

Computer Organisation & Architecture

positive logic and negative logic:

→ positive logic:

Here, we denote high voltage is 1 & low voltage is represented as 0.

eg →

| A | B | $y = A \text{ op }^x B$ |
|---|---|-------------------------|
| L | L | L |
| L | H | L |
| H | L | L |
| H | H | H |

→

| A | B | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

↓
AND op^x .

| |
|----------|
| High → 1 |
| Low → 0 |

→ Negative logic:

| |
|----------|
| High → 0 |
| Low → 1 |

| A | B | $y = A \text{ op }^y B$ |
|---|---|-------------------------|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

→ OR op^y .

the logic \rightarrow AND opr.
+ -ve logic \rightarrow OR opr.

we ~~are~~ ^{always} use the logic for developing digital circuit.

\rightarrow Basic logic gates :

- 1) OR ($A+B$)
- 2) AND ($A \cdot B$)
- 3) NOT (\bar{A})

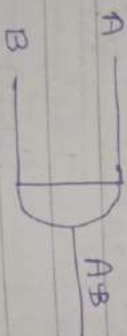
1) OR gate

| A | B | $Y = A+B$ |
|---|---|-----------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |



2) AND gate

| A | B | $Y = A \cdot B$ |
|---|---|-----------------|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

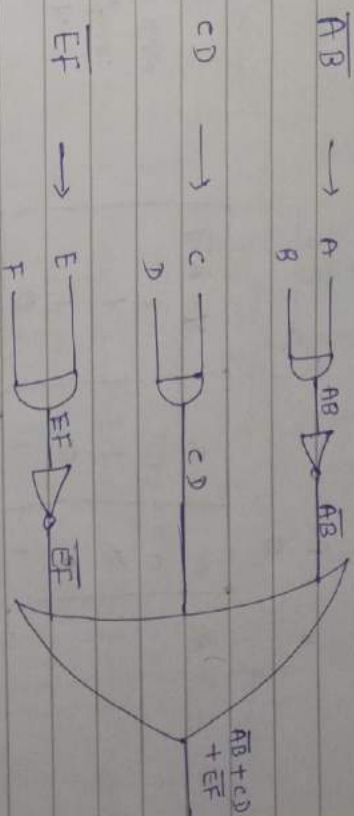


3) NOT gate

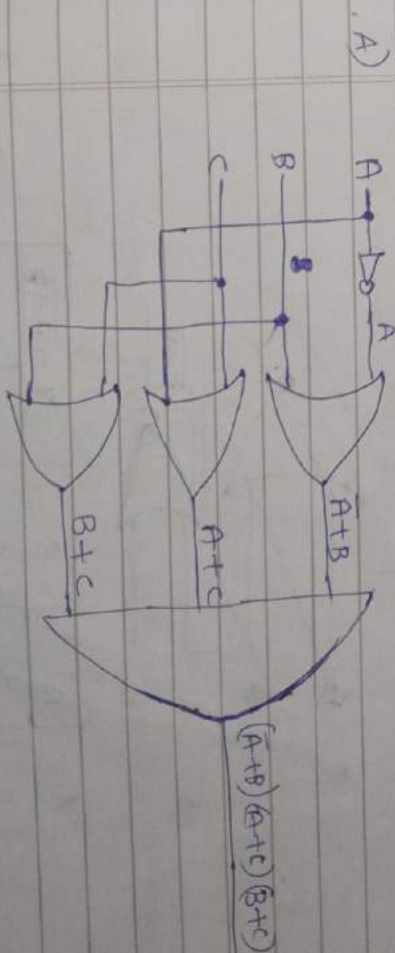
| A | $Y = \bar{A}$ |
|---|---------------|
| 0 | 1 |
| 1 | 0 |



- Q
- 1) $F = \bar{A}\bar{B} + CD + \bar{E}F$
6 inputs $\rightarrow A, B, C, D, E, F$



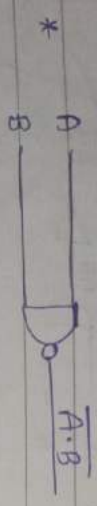
- 2) $F = (\bar{A}+B)(A+C)(B+C)$



* Universal Gates :

1) NAND :

* Combination of AND gate NOT gate



| A | B | $Y = \overline{A \cdot B}$ |
|---|---|----------------------------|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

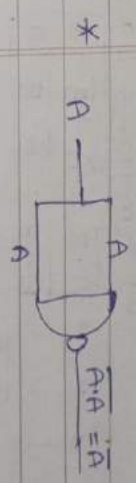
AND-gate and complement
 $Y = \overline{A \cdot B}$

NOT gate has only 1 input

→ Universal Gates properties : (NAND).

① * NOT gate implementation with NAND gate

* NOT → 1 input
NAND → 2 input

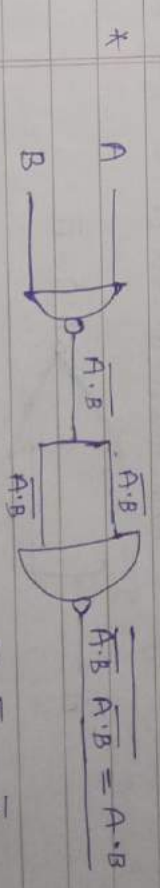


$A \cdot A = A$
 $\therefore A \cdot A = \overline{A}$

$\overline{A} \rightarrow$ NOT

② * AND gate implementation with NAND gate:

* AND → 2 input
NAND → 2 input



$\overline{A \cdot A} = \overline{A}$
 $\overline{A \cdot B} \cdot \overline{A \cdot B} = \overline{A \cdot B}$

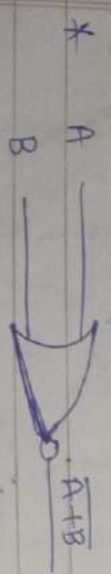
$\overline{A \cdot B} \cdot \overline{A \cdot B} = \overline{A \cdot B}$

③ * OR gate implementation with NAND gate:

* OR → 2 inputs
NAND → 2 inputs

2) NOR :

* Combination of OR and NOT gate



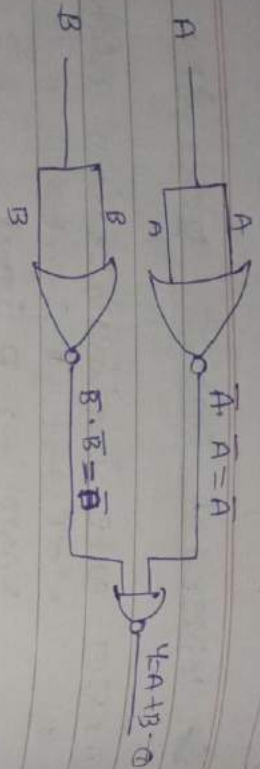
| A | B | $Y = \overline{A + B}$ |
|---|---|------------------------|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

OR-gate and complement
 $Y = \overline{A + B}$

③ * OR gate implementation with NAND gate:

* OR → 2 inputs
NAND → 2 inputs

(vi) \rightarrow gate.



①

$$\bar{A} \cdot \bar{B} = \overline{A+B}$$

by DeMorgan's Law,

$$\bar{A} \cdot \bar{B} \rightarrow \overline{A+B} = \bar{A} + \bar{B}$$

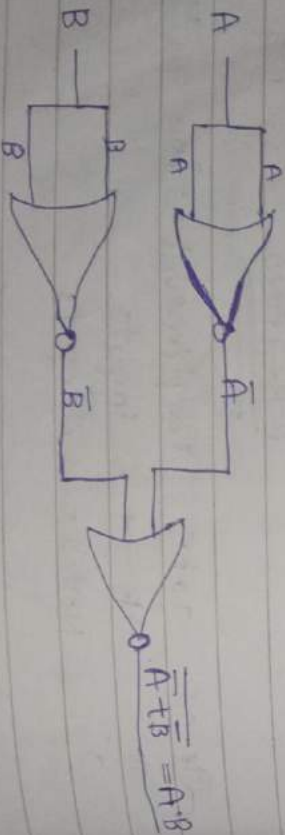
$$\bar{A} + \bar{B} = \overline{A \cdot B}$$

\rightarrow Universal gate properties (NOR):

① NOT gate implementation with NOR gate



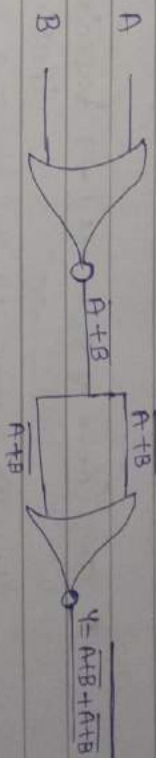
② AND gate implementation with NOR (vi).



$$\bar{A} + \bar{B} = \overline{A+B} = \bar{A} \cdot \bar{B} = A \cdot B$$

(by DeMorgan's)

③ ~~NOR~~ OR (vi) implementation with NOR (vi):



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

(DeMorgan's)

\rightarrow Exclusive OR \rightarrow XOR \rightarrow \oplus symbol
Exclusive NOR \rightarrow XNOR \rightarrow \odot symbol

* XOR:

* It produces 1st = 1 when odd no. of inputs are 1.



* $Y = A \oplus B$

| A | B | $Y = A \oplus B$ |
|---|---|------------------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

odd \rightarrow
 $0+1=1$
 $1 \rightarrow$ odd $\rightarrow 1$

* SOP (Sum of product) \rightarrow based on this truth table.

$Y = A \oplus B$ from column \rightarrow 0 & 1 values
 example \rightarrow then 0 & 1 complement
 example \rightarrow 1 for 5 value.
 eg $\rightarrow Y = \bar{A}B + A\bar{B}$ (above T-table).

* POS (product of sum) \rightarrow based on this truth table.

$Y = A \oplus B$ from column \rightarrow 0 values
 example \rightarrow then 1 & 0 complement
 eg $\rightarrow Y = (\bar{A} + B)(A + \bar{B})$ (above T-table)

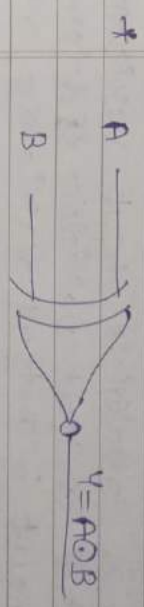
* XNOR:

* produces 1 when even no of inputs are 1.

* $Y = A \odot B$

| A | B | $Y = A \odot B$ |
|---|---|-----------------|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

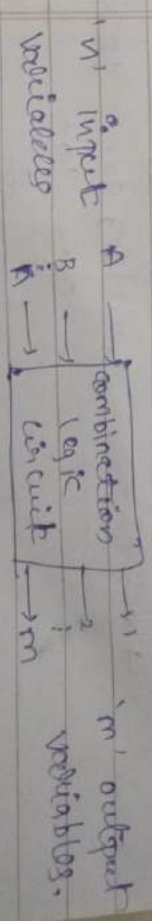
$0+0=1$
 $0+1=0$
 $1+1=1$



* SOP $Y = \bar{A}\bar{B} + AB$ (consider 1)
 POS $Y = (\bar{A} + \bar{B})(A + B)$ (consider 0)

\rightarrow combinational logic circuit:

* It consists of input variables, logic gate & output variables
 * logical gate accept signals from inputs & generate signals to the outputs.
 * A block diagram of c. logic circuit:



* In C-logic circuit depending on the input, it produces same output.

* There is no memory element to store previous element's output.

* Step to develop C.L circuit:

- The problem is stated.
- The no. of available input variables & required output variables is determined.
- The input & output variables are assigned letter symbols.
- The truth table that defines the required relationship b/w input & output is derived.
- The simplified boolean (C) is for each output is obtained.
- The logic diagram is drawn.

1. Adder: (for adding purpose)

→ Half Adder → 2 inputs, 2 outputs possible

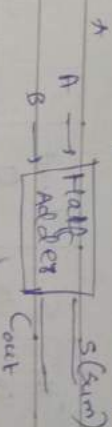
$$\begin{array}{r} 10 \\ 10 \\ \hline 20 \end{array}$$

→ Full Adder → 3 inputs, 2 outputs possible.

$$\begin{array}{r} 10 \\ 10 \\ \hline 20 \end{array}$$

1) Half Adder:

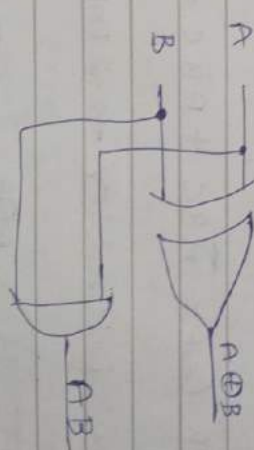
* 2 inputs & 2 outputs



| A | B | S | Cout |
|---|---|---|------|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

→ SOP $S = \bar{A}B + A\bar{B} = A \oplus B$
 SOP $Cout = AB$

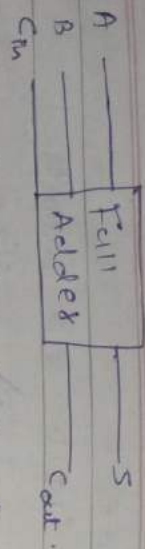
* Half adder circuit:



2) Full Adder:

* used for addition

* 3 inputs & 2 outputs.



| A | B | Cin | S | 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 |

$S = 0 \rightarrow 01010$

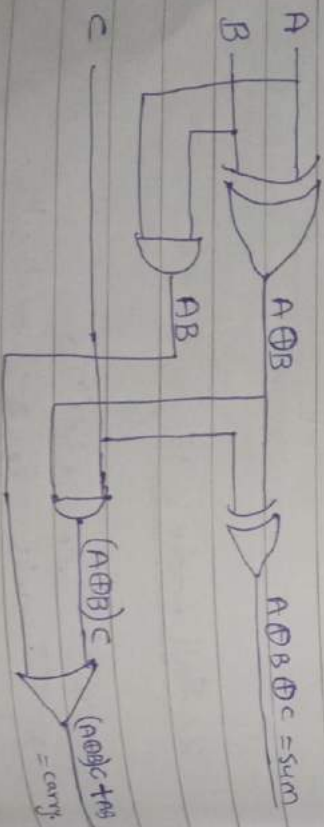
$1+1 = 01$

$1+1+0 = 01+1 = 11$

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

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

To draw full Adder \rightarrow 2 half adders & 1 OR gate



\rightarrow Binary parallel Adder (Ripple carry adder):



For 4 bit Full Adder we need 4 Full Adders.

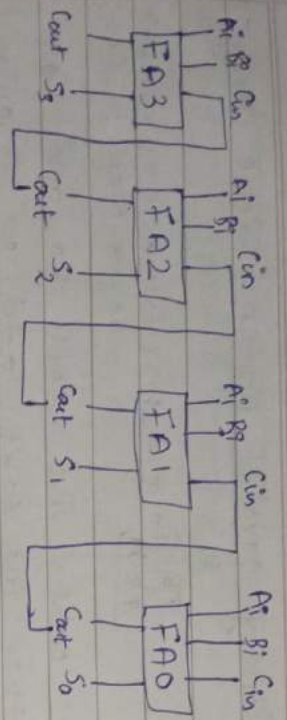
When we want to add 1011 & 0011 then we can't add with only 1 Full adder. So, here we are using series of Full adder.

There is 'n' bits, then 'n' no. of full adder needed.

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

LSB \rightarrow 10

Here, last significant bit always 0 (carry) there is no addition before. The carry (Cout) will move to Cin after each bit addition. Here, we are using 4 full adders bcz it has 4 bits.



Sum (S) = 0 1 1 0
Carry (Cout) = 0 0 1 1

~~Carry out with Cin = 0011 (Carry) and 0110 (Sum) = 10001 (10) in decimal~~

In this circuit FA1 will be worked only after getting the 1st Cout from FA0 & so on, so there is a delay for this propagation.
The delay is from left to right. (\leftarrow)
This propagation is \rightarrow carry propagation delay, so that's why it is also \rightarrow ripple carry adder.

\rightarrow Decoder:

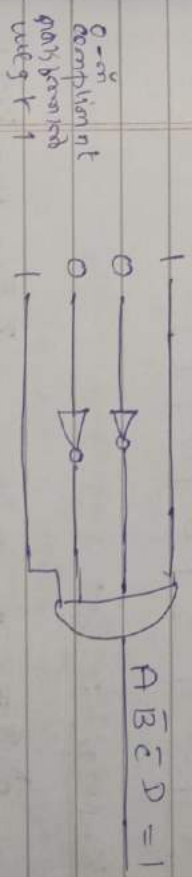
- * It is combinational logic circuit.
- * The basic C of decoders is detects a presence of some input combinations & indicates specific

- * Output level.
- * If there is a n input combination then more $m \leq 2^n$ output combination.
- * eg $\rightarrow 2 \times 4, 3 \times 8, 4 \times 16$ (2 input - 4 output)

a) Basic Binary Decoder:

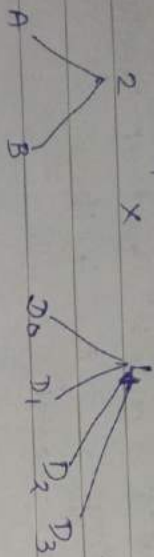
eg \rightarrow 1001 combination assigned 2 outputs
[1 0 1 0, even number assignments \rightarrow 0 produce]

logical circuit:



~~here we are using AND gate~~
for eg, take the combination 1001 = 1 and for all other combination it produces 0.
In this eg we are using AND gate (010 produces 0)
(egs input - 101 combination AND gate used)

2x4 line decoder:
* 2 inputs & 4 outputs



* Truth table:

| 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 |

using K-map:

| A \ B | \bar{B} | B |
|-----------|----------------------|----------------|
| \bar{A} | $\bar{A}\bar{B} D_0$ | $\bar{A}B D_1$ |
| A | $AB D_2$ | $AB D_3$ |

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

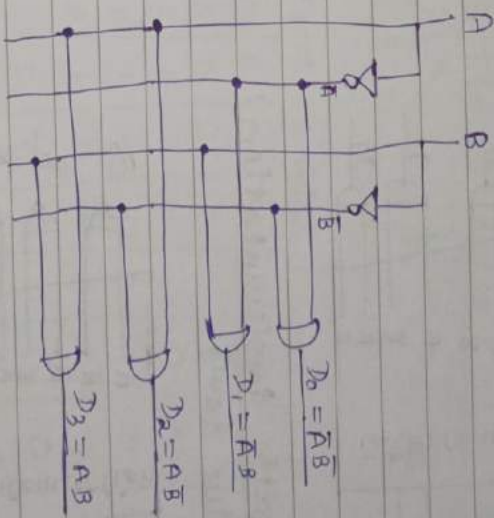
$$D_1 = \bar{A}B$$

$$D_2 = AB$$

$$D_3 = AB$$

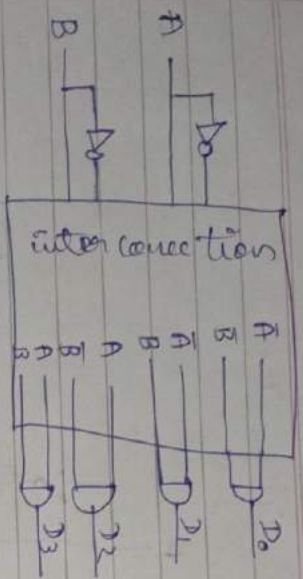
$$\begin{aligned}\bar{A}\bar{B} &\rightarrow 0 \text{ to } 1 \\ \bar{A}B &\rightarrow 1 \\ AB &\rightarrow 1 \\ AB &\rightarrow 1\end{aligned}$$

* Logical circuit:



Here we are ~~using~~ designing decoder logic circuit using AND gate. So it is in active high. Suppose we are designing with NAND gate, then it is active low.

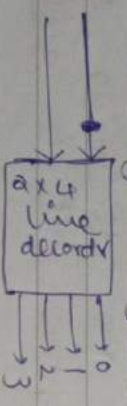
* AND gate implementation:
(active high)



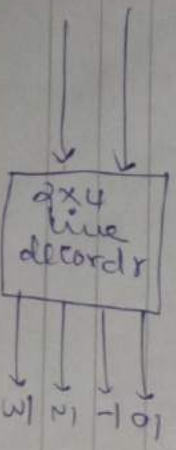
* NAND gate Implementation:
(active low)



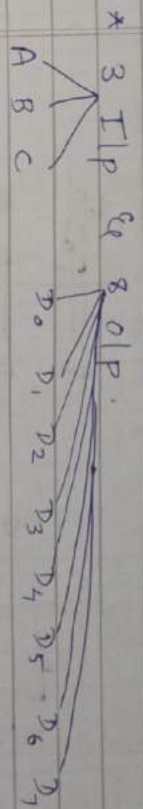
* AND logic symbol:



* NAND logic symbol:



c) 3x8 line decoder:



* Truth Table:

| 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 |

| 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 |

Key

eg →

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

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

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

$$D_3 = \bar{A}BC$$

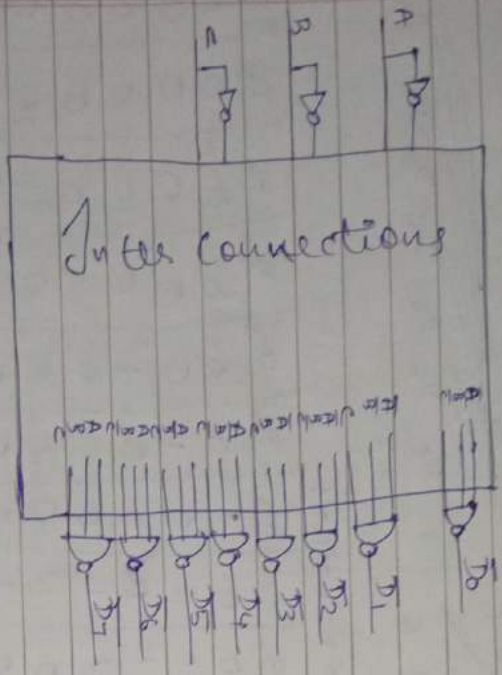
$$D_4 = A\bar{B}\bar{C}$$

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

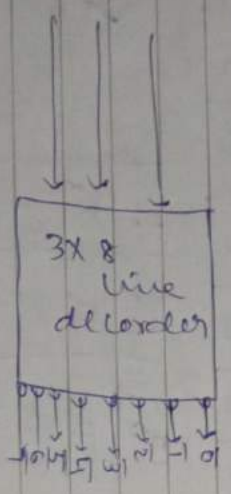
$$D_6 = AB\bar{C}$$

$$D_7 = ABC$$

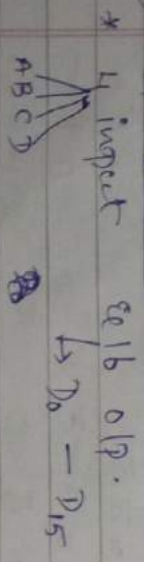
Here we are implementing Active low 3x8 Decoder using NAND gate



* Logic Symbol:



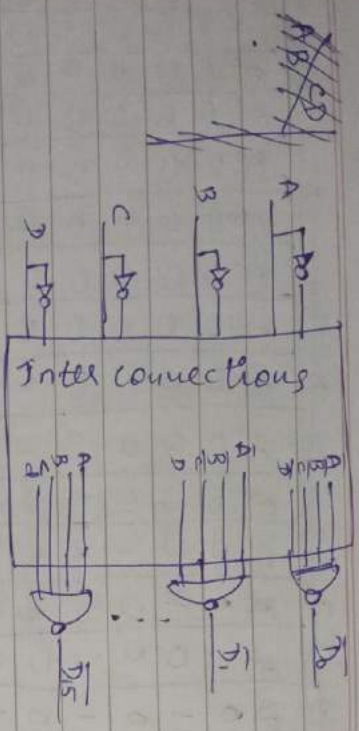
d) 4x16 line decoder:



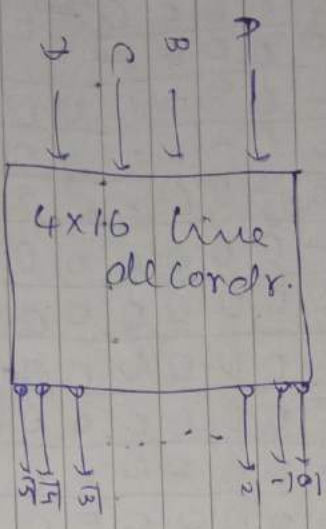
| A | B | C | D | D ₀ | D ₁ | D ₂ | D ₃ | D ₄ | D ₅ | D ₆ | D ₇ | D ₈ | D ₉ | D ₁₀ | D ₁₁ | D ₁₂ | D ₁₃ | D ₁₄ | D ₁₅ |
|---|---|---|---|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

eg →

| | | |
|--|---|--|
| $D_0 = \overline{A}\overline{B}\overline{C}\overline{D}$ | $D_6 = \overline{A}\overline{B}C\overline{D}$ | $D_{12} = A\overline{B}\overline{C}\overline{D}$ |
| $D_1 = \overline{A}\overline{B}C\overline{D}$ | $D_7 = \overline{A}B\overline{C}\overline{D}$ | $D_{13} = A\overline{B}C\overline{D}$ |
| $D_2 = \overline{A}B\overline{C}\overline{D}$ | $D_8 = A\overline{B}C\overline{D}$ | $D_{14} = A\overline{B}C\overline{D}$ |
| $D_3 = \overline{A}B\overline{C}D$ | $D_9 = A\overline{B}CD$ | $D_{15} = A\overline{B}CD$ |
| $D_4 = A\overline{B}\overline{C}\overline{D}$ | $D_{10} = A\overline{B}CD$ | |
| $D_5 = A\overline{B}C\overline{D}$ | $D_{11} = A\overline{B}CD$ | |



* Logic circuit:

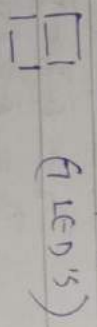


→ BCD to 7 segment decoder:

BCD → 0-9 (decimal)

eg → calculator (0-9)

In cal, 8 is displayed as



* When a decimal no. is representing in a digital screen it needs 7

Segments / 7 LED's.

* Here we are using the input as BCD

if we get a particular output

(def) * It is combinational logic circuit which

has input binary code usually BCD

& convert the input into 7 segment

output LED's.

Decimal no = 0 to 9.

$\begin{matrix} a \\ f \\ b \\ e \\ c \end{matrix}$ (this order is imp)

Here, each 1 is a LED (a, b, c, d, e, f, g)

So there are 7 LED's. 80

that's it is → 7 segment decoders.

| | | | | | | | | | |
|--|---|--|--|--|--|--|--|--|--|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| $\begin{matrix} a \\ f \\ b \\ e \\ c \\ d \end{matrix}$ | $\begin{matrix} a \\ b \\ c \end{matrix}$ | $\begin{matrix} a \\ b \\ c \\ d \end{matrix}$ | $\begin{matrix} a \\ b \\ c \\ d \end{matrix}$ | $\begin{matrix} a \\ b \\ c \\ d \end{matrix}$ | $\begin{matrix} a \\ b \\ c \\ d \end{matrix}$ | $\begin{matrix} a \\ b \\ c \\ d \end{matrix}$ | $\begin{matrix} a \\ b \\ c \\ d \end{matrix}$ | $\begin{matrix} a \\ b \\ c \\ d \end{matrix}$ | $\begin{matrix} a \\ b \\ c \\ d \end{matrix}$ |

No. of inputs = 9

BCD code upto 9 will be the output.

The relation ship b/w I/P & O/P:

| Decimal | 8 | 4 | 2 | 1 | a | b | c | d | e | f | g |
|---------|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 4 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

* Look on-wise $a \rightarrow \sum m(0, 2, 3, 5, 6, 7, 8, 9)$

(a-norm (el-conv) 122222 6m1222, other) avoid 222222

- * $b = \sum m(0, 1, 2, 3, 4, 7, 8, 9)$
- * $c = \sum m(0, 1, 3, 4, 5, 6, 7, 8, 9)$
- * $d = \sum m(0, 2, 3, 5, 6, 8, 9)$
- * $e = \sum m(0, 2, 6, 8)$
- * $f = \sum m(0, 4, 5, 6, 8, 9)$
- * $g = \sum m(2, 3, 4, 5, 6, 8, 9)$

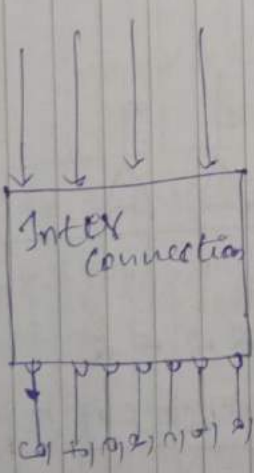
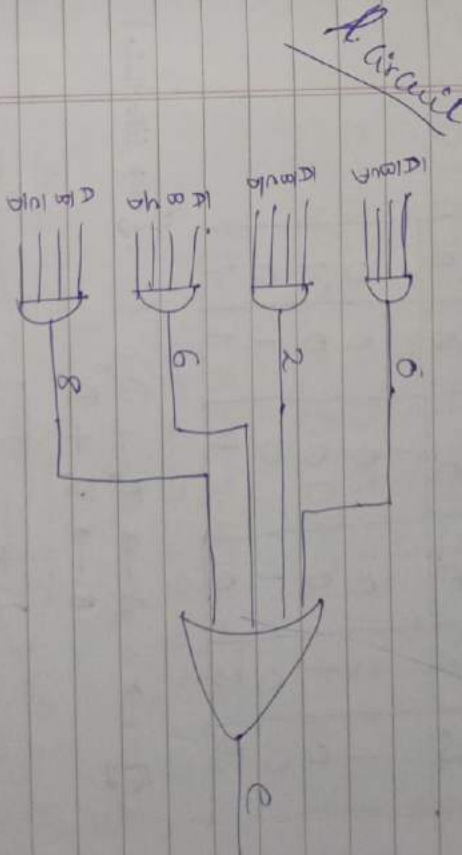
a to g Again 222222 222222 are take e only. for drawing logical circuit

DE $\rightarrow 2^N \rightarrow 2^N$
EN $\rightarrow 2^N \rightarrow 2^N$

$e = \sum m(0, 2, 6, 8)$

$0 \rightarrow \bar{a}\bar{b}\bar{c}\bar{d}$
 $2 \rightarrow \bar{a}\bar{b}c\bar{d}$
 $6 \rightarrow \bar{a}b\bar{c}\bar{d}$
 $8 \rightarrow a\bar{b}\bar{c}\bar{d}$

$e \rightarrow \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}c\bar{d} + \bar{a}b\bar{c}\bar{d} + a\bar{b}\bar{c}\bar{d}$



\rightarrow Encoders:

DE \rightarrow 2x4
 EN \rightarrow 4x2

classmate
 Date _____
 Page _____

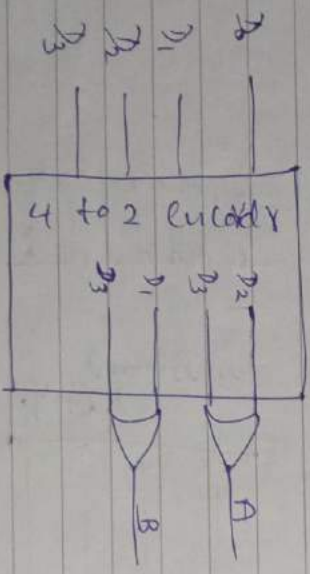
I 4 to 2 Encoder:
 4 I/P & 2 o/p.
 $4 \rightarrow D_0, D_1, D_2, D_3$
 $2 \rightarrow A, B$

* Truth table:

| D_0 | D_1 | D_2 | D_3 | A | B |
|-------|-------|-------|-------|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |

eg \rightarrow $A \rightarrow D_2 + D_3$
 $B \rightarrow D_1 + D_3$

(check like this)



II 8 to 3 Encoder:

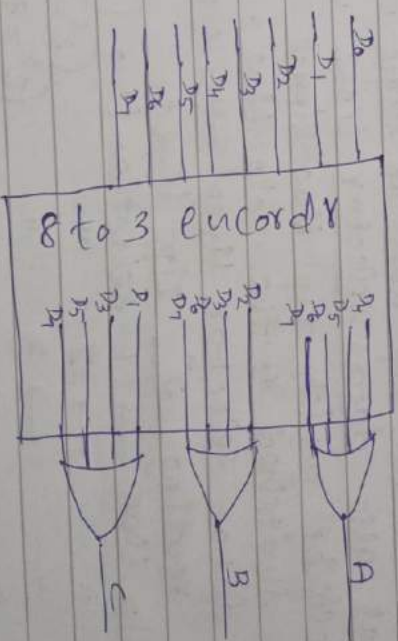
$8 \rightarrow D_0 - D_7$
 $3 \rightarrow A, B, C$

A, B, C \rightarrow 3 rows

Date _____
 Page _____

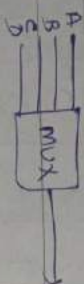
| D_0 | D_1 | D_2 | D_3 | D_4 | D_5 | D_6 | D_7 | A | B | C |
|-------|-------|-------|-------|-------|-------|-------|-------|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

eg \rightarrow $A \rightarrow D_4 + D_5 + D_6 + D_7$
 $B \rightarrow D_2 + D_3 + D_6 + D_7$
 $C \rightarrow D_1 + D_3 + D_5 + D_7$



⇒ Multiplexers & De-multiplexers :

→ Multiplexers (MUX) :



$n \rightarrow 1$ [n no. of I/p \rightarrow 1 o/p]
we get only 1 o/p

[n input pass independently
as per I/p system & control
Select address
selector \rightarrow MUX \rightarrow o/p
we get certain o/p.]

* MUX is a device that allows digital

signals from several sources to be

routed onto a single line of o/p.

* Has several I/p lines & single o/p line.

* Has also data selector lines which

specifies which I/p ~~the~~ signal has to

be switched to the o/p line.

* n data selector lines are required

for routing 2^n I/p lines.

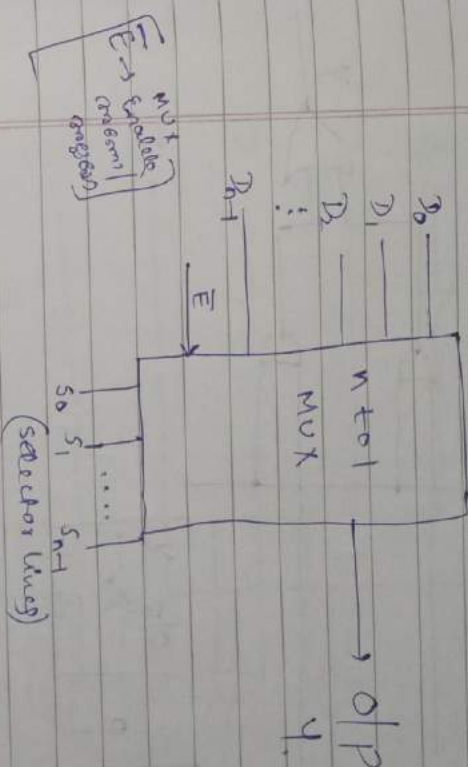
• 2 I/p lines \rightarrow 1 selection line ($2^0 = 2$)

• 4 " " \rightarrow 2 " ($2^2 = 4$)

• 8 " " \rightarrow 3 " ($2^3 = 8$)

• 16 " " \rightarrow 4 " ($2^4 = 16$)

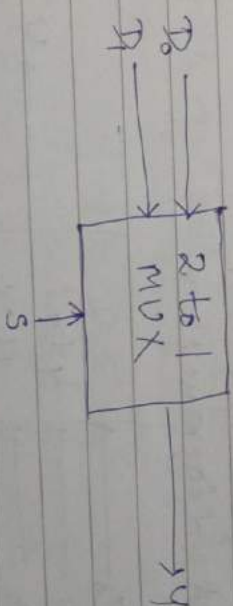
* n to 1 MUX Block diagram :

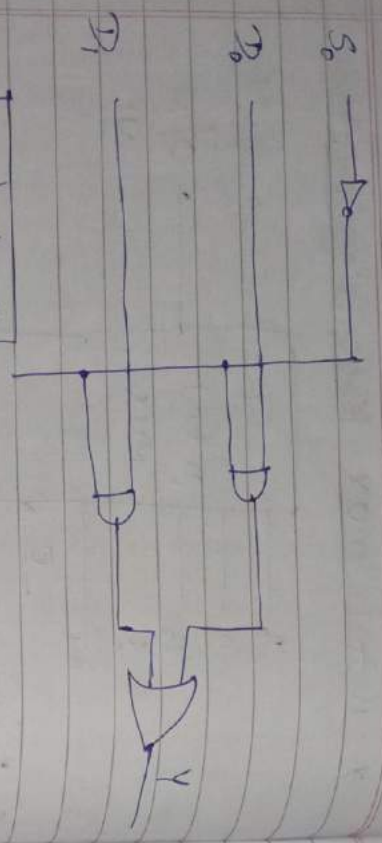


1. 2 to 1 MUX :

* 2 I/p data line and 1 o/p line. &

1 Block diagram :





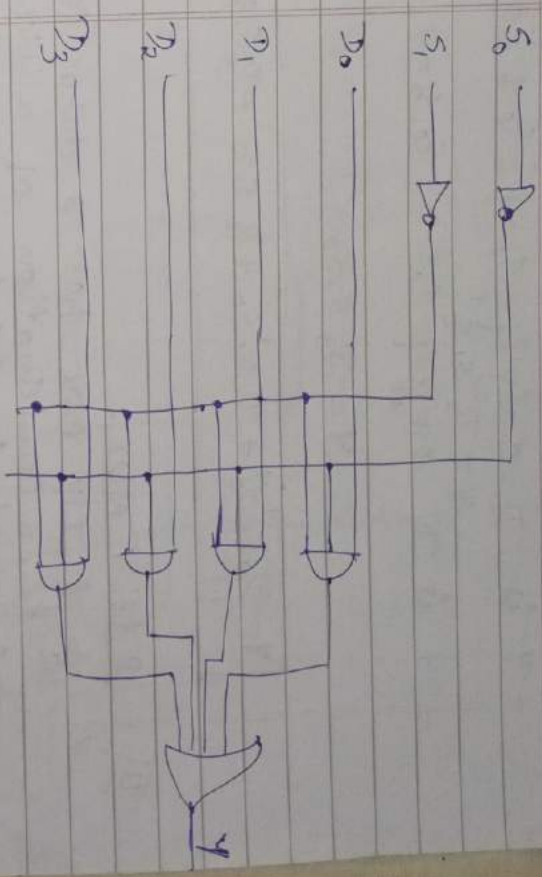
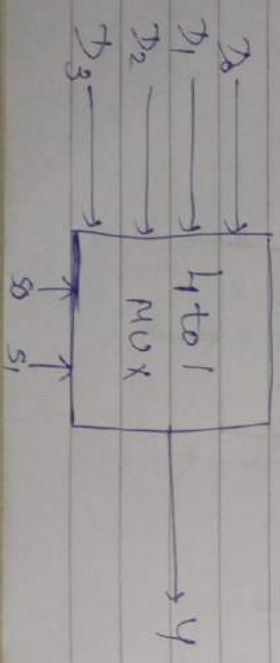
| S | 0 | 1 |
|---|----------------|----------------|
| 0 | D ₀ | D ₁ |
| 1 | D ₁ | D ₀ |

$\therefore Y = D_0 S' + D_1 S$

II. 4 to 1 MUX :

* A 4 to 1 MUX has 2 data selector lines ($2^2=4$), a combination of which are used to select any of the 4 data IP lines.

* Block diagram:



| S ₁ | S ₀ | Y |
|----------------|----------------|----------------|
| 0 | 0 | D ₀ |
| 0 | 1 | D ₁ |
| 1 | 0 | D ₂ |
| 1 | 1 | D ₃ |

- Y is D₀ only if S₁=0 & S₀=0.

$\therefore Y = D_0 S_1' S_0'$

- Y is D₁ only if S₁=0 & S₀=1

$Y = D_1 S_1' S_0$

- y is D_2 only if $s_1 = 1$ & $s_0 = 0$.

$$y = D_2 s_1 s_0'$$

- y is D_3 only if $s_1 = 1$ & $s_0 = 1$

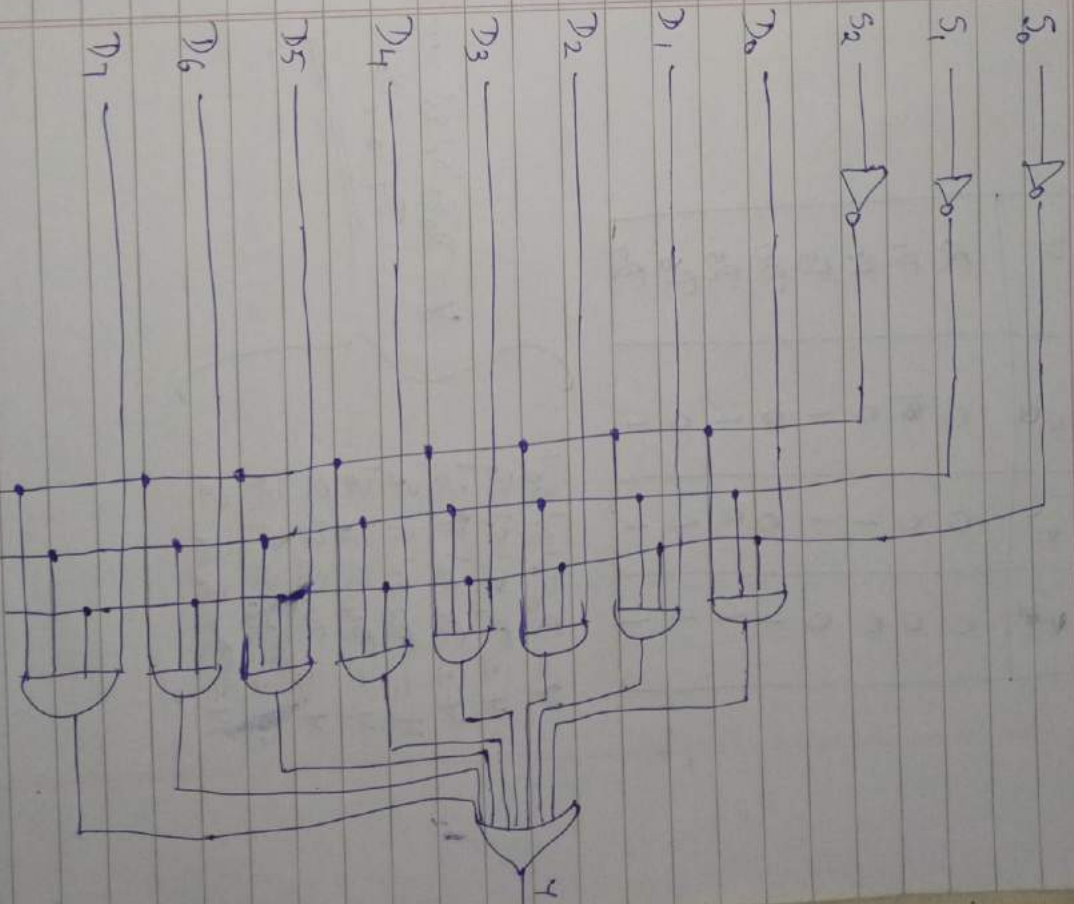
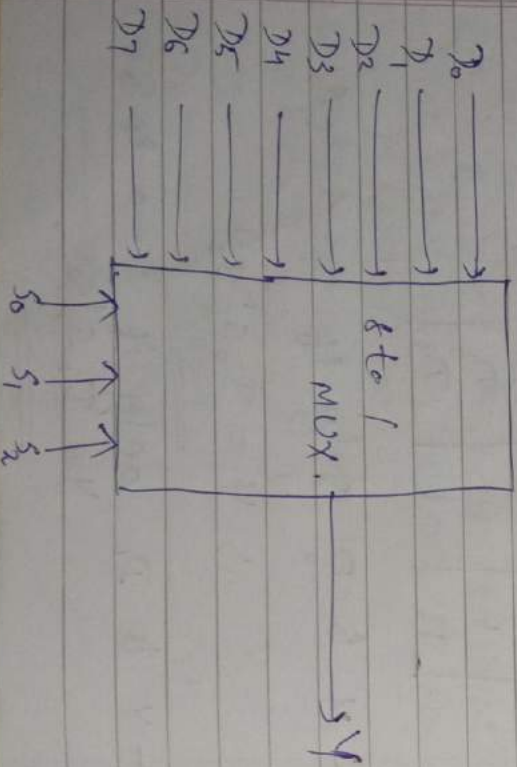
$$\therefore y = D_3 s_1 s_0$$

$$y = D_0 s_1' s_0' + D_1 s_1' s_0 + D_2 s_1 s_0' + D_3 s_1 s_0$$

III 8 to 1 MUX:

* A 8 1P MUX has 3 data selector ($2^3 = 8$), a combination of which are used to select any of 8 data lines

* Block diagram:



| 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 |

$$y = D_0 s_2' s_1' s_0'$$

$$y = D_1 s_2' s_1' s_0$$

$$y = D_2 s_2' s_1 s_0'$$

$$y = D_3 s_2' s_1 s_0$$

$$y = D_4 s_2 s_1' s_0'$$

$$y = D_5 s_2 s_1' s_0$$

$$y = D_6 s_2 s_1 s_0'$$

$$y = D_7 s_2 s_1 s_0$$

$$y = D_0 s_2' s_1' s_0' \dots +$$

$$D_7 s_2 s_1 s_0$$