
0 1 1 0
1 0 1 0
1 1 1 0
(iv) x x x 1
0 0 0 1
0 0 1 1
0 1 0 1
0 1 1 1
1 0 0 1
1 0 1 1
1 1 0 1
1 1 1 1

G_1

D_0	D_1	D_2D_3
0	0	11
0	0	10
0	1	11
0	1	10
1	1	11
1	1	10
1	0	11
1	0	10

D_2

G_2

D_0	D_1	D_2	D_3
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	1	0	1
1	1	1	1
1	0	0	1
1	0	1	1

D_3

G_3

D_0	D_1	D_2D_3
0	1	00
0	1	01
0	1	11
0	1	10
1	1	00
1	1	01
1	1	11
1	1	10

D_1

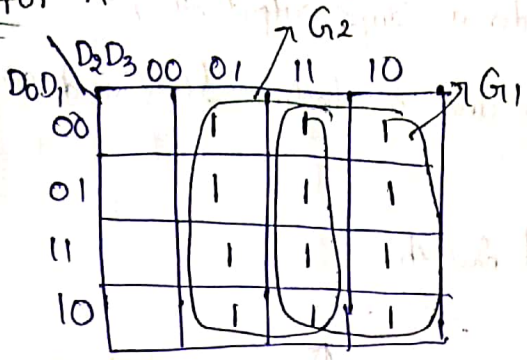
G_4

D_0	D_1	D_2D_3
1	1	00
1	1	01
1	1	11
1	1	10
1	0	00
1	0	01
1	0	11
1	0	10

D_0

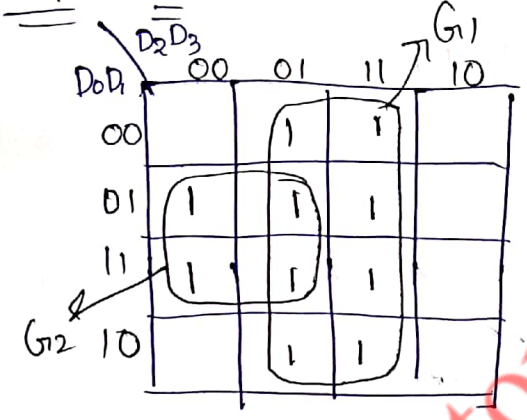
$V = D_0 + D_1 + D_2 + D_3$

k-map for "A"

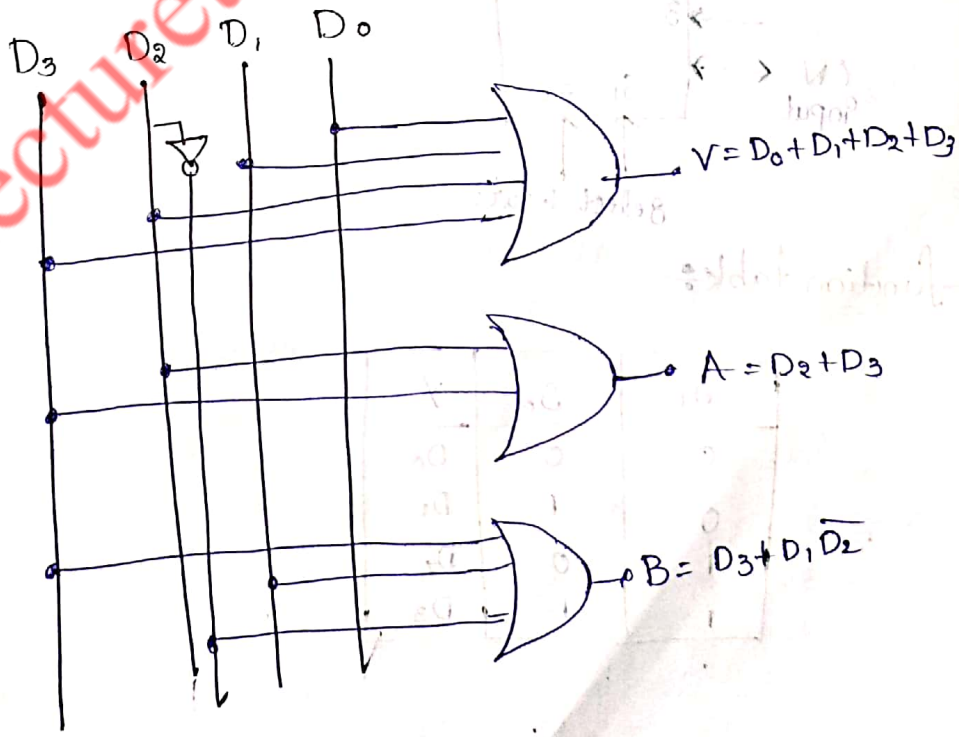


$A = D_2 + D_3$

k-map for "B"



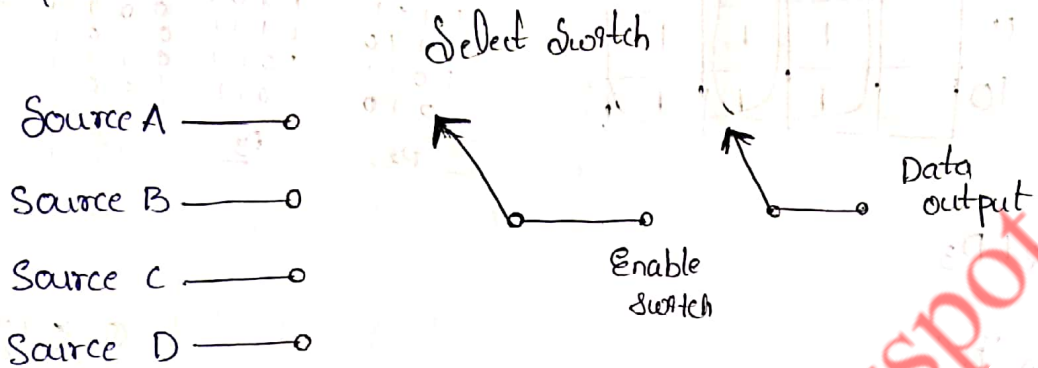
$B = D_3 + D_1\overline{D_2}$



Multiplexers (2^n inputs – n selection lines) – 1 output)

Multiplexer is a digital switch. It allows digital information from several sources to be routed onto a single output line.

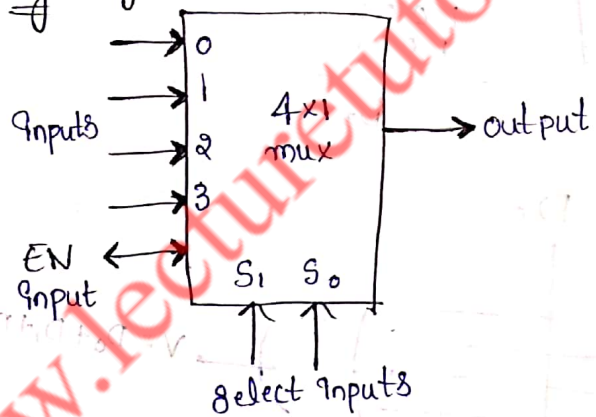
It consists of 2^n input lines and n selection lines and only one output.



Analog selector switch

4 to 1 line multiplexer:-

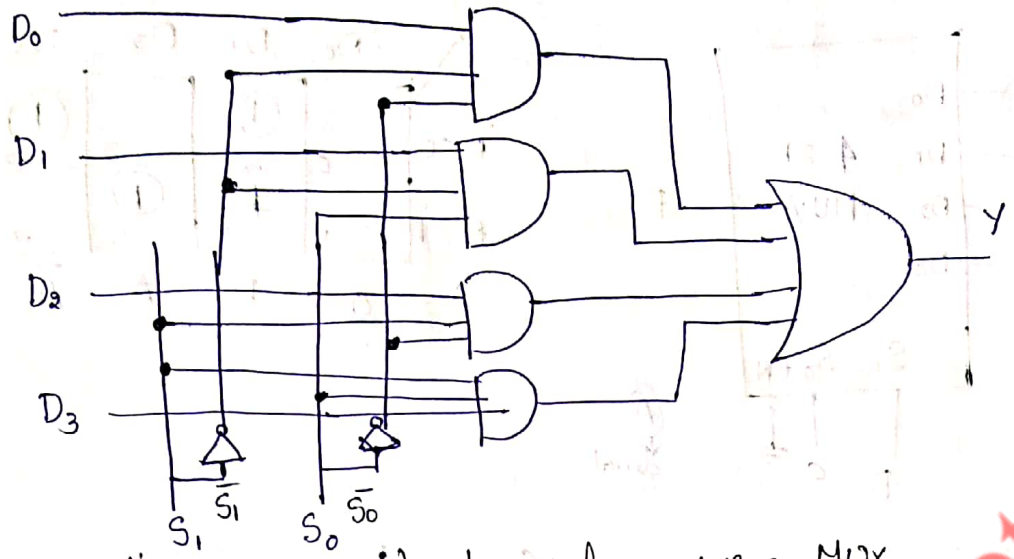
Logic symbol:-



function table:-

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

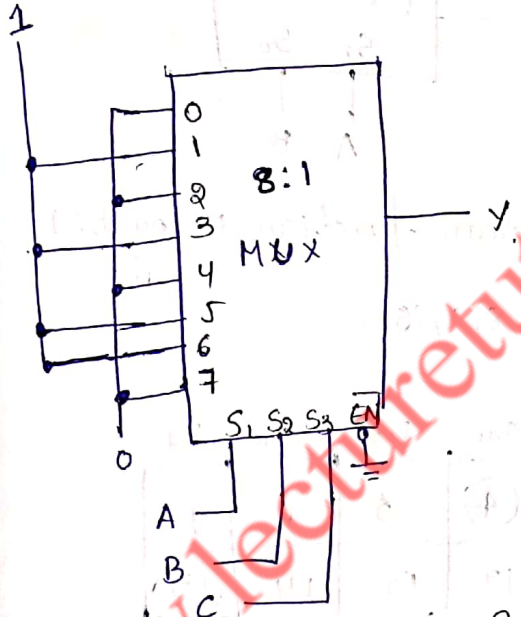
Logic diagram:



Implementation of combinational logic using MUX

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

$F(A,B,C) = \Sigma m(1,3,5,6)$



Boolean function Implemention using MUX

Implemention the following Boolean function 4:1 multiplexer

$F(A,B,C) = \Sigma m(1,3,5,6)$

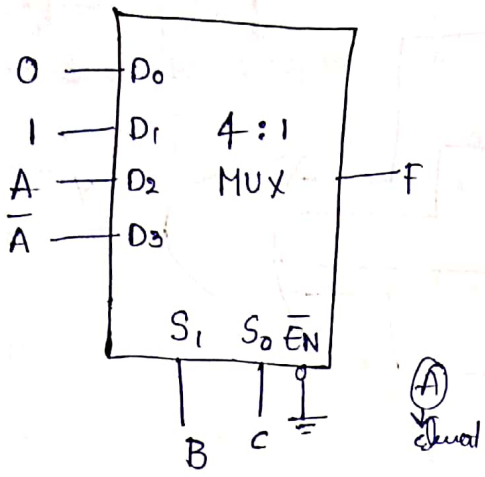
truth table

minterm	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

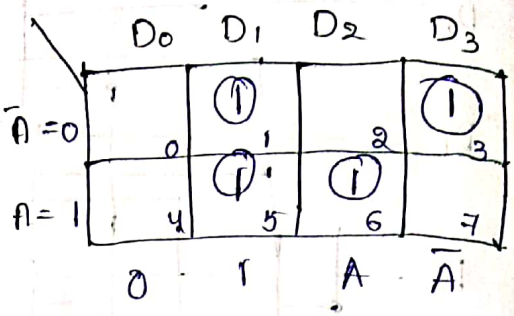
multiplexer Implementation

Implementation table

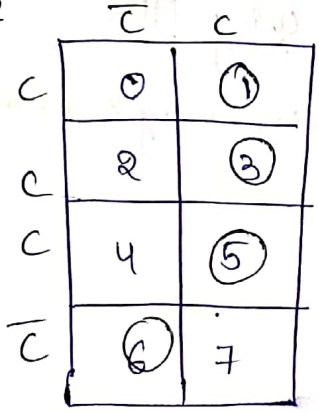
$F = \Sigma m(1, 3, 5, 6)$

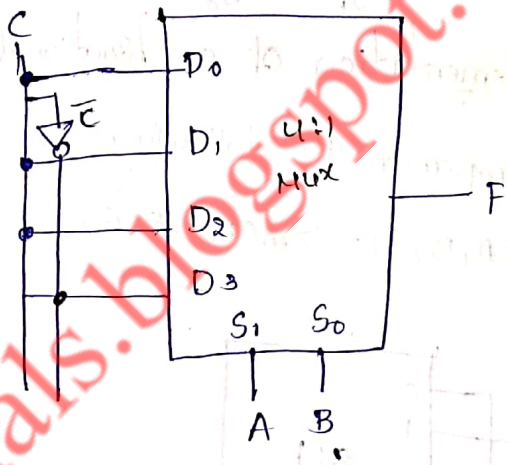


(A) Dual



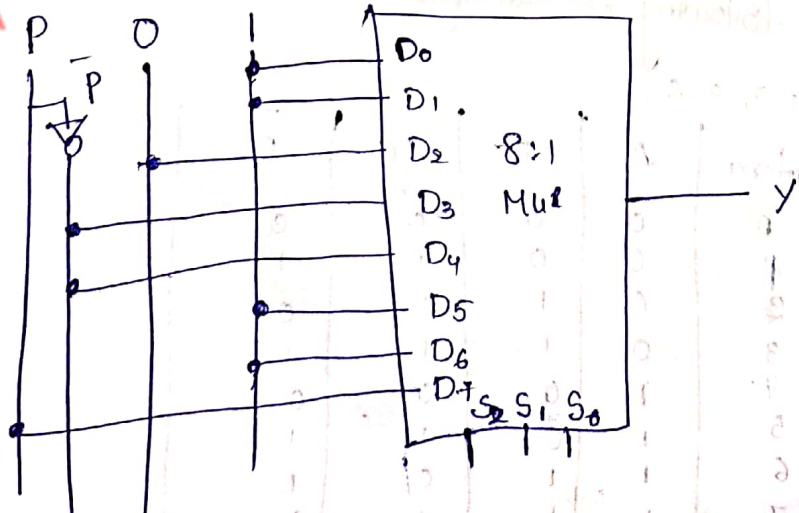
Case ÷ 2





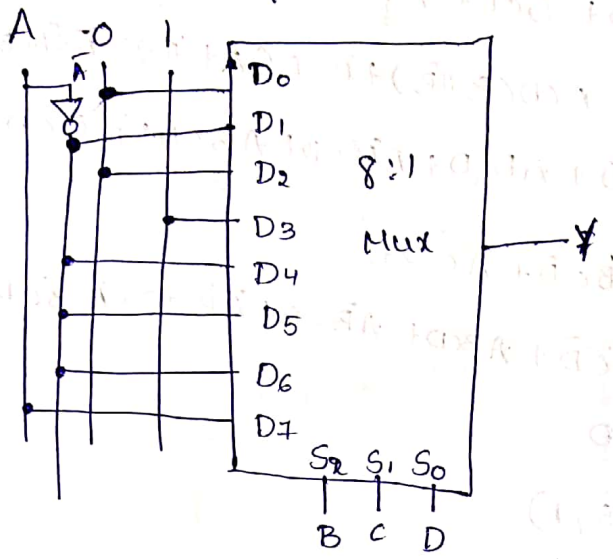
Ex ÷ Implement the following Boolean function using 8:1 Mux. $F(P, Q, R, S) = \Sigma m(0, 1, 3, 4, 8, 9, 15)$

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



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

Implementation table



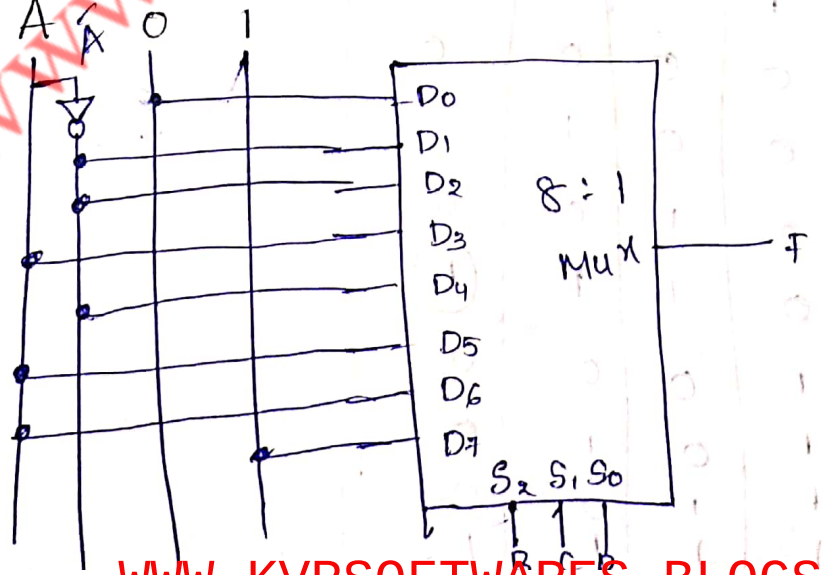
Multiplexer Implementation

* Implement the following Boolean function using 8:1 multiplexer

$F(A,B,C,D) = \Pi M(0,3,8,5,9,10,12,14)$

Here, instead of minterms, maxterms are specified Thus, we have to circle maxterms which are not included in the Boolean function shown below

	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
	0	$\bar{A}$	$\bar{A}$	A	$\bar{A}$	A	$\bar{A}$	1



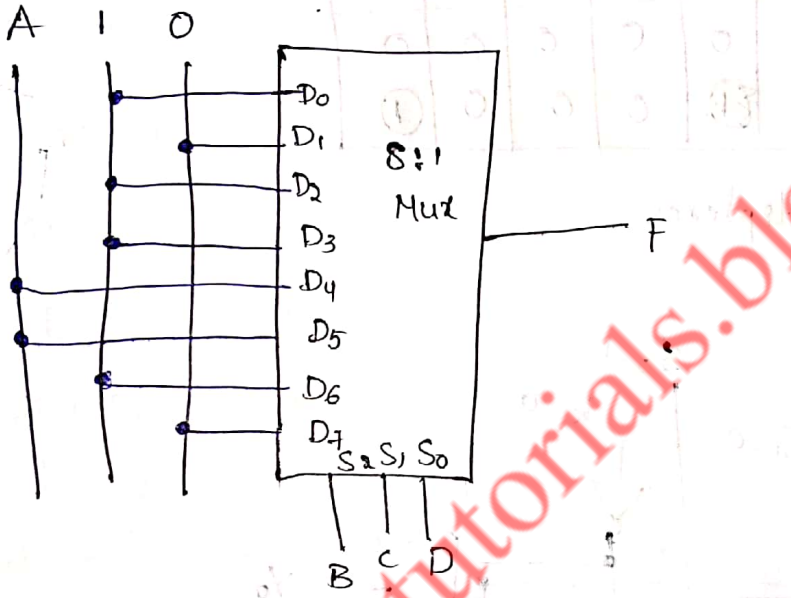
Ex: Implement the following Boolean function with 8:1 Multiplexer

$F(A,B,C,D) = \Sigma m(0,2,6,10,11,12,13) + d(3,8,14)$

Sol:

	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
	1	0	1	1	A	A	0	1

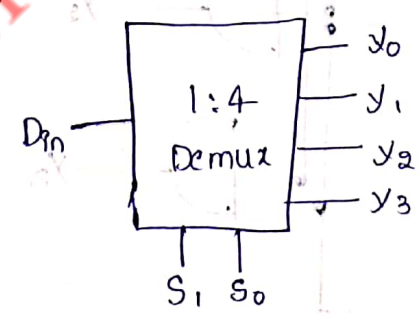
Here don't cares are treated as 1's



Demultiplexer (1 input - n selection lines - 2^n outputs)

1:4 Demux:

Logic symbol:



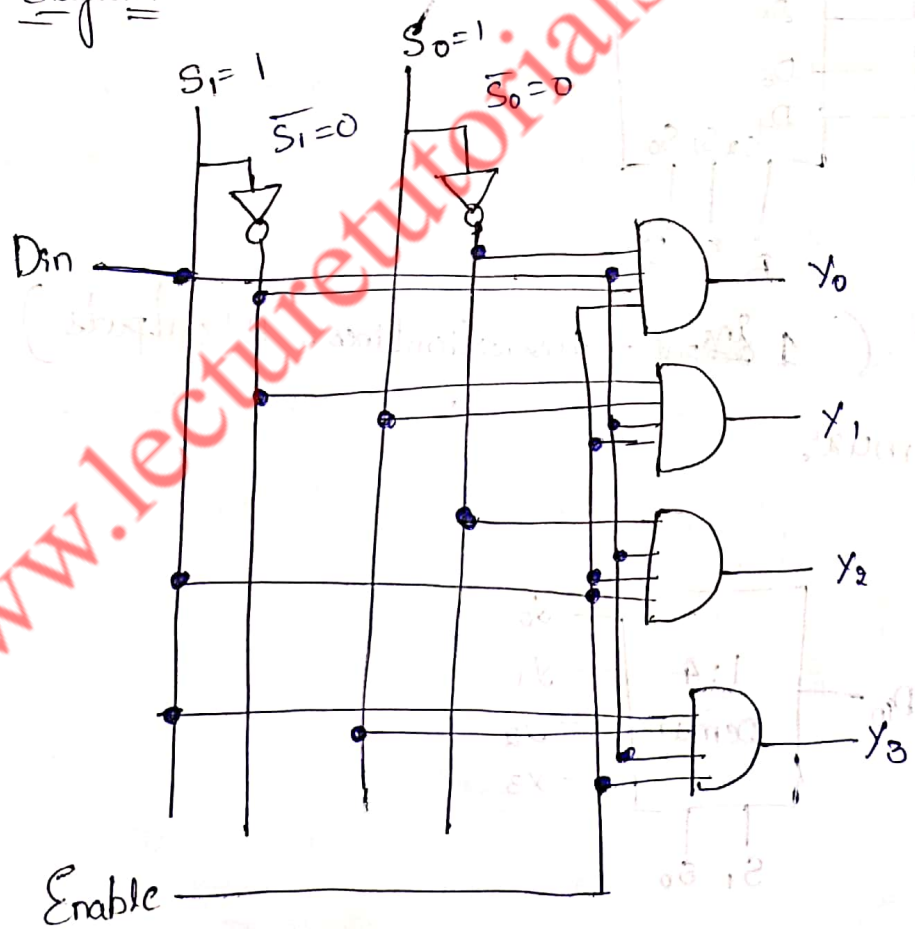
Enable (0) - Inputs are Don't care
outputs are zero

Truth table :-

Enable	S_1	S_0	D_{in}	y_0	y_1	y_2	y_3
0	x	x	x	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	0	0	0	0
1	0	1	1	0	1	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	1

1:4 demultiplexer

Logic diagram



Ex:- Implement the full subtractor by using demultiplexer

Sol:-

A	B	C	D (Difference)	B (Borrow)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

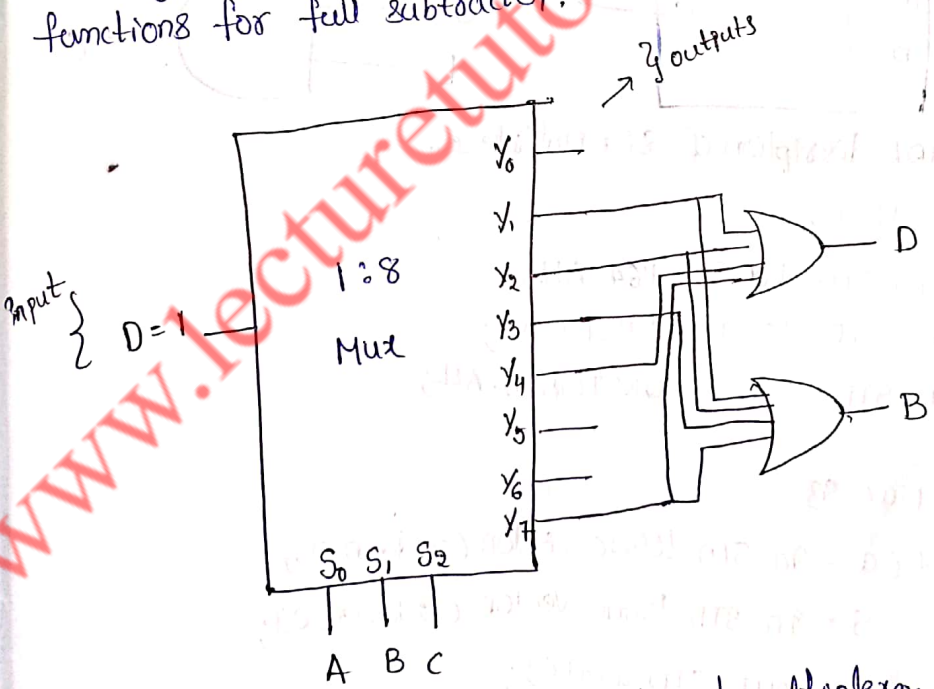
A	B	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

truth table of full subtractor

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

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

With Din input 1, demultiplexer gives minterms at the output So by logically ORing required minterms we can implement Boolean functions for full subtractor.



Full subtractor using 1:8 demultiplexer

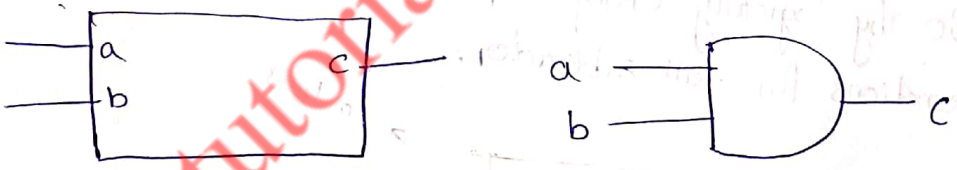
HDL programming :- (Hardware discription language)

Ex:- write the HDL discription of AND gate

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
Use IEEE.STD_LOGIC_ARITH.ALL;
Use IEEE.STD_LOGIC_UNSIGNED.ALL;

Entity andgate is
    port (a,b : in STD_LOGIC;
          c : out STD_LOGIC);
end andgate;
architecture Behavioral of andgate is
begin
c<=a and b;
End Behavioral;
```

Output:-



Ex:- write the HDL discription of 8:1 Multiplexer

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
Use IEEE.STD_LOGIC_ARITH.ALL;
Use IEEE.STD_LOGIC_UNSIGNED.ALL;

Entity Mux is
    port(a : in STD LOGIC VECTOR(7 down 0);
         S : in STD LOGIC VECTOR(2 down 0);
         Y : out STD LOGIC);

End MUX;
architecture Behavioral of MUX is

begin
Y <= a(0) when S = "000" else
     a(1) when S = "001" else
```

```
a(2) when S = "010" else
a(3) when S = "011" else
a(4) when S = "100" else
a(5) when S = "101" else
a(6) when S = "110" else
a(7) when S = "111" ;

End Behavioral;
```

Output:-