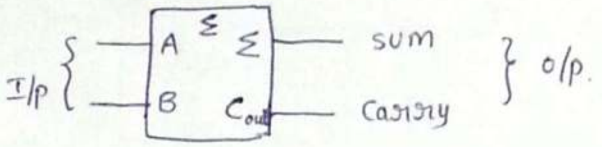
# Unit1 Lecture 1 MUX Demux Encoder Decoder

Computer Organisation and Design (Lovely Professional University)

## THE HALF ADDER :-

→ Logic symbol is

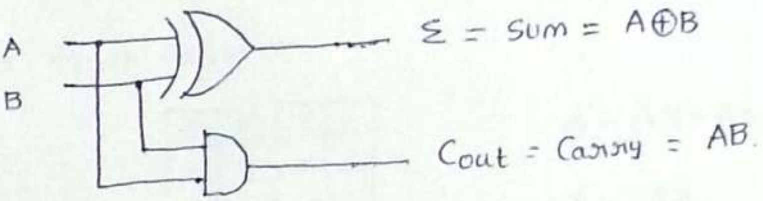


→ Half Adder truth table is

| A | B | Σ | Cout |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

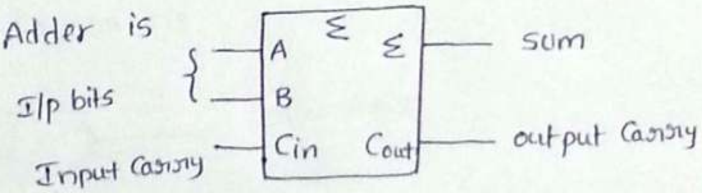
$$\Sigma = \bar{A}B + A\bar{B}$$
$$= A \oplus B$$
$$C_{out} = AB$$

→ Half Adder logic diagram is



## THE FULL ADDER ;-

→ Logic symbol for full Adder is



→ Full Adder truth table is

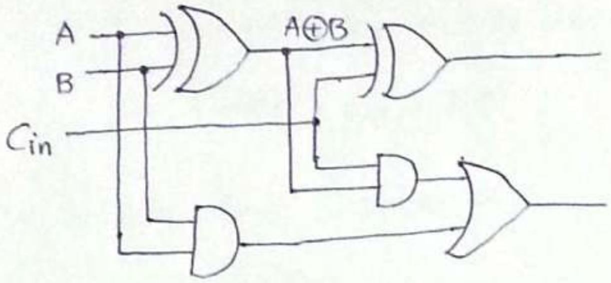
| A | B | Cin | Σ | Cout |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

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

$$\therefore \boxed{\begin{aligned} \Sigma &= A \oplus B \oplus C_{in} \\ C_{out} &= AB + (A \oplus B)C_{in} \end{aligned}}$$

$$C_{out} = \bar{A}BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in} + ABC_{in}$$
$$= (\bar{A}B + A\bar{B})C_{in} + AB(\bar{C}_{in} + C_{in})$$
$$= (A \oplus B)C_{in} + AB.$$

→ Full Adder logic diagram is.

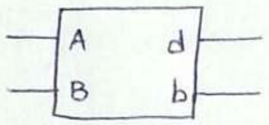


$\Sigma = A \oplus B \oplus Cin$

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

## THE HALF SUBTRACTOR :-

→ Logic symbol is



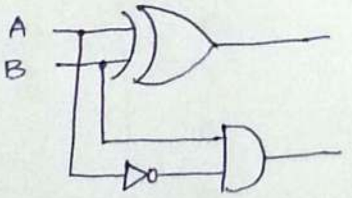
→ Half subtractor truth table is

| A | B | d | b |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

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

$b = \bar{A}B.$

→ Half subtractor logic diagram is

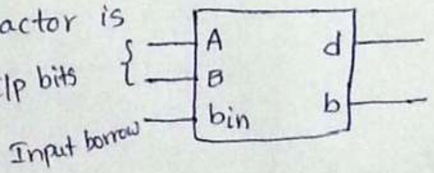


$d = A \oplus B$

$b = \bar{A}B$

## → THE FULL SUBTRACTOR :-

→ Logic symbol for Full subtractor is



→ Full subtractor truth table is

| A | B | bin | 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 |

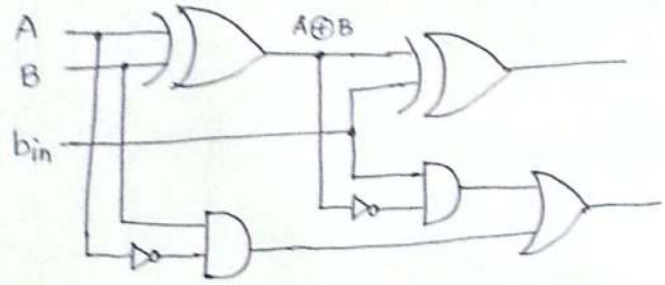
$$\begin{aligned} d &= \bar{A}\bar{B}b_{in} + \bar{A}B\bar{b}_{in} + A\bar{B}\bar{b}_{in} + ABbin \\ &= (\bar{A}\bar{B} + AB)\,bin + (\bar{A}B + A\bar{B})\,\overline{b_{in}} \\ &= (\overline{A \oplus B})\,bin + (A \oplus B)\,\bar{b}in \\ &= A \oplus B \oplus bin \end{aligned}$$

$$b = \bar{A}\bar{B}\,b_{in} + \bar{A}B\,\bar{b}_{in} + \bar{A}B\,b_{in} + AB\,b_{in}$$

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

$$= (\overline{A\oplus B})\,b_{in} + \bar{A}B.$$

→ Full subtractor logic diagram is



$d = A\oplus B\oplus b_{in}$

$b = \bar{A}B + (\overline{A\oplus B})\,b_{in}$

PARITY BIT GENERATOR :-

i) Design of an even parity bit generator. for a 4-bit input. :-

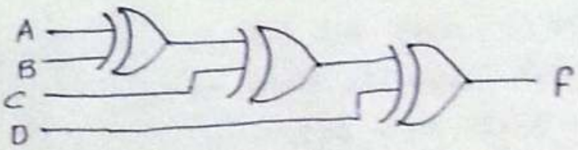
| 4-bit data input A | B | C | D | output parity bit (f) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

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

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | $^{0}$ | 1 $^{1}$ | $^{3}$ | 1 $^{2}$ |
| 01 | 1 $^{4}$ | $^{5}$ | 1 $^{7}$ | $^{6}$ |
| 11 | $^{12}$ | 1 $^{13}$ | $^{15}$ | 1 $^{14}$ |
| 10 | 1 $^{8}$ | $^{9}$ | 1 $^{11}$ | $^{10}$ |

Chess board configuration.

$\therefore\ f = A\oplus B\oplus C\oplus D.$



Logic circuit diagram for even parity generator ↑

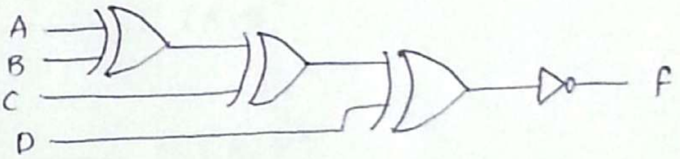
## ⇒ Design of an odd parity bit generator for a 4-bit input

| A | B | C | D | f |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

$f = \Sigma m(0, 3, 5, 6, 9, 10, 12, 15)$

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | $1^{0}$ | $^{1}$ | $1^{3}$ | $^{2}$ |
| 01 | $^{4}$ | $1^{5}$ | $^{7}$ | $1^{6}$ |
| 11 | $1^{12}$ | $^{13}$ | $1^{15}$ | $^{14}$ |
| 10 | $^{8}$ | $1^{9}$ | $^{11}$ | $1^{10}$ |

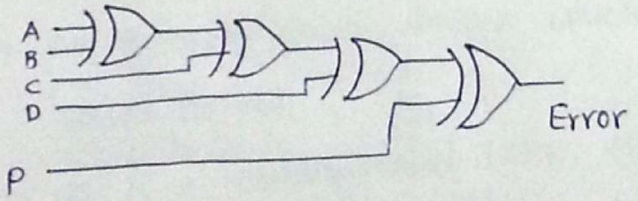
$f = \overline{A \oplus B \oplus C \oplus D}$

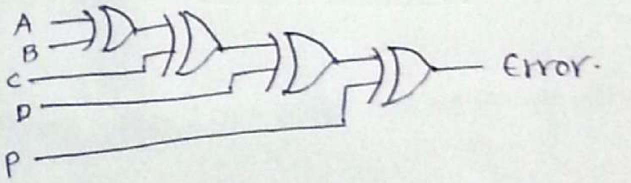


Logic circuit diagram for odd bit Parity generator.

## ⇒ Even parity bit Checker & odd parity bit checker.

→ At the receiving end, if the word received has an even number of 1s in odd parity system (or) an odd number of 1's in even parity system, it implies that an error occured.





**Even parity bit checker**

If we get Error (0) then no error occured P(0). Error(1) then error occured.

**odd parity bit checker**

If we get Error(1) then no error occured. P(0). ∴ Error(0) then error occured.

## 1-bit Magnitude comparator :- (A, B are input bits, E, G, L are o/p bits)

- → If $A = B \Rightarrow E = A \odot B$
- → If $A > B$ : $G = A\bar{B}$
- → If $A < B$ : $L = \bar{A}B$.

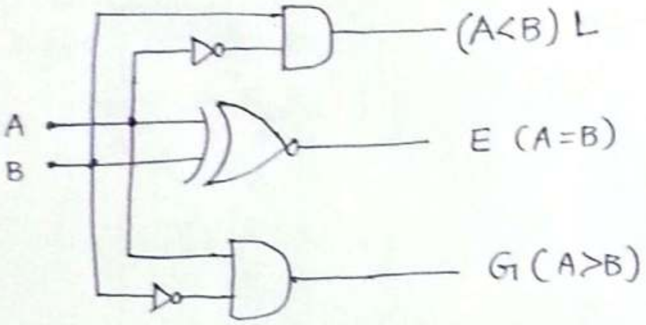
(or)

If $A = B \Rightarrow E = 1$

$A > B \Rightarrow G = 1$

$A < B \Rightarrow L = 1$.

→ Logic circuit diagram for 1-bit magnitude comparator is given by



→ Truth table for 1-bit magnitude comparator is given by.

| A | B | L | E | G |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |

## ⇒ ENCODER :-

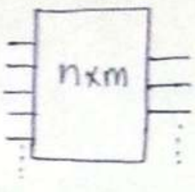
→ An Encoder is a device which converts familiar numbers or symbols into coded format.

→ It is a combinational ckt that performs the reverse operation of decoder

→ It is used to minimize the data lines.

→ out of n inputs, if every time only single input is high, then there is no need to use n inputs, then we can represent same information in output in m bits & m outputs. ($m < n$).

→ Relation between ~~n~~ & ~~m~~ is ~~$n \geq 2^m$~~. (n = ~~no. of input lines~~ )
(m = ~~no. of output lines~~).

→ 

n- inputs

m - outputs.

n > m.

Encoder.

⇒ **octal to Binary Encoder** :- (8-line to 3-line encoder).

→ Truth table is given as.

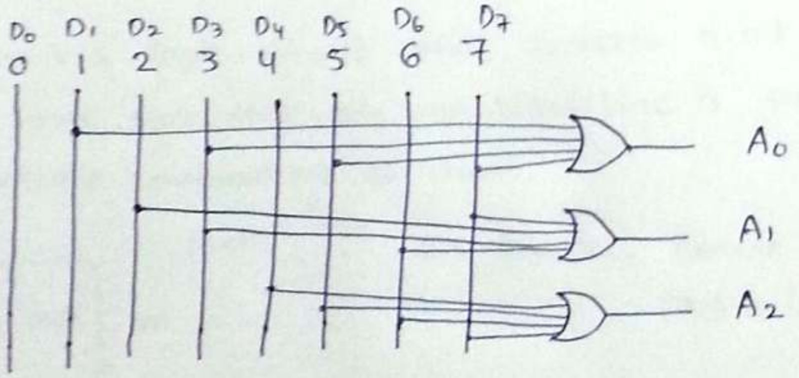
| octal digits | | Binary | | |
|---|---|---|---|---|
| | | $A_2$ | $A_1$ | $A_0$ |
| $D_0$ | 0 | 0 | 0 | 0 |
| $D_1$ | 1 | 0 | 0 | 1 |
| $D_2$ | 2 | 0 | 1 | 0 |
| $D_3$ | 3 | 0 | 1 | 1 |
| $D_4$ | 4 | 1 | 0 | 0 |
| $D_5$ | 5 | 1 | 0 | 1 |
| $D_6$ | 6 | 1 | 1 | 0 |
| $D_7$ | 7 | 1 | 1 | 1 |

$$A_2 = 4+5+6+7$$
$$= D_4 + D_5 + D_6 + D_7$$

$$A_1 = 2+3+6+7$$
$$= D_2 + D_3 + D_6 + D_7$$

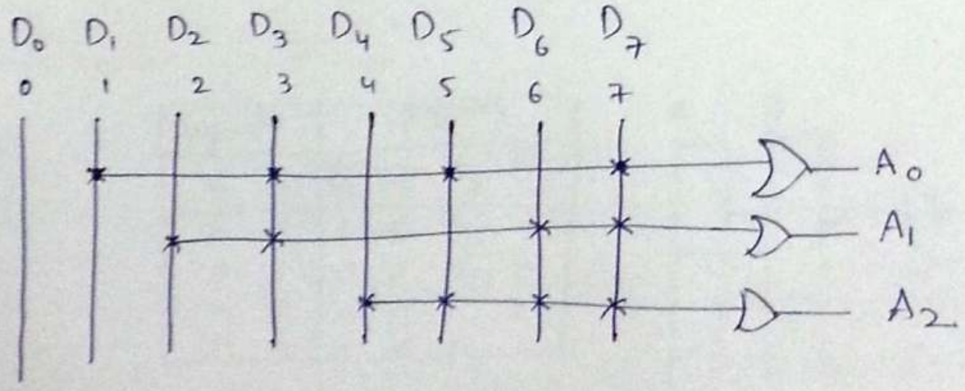
$$A_0 = 1+3+5+7$$
$$= D_1 + D_3 + D_5 + D_7$$

→ Logic diagram for octal- binary encoder is given by.



(write this in exam)

(or)



## Decimal-to-BCD ENCODER :-

→ Truth table is given by

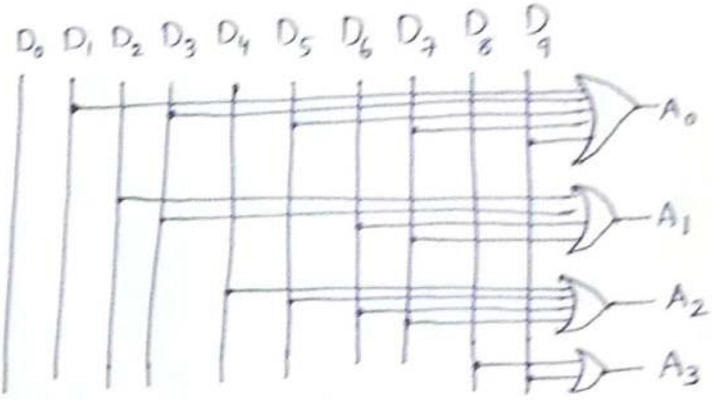
| Decimal inputs | | Binary | | | |
|---|---|---|---|---|---|
| | | $A_3$ | $A_2$ | $A_1$ | $A_0$ |
| $D_0$ | 0 | 0 | 0 | 0 | 0 |
| $D_1$ | 1 | 0 | 0 | 0 | 1 |
| $D_2$ | 2 | 0 | 0 | 1 | 0 |
| $D_3$ | 3 | 0 | 0 | 1 | 1 |
| $D_4$ | 4 | 0 | 1 | 0 | 0 |
| $D_5$ | 5 | 0 | 1 | 0 | 1 |
| $D_6$ | 6 | 0 | 1 | 1 | 0 |
| $D_7$ | 7 | 0 | 1 | 1 | 1 |
| $D_8$ | 8 | 1 | 0 | 0 | 0 |
| $D_9$ | 9 | 1 | 0 | 0 | 1 |

$$A_0 = 1+3+5+7+9$$
$$= D_1 + D_3 + D_5 + D_7 + D_9$$

$$A_1 = 2+3+6+7$$
$$= D_2 + D_3 + D_6 + D_7$$

$$A_2 = D_4 + D_5 + D_6 + D_7$$

$$A_3 = D_8 + D_9$$

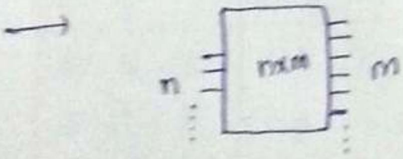


Logic diagram for DEC-BCD ENCODER.

## ⟹ Decoder :-

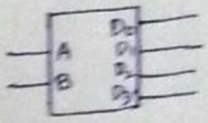
→ A decoder is a logic circuit that converts n-bit binary input code into M output lines such that only one output line is activated for each one of the possible combinations of input.

→ $(n<m)$

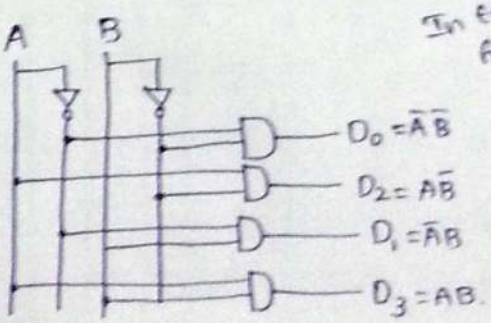


In Decoder, Enable is also present. (Active low & Active high).

### → 2-Line-to-4-Line Decoder :-



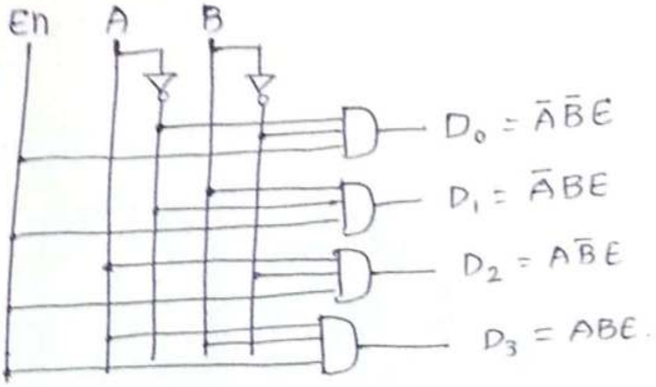
| Inputs | | outputs | | | |
|---|---|---|---|---|---|
| A | B | $D_0$ | $D_1$ | $D_2$ | $D_3$ |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |



In exam follow order
$D_3$
$D_2$
$D_1$
$D_0$

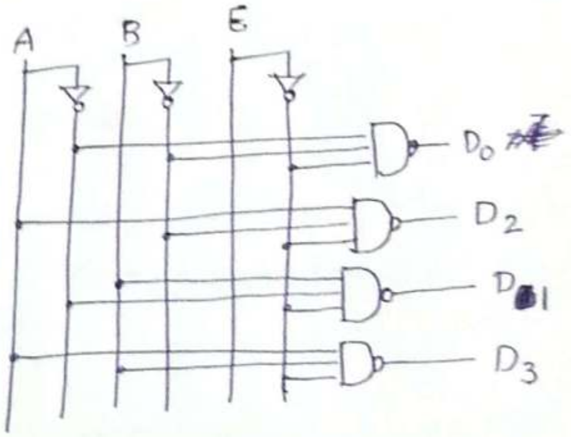
→ Truth-table with Active high enable

| E | A | B | $D_0$ | $D_1$ | $D_2$ | $D_3$ |
|---|---|---|---|---|---|---|
| 0 | x | x | 0 | 0 | 0 | 0 |
| 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 |



→ Truth table with Active ~~high~~ LOW enable with NAND gates.

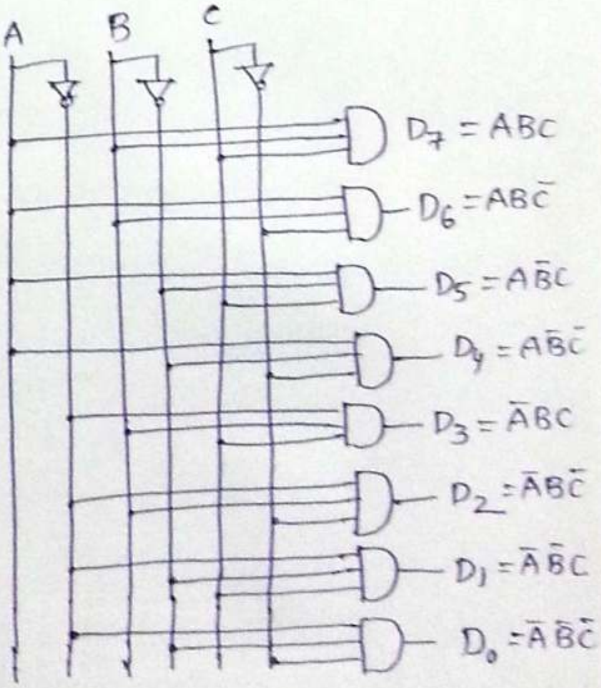
| E | A | B | $D_0$ | $D_1$ | $D_2$ | $D_3$ |
|---|---|---|---|---|---|---|
| 1 | x | x | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |



⟹ 3-Line-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 |
| 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 |

Truth-table (with No enable & Active high enable)



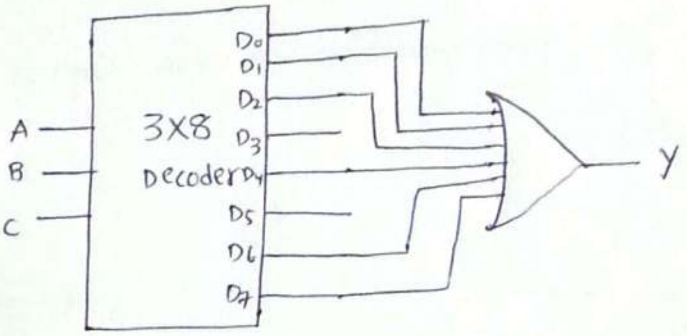
LOGIC DIAGRAM.

* Implement the following Boolean function with Decoder.

$$Y(A,B,C) = \Sigma m(0,1,2,4,6,7)$$

Sol:- Truth table is

| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |



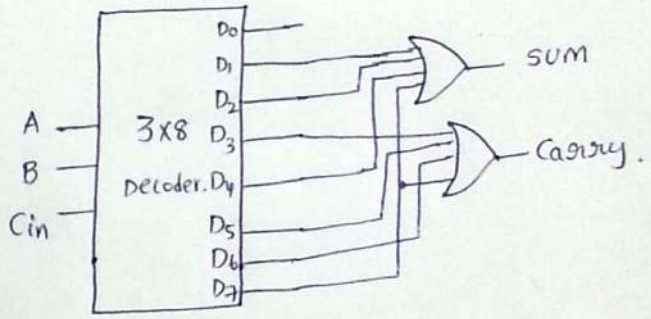
* Implement or Design a Full Adder using Decoder.

Sol:- Truth table for Full Adder is

| 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 |

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

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



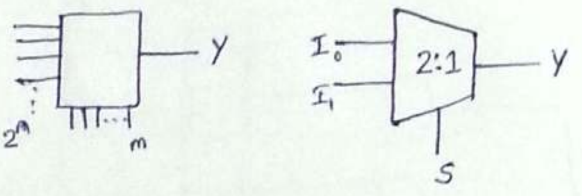
## Multiplexer :- (Data selector)

→ It is a combinational circuit that selects binary information from one of many input lines and directs it to o/p line.

→ It is simply a Data selector.

→ It has $2^m = n$ inputs and one output and m selection lines. $(2^m : 1)$

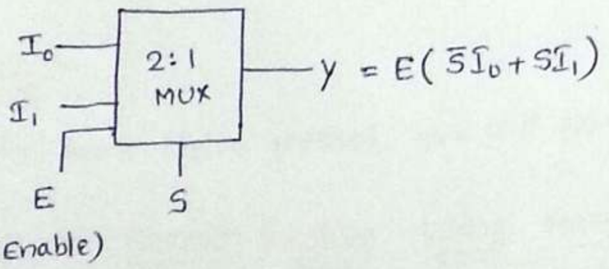
→ Logic symbol for Multiplexer is



→ Multiplexer shortly known as MUX.

## ⟹ 2 : 1 MUX :-

→ Logic symbol for 2:1 MUX is

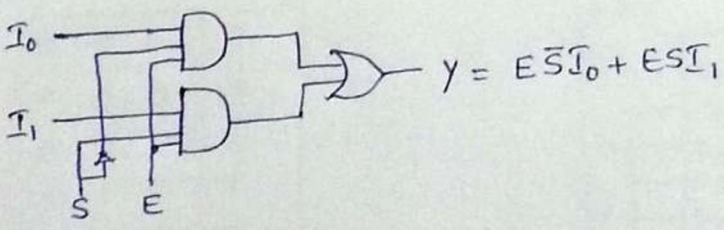


Truth table for 2 : 1 MUX is

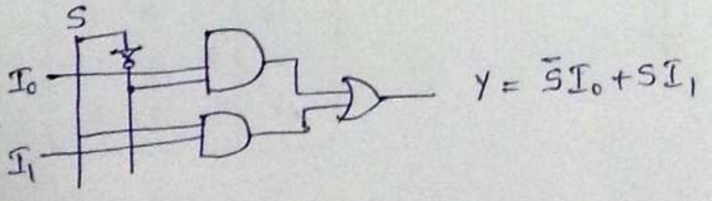
| E | S | Y |
|---|---|---|
| 0 | X | 0 |
| 1 | 0 | $I_0$ |
| 1 | 1 | $I_1$ |

→ Logic function of 2:1 MUX is $y = E(\bar{S}I_0 + SI_1)$

→ Logic circuit diagram is given by

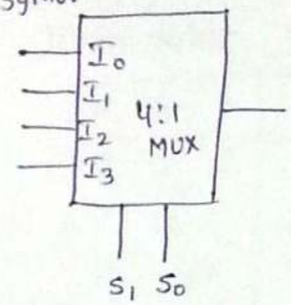


→ If we neglect Enable,



## 4:1 MUX :-

→ logic symol



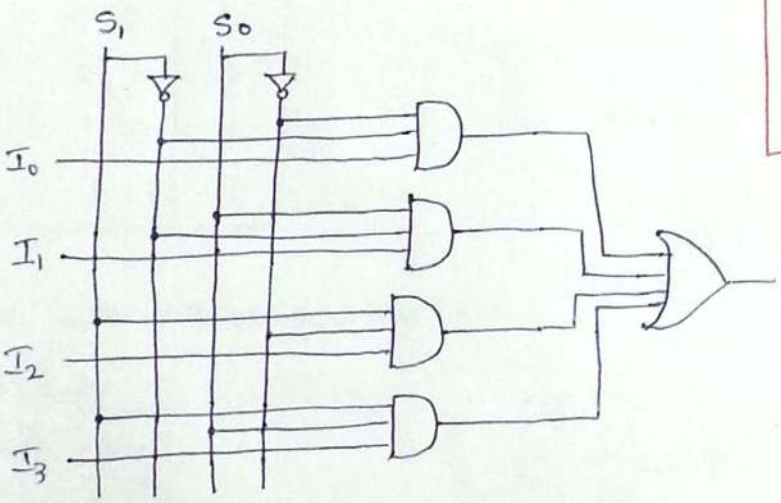
↓ Logic function

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

Truth table is given by

| $S_1$ | $S_0$ | Y |
|---|---|---|
| 0 | 0 | $I_0$ |
| 0 | 1 | $I_1$ |
| 1 | 0 | $I_2$ |
| 1 | 1 | $I_3$ |

→



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

## 8:1 MUX :-

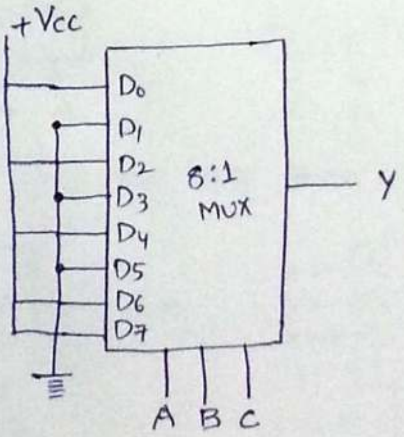
→ Apply same above method, you will get 8:1 MUX also.

⇒ Design of Boolean function using Multiplexer (4:1 & 8:1 MUX)

* Implement or Design $Y(A,B,C) = \Sigma m(0,2,4,6,7)$ using 8:1 MUX.

Sol:- Truth table of $Y(A,B,C)$ is

| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |



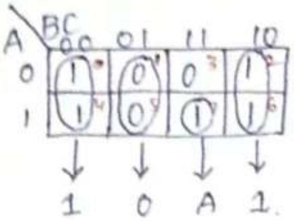
+Vcc means 1

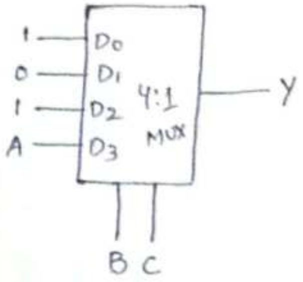
GND Means 0.

* Implement or Design $Y(A,B,C) = \Sigma m(0,2,4,6,7)$ using 4:1 MUX.

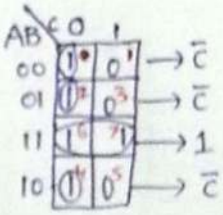
Sol:- Truth table is

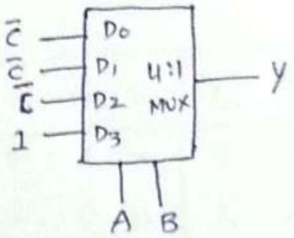
| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |





If we take AB as selection lines,





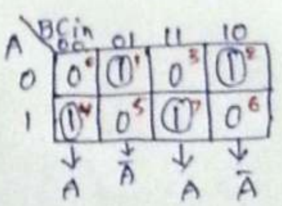
* Implement or Design Full Adder using 4:1 MUX.

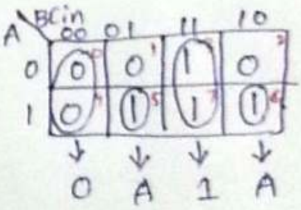
→ Truth table of Full Adder is

| 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 |

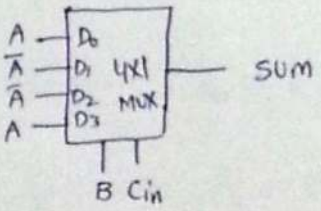
$SUM = \Sigma m(1,2,4,7)$

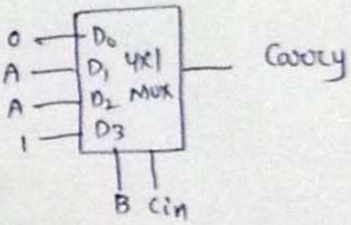
$Carry = \Sigma m(3,5,6,7)$

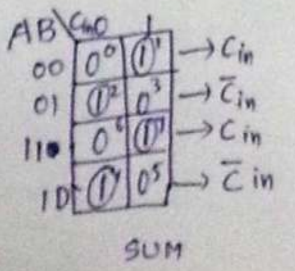


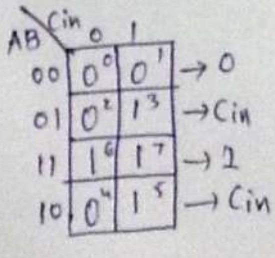


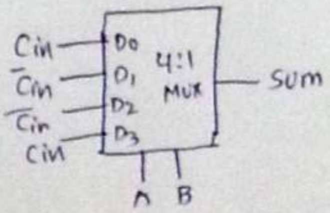
If B, Cin are selection lines then

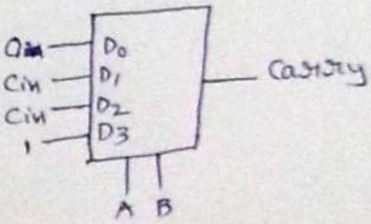






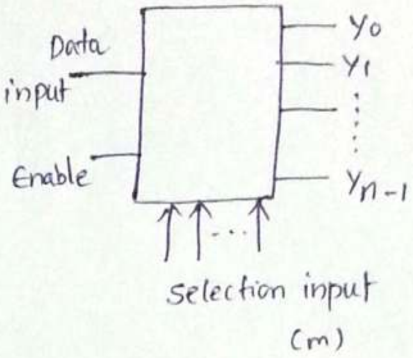




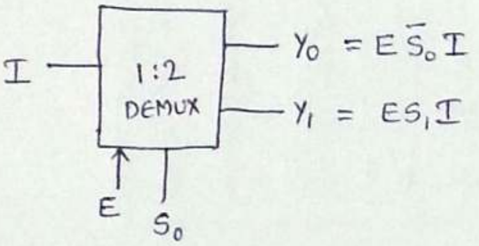


## ⇒ DEMULTIPLEXER :- (Data Distributor)

- → 1 input and many outputs. ($1:2^m$)
- → Reverse operation of Multiplexer.
- → Data Distributor



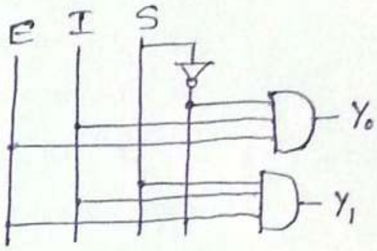
### i) 1:2 DEMUX :-



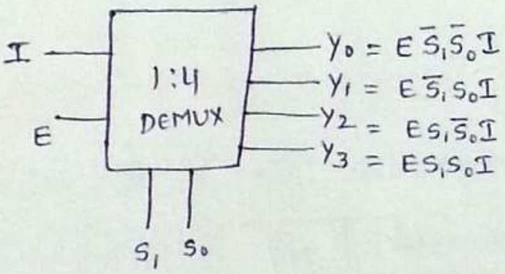
$Y_0 = E\bar{S}_0 I$

$Y_1 = E S_1 I$

| E | $S_0$ | $Y_0$ | $Y_1$ |
|---|---|---|---|
| 0 | x | 0 | 0 |
| 1 | 0 | I | 0 |
| 1 | 1 | 0 | I |



### ii) 1:4 DEMUX :-



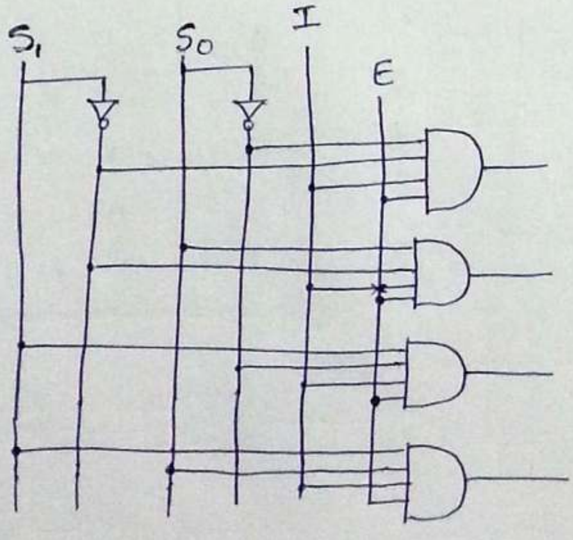
$Y_0 = E\bar{S}_1\bar{S}_0 I$

$Y_1 = E\bar{S}_1 S_0 I$

$Y_2 = E S_1\bar{S}_0 I$

$Y_3 = E S_1 S_0 I$

| E | $S_1$ | $S_0$ | $Y_0$ | $Y_1$ | $Y_2$ | $Y_3$ |
|---|---|---|---|---|---|---|
| 0 | x | x | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | I | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | I | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | I | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | I |



$Y_0 = E\bar{S}_0\bar{S}_1 I$

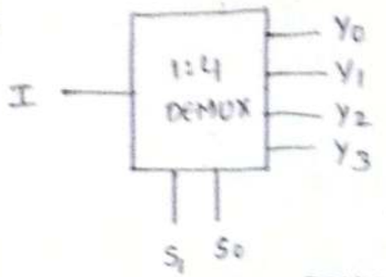
$Y_1 = E S_0\bar{S}_1 I$

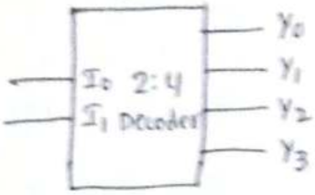
$Y_2 = E S_1\bar{S}_0 I$

$Y_3 = E S_1 S_0 I$

## ⇒ Demultiplexer as a Decoder :-

→





DEMUX TRUTH TABLE

| $S_1$ | $S_0$ | $Y_0$ | $Y_1$ | $Y_2$ | $Y_3$ |
|---|---|---|---|---|---|
| 0 | 0 | I | 0 | 0 | 0 |
| 0 | 1 | 0 | I | 0 | 0 |
| 1 | 0 | 0 | 0 | I | 0 |
| 1 | 1 | 0 | 0 | 0 | I |

| $I_1$ | $I_0$ | $Y_0$ | $Y_1$ | $Y_2$ | $Y_3$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

→ * In demux, if Input (I) = 1 and $S_1 = I_1$ and $S_0 = I_0$ then DEMUX act as Decoder.

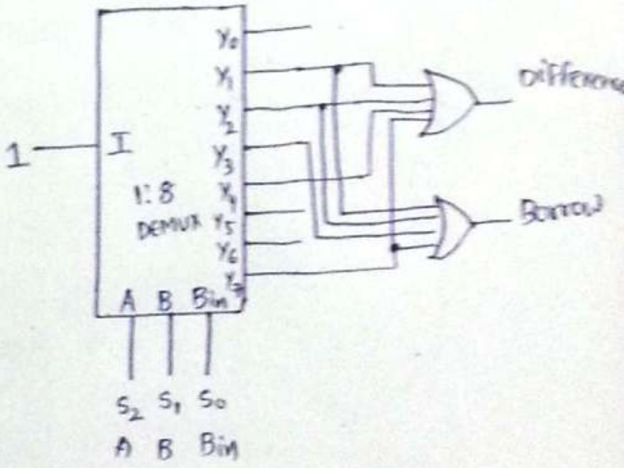
## * Design a Full Subtractor using 1:8 DEMUX

Truth table of Full subtractor is given as

| A | B | Bin | Difference | 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 |

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

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



* Design a Boolean expression $Y(A,B,C) = AB+C$ using 1:8 DEMUX.

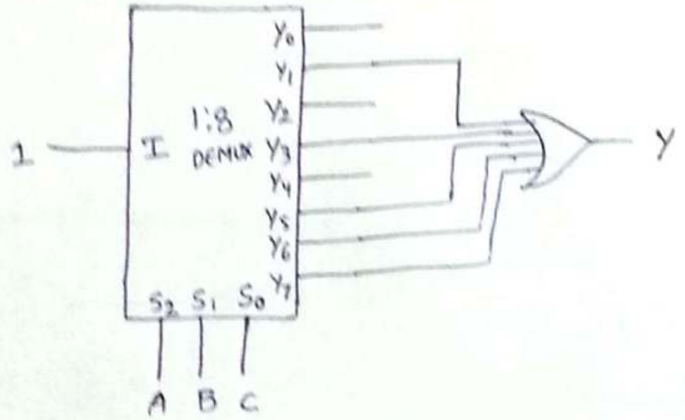
Sol:- $Y(A,B,C) = AB+C$

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

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

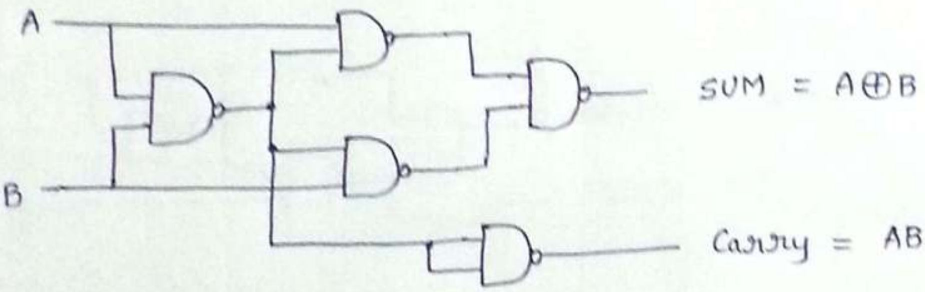
$= ABC + AB\bar{C} + A\bar{B}C + \bar{A}BC + \bar{A}\bar{B}C$

$= \Sigma m(1,3,5,6,7)$



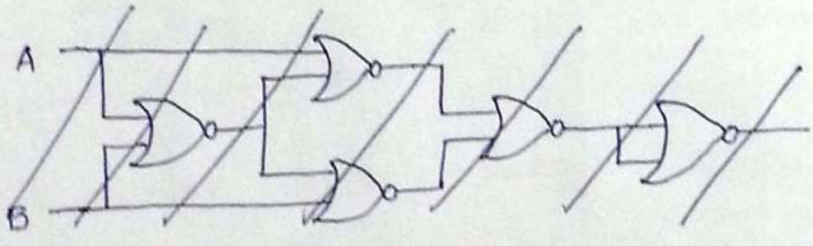
⟹ Design Half Adder using only NAND gates.

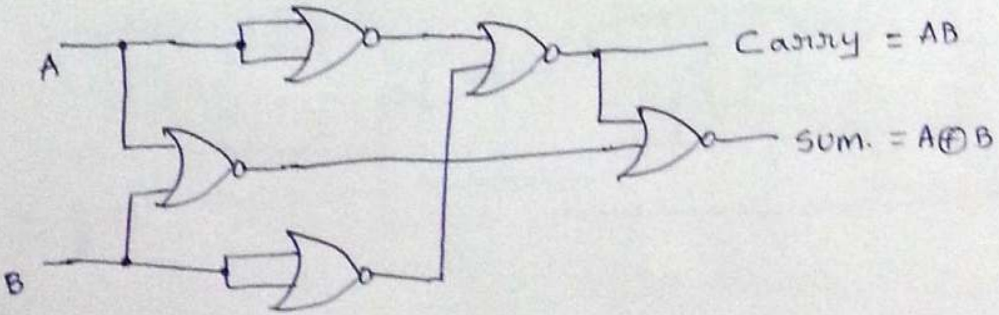
Sol:-



⟹ Design Half Adder using only NOR gates.

Sol:-



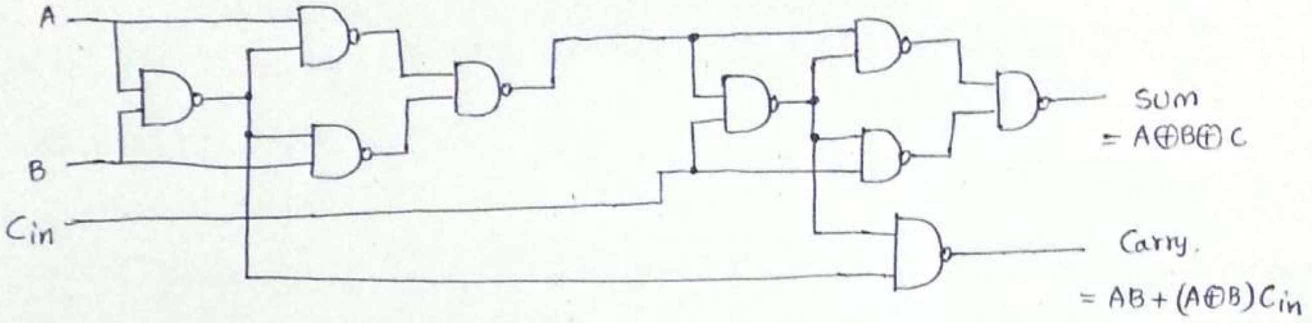


* Design Full Adder using only

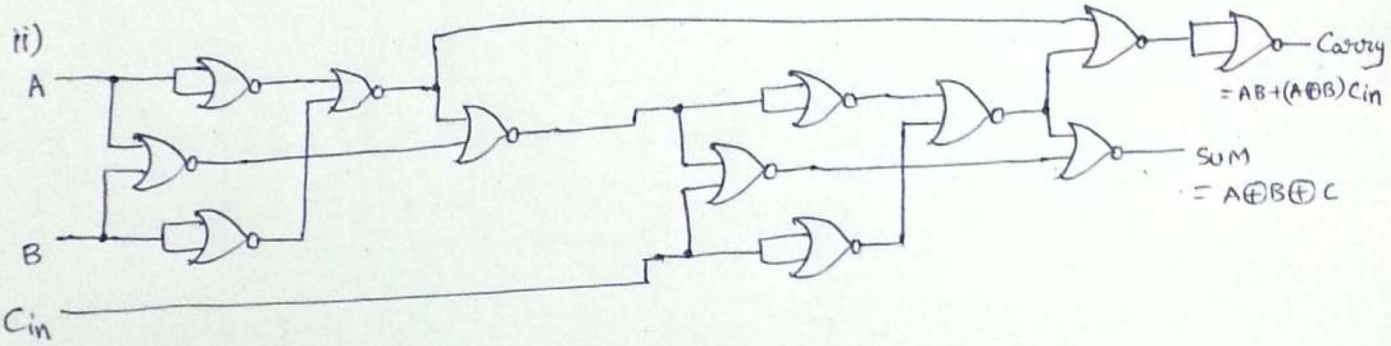
i) NAND gates

ii) NOR gates.

sol :- (i)

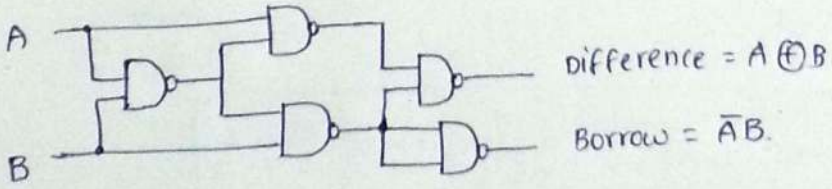


ii)

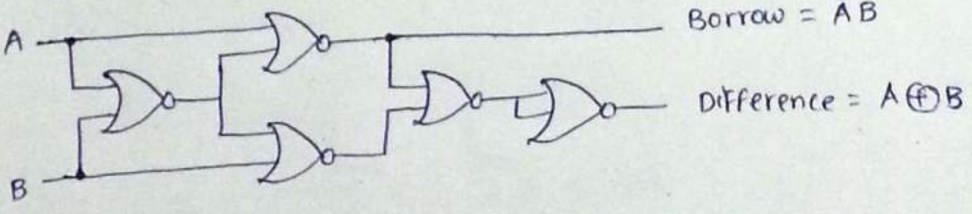


* Design Half subtractor using only i) NAND ii) NOR.

sol:- i)

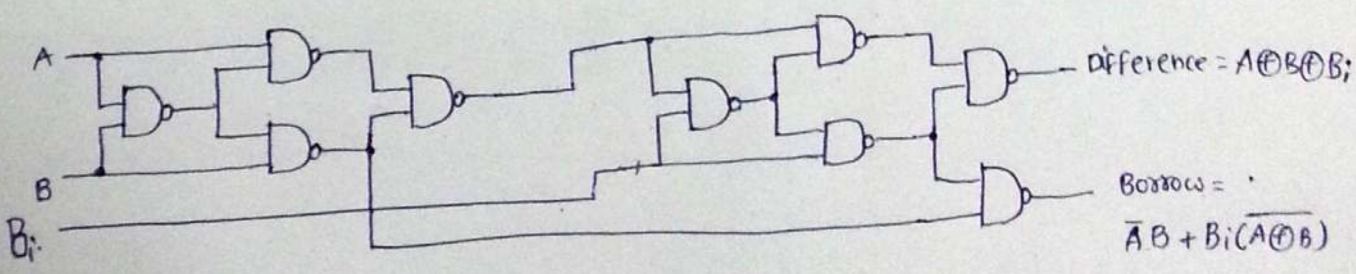


ii)

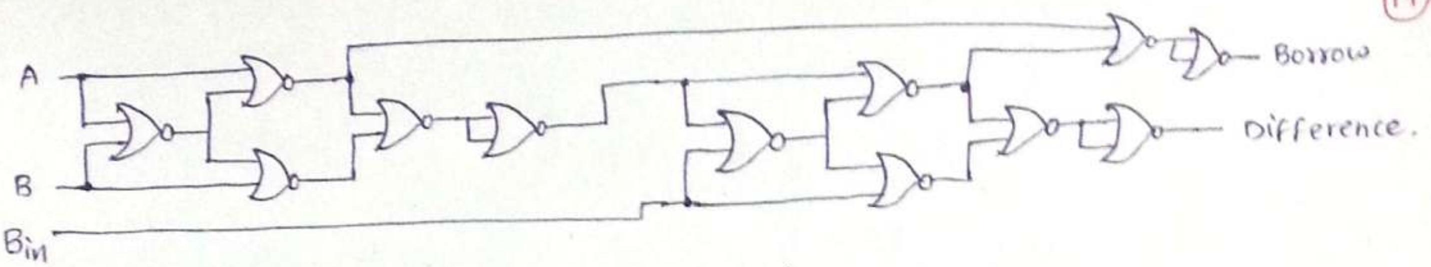


* Design Full subtractor Using only i) NAND ii) NOR.

sol:- i)



ii)

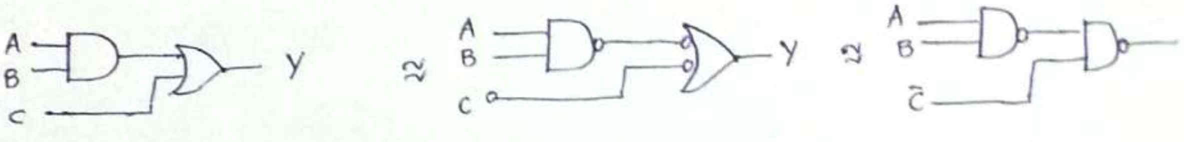


* Implement the following Boolean function $Y(A,B,C) = AB+C$ using only

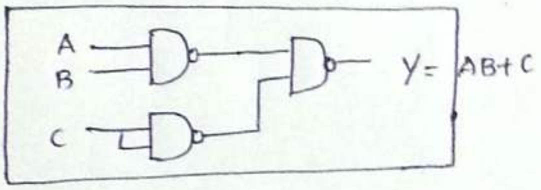
i) NAND gates   ii) NOR gates.

Sol:-

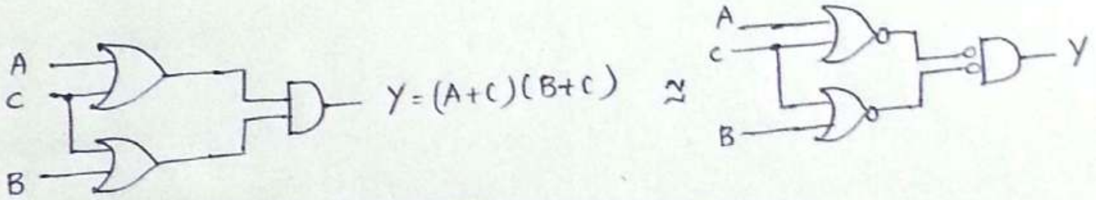
i)

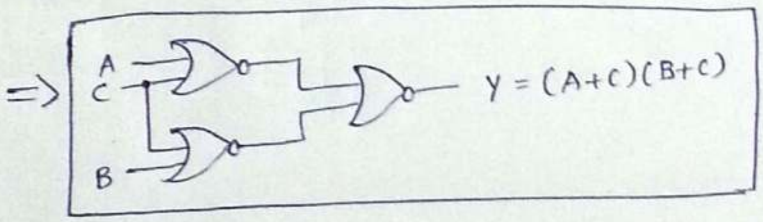


(or)

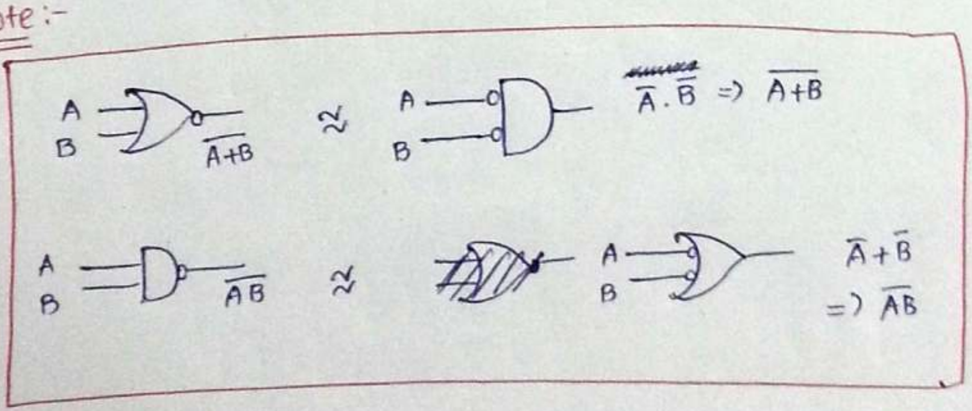


ii) $Y(A,B,C) = (A+C)(B+C)$.





Note:-



$\overline{A}.\overline{B} \Rightarrow \overline{A+B}$

$\overline{A}+\overline{B}$
$\Rightarrow \overline{AB}$