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 op^r 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^r.

High → 1
Low → 0

→ **Negative logic:**

High → 0
Low → 1

A	B	y = A op^r B
1	1	1
1	0	1
0	1	1
0	0	0

→ OR op^r.

* +ve logic → AND opr.
* -ve logic → OR opr.

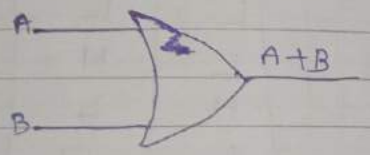
* we always uses +ve logic for developing digital circuit.

→ Basic logic gates :

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

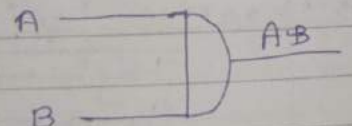
1) OR Gate

A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1



2) AND gate.

A	B	Y=A·B
0	0	0
0	1	0
1	0	0
1	1	1



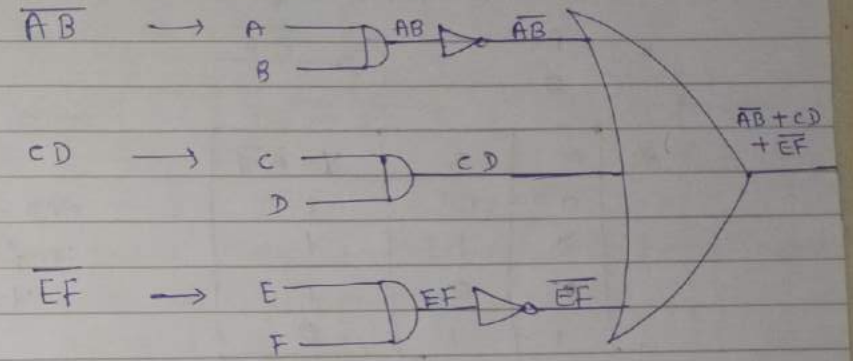
3) NOT gate

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

Q

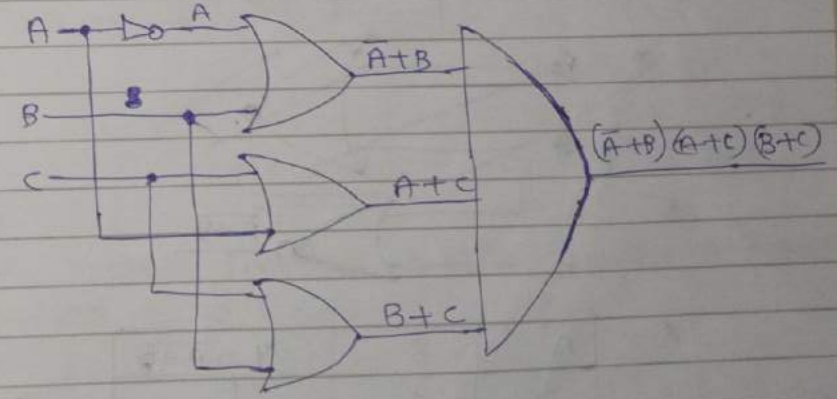
1) $F = \overline{AB} + CD + \overline{EF}$

A) 6 inputs → A, B, C, D, E, F



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

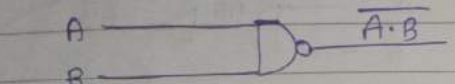
A)



* Universal Gates :

1) NAND :

* Combination of AND gate NOT gate.

* 

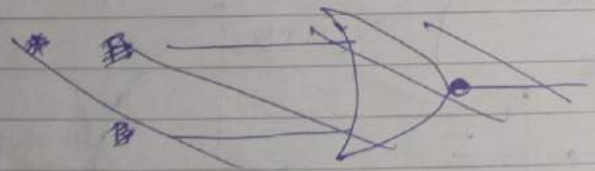
*

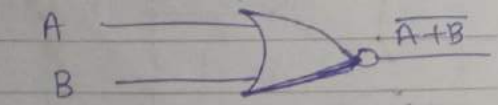
A	B	$Y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

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

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

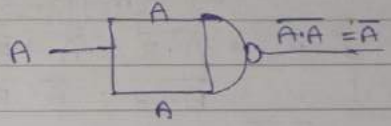
or - ork Compliment.
$Y = \overline{A+B}$

NOT gate has only 1 input

→ Universal Gates properties : (NAND).

— ① * NOT gate implimentation with NAND gate:

* NOT → 1 input
  NAND → 2 input

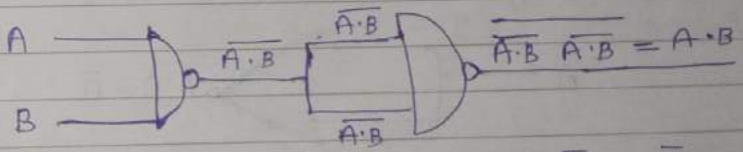
* 

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

$\overline{A} \rightarrow$ NOT

— ② * AND gate implementation with NAND gate:

* AND → 2 input
  NAND → 2 input

* 

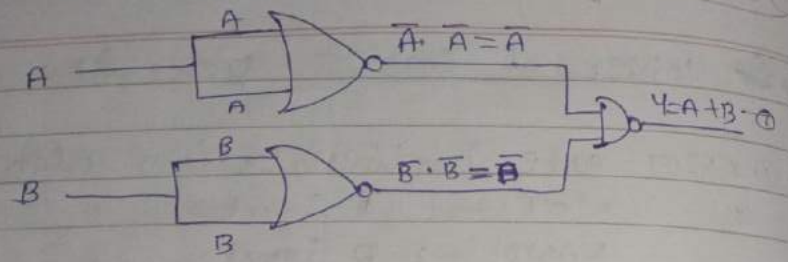
$\overline{A} \cdot \overline{A} = \overline{A}$
$\overline{A \cdot B} \cdot \overline{A \cdot B} = \overline{A \cdot B}$
$\therefore \overline{\overline{A \cdot B}} = A \cdot B$

$A \cdot B \rightarrow$ AND

— ③ * OR gate Implimentation with NAND gate:

* OR → 2 inputs
  NAND → 2 inputs

(OR) → gate.



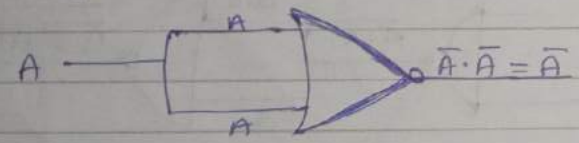
② by Demorgan's Law.

$$\overline{A\cdot B} \rightarrow \overline{A\cdot B} = \bar{A}+\bar{B} \quad ③$$
$$\overline{A+B} = \bar{A}\cdot\bar{B} \quad ④$$

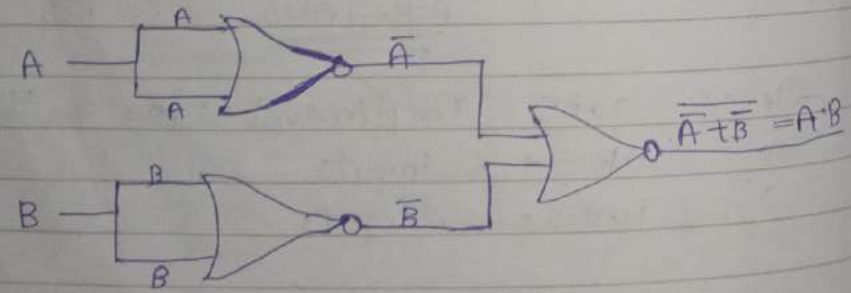
① $\therefore \overline{\bar{A}\cdot\bar{B}} = \overline{\overline{A\cdot B}} = \overline{\bar{A}+\bar{B}} = A+B$

→ Universal gate properties (NOR):

① NOT gate implimentation with NOR Gate

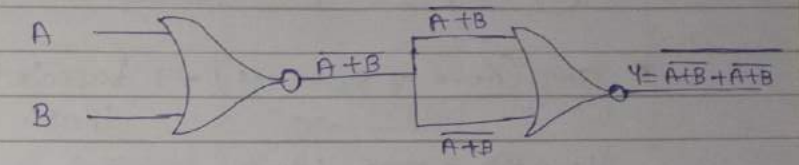


② AND gate implimentation with NOR (OR).



$$\overline{\bar{A}+\bar{B}} = \overline{\overline{A+B}} = \overline{\bar{A}\cdot\bar{B}} = A\cdot B$$
(by D-morgn's)

③ ~~NOT~~ OR (OR) implimentation with NOR (OR):



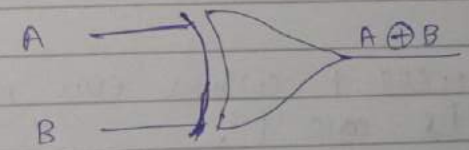
$$\overline{A+B}+\overline{A+B} = \overline{A+B} = A+B \qquad \overline{A+B}\cdot\overline{A+B} = \overline{A+B} = A+B$$
(D-morgn's)

→ Exclusive OR → XOR → '⊕' symbol
Exclusive NOR → XNOR → '⊙' symbol

✳ XOR :

* It produce Rslt = 1 when odd no.. of inputs are 1.

✳



*

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

odd →
0+1 = 1
1 → odd.
→ 1

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

$Y = A \oplus B$ ന്റെ [uncertain] column-ത്തിൽ 1 വരുന്നത് [uncertain] നോക്കുക [uncertain] → then 0-ന് complement ആക്കുക [uncertain] & 1 for same.

eg → $Y = \bar{A}B + A\bar{B}$ (above T.table).

* POS (product of sum) → based on this t. table.

$Y = A \oplus B$ ന്റെ [uncertain] col-ത്തിൽ 0 വരുന്നത് [uncertain] നോക്കുക [uncertain] → then 1-ന് compliment [illegible]

eg → $Y = (A+B)(\bar{A}+\bar{B})$ (above T. table)

XNOR :

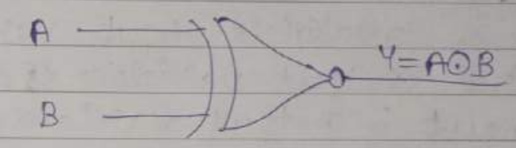
* produces 1 when even no. of inputs are 1.

*

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

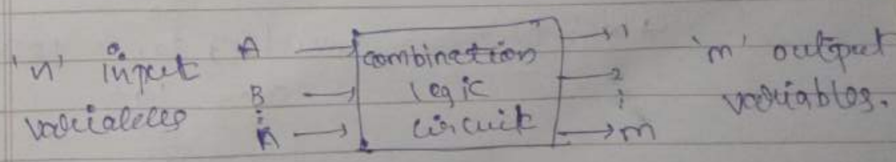
*



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

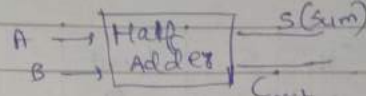
→ combinational logic circuit :

* It consist 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 prsnt, it produces same output.
* There is no memory element to store previous ~~element~~ rslt.
* Step to develop c. l circuit:
a) The prbm is stated.
b) The nom of available input variables & required output variables is determined
c) The input & output variables are assigned letter symbols.
d) The truth table that defines the required relationship b/w input & output is derived.
e) The simplified boolean ()s for each output is obtained.
f) The logic diagram is drawn

1 → Adder : (for adding purpose)
(2 types)
↳ Half (A) → 2 inputs var no carry possible
eg → 10 + 10 = 00
↳ Full (A) → 3 inputs carry possible.
eg → 10 + 10 = 100

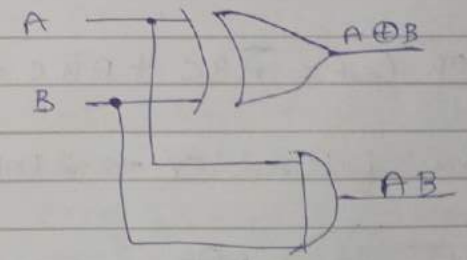
1) Half Adder :
* 2 inputs & 2 outputs
*


*

A	B	S	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

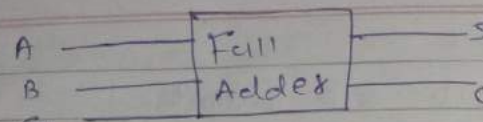
consider → sop $S = \bar{A}B + A\bar{B} = A \oplus B$
sop $C_{out} = AB$

* Half adder circuit:



2) Full Adder :

* used for addition
* 3 inputs & 2 outputs.

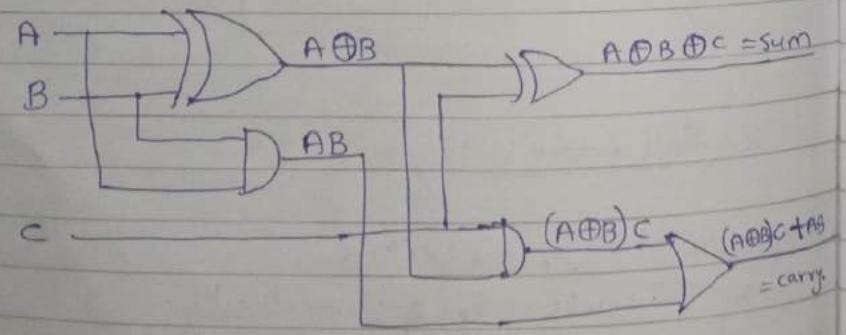


A	B	C_{in}	S	C_{out}	
0	0	0	0	0	S=0 → 0+0+0.
0	0	1	1	0	
0	1	0	1	0	
0	1	1	0	1	1+1 = 01
1	0	0	1	0	
1	0	1	0	1	
1	1	0	0	1	
1	1	1	1	1	1+1 = 01 +1 = 11

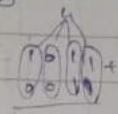
SOP $S = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$ ← $A \oplus B \oplus C$

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

* To draw full Adder → 2 Half adder & 1 OR gate



→ Binary parallel Adder (Ripple carry Adder):



+ ഇത് ഒരു Full Adder ഉപയോഗിച്ച് ചെയ്യാം.
So we need 4 Full Adder.

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 are needed.

C_{in}	0	1	1	0	+
A	1	0	1	1	→ LSB
B	0	0	1	1	
S	1	1	1	0	
C_{out}	0	0	1	1	

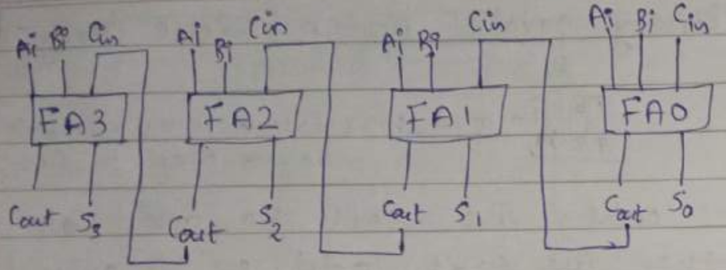
1+ 1 / 10

1+ 1 1 / 11

Here, last significant bit always 0 (carry) there is no addition before.

The carry (C_{out}) will move to C_{in} after each bit addition.

Here, we are using 4 full Adders bcz it has 4 bits.



Sum (S) = 01110
Carry (C_{out}) = 0011

~~Cout [illegible] Cin [illegible] propagation delay.~~

* In this circuit FA1 will be worked only after getting the rslt Cout from FA0 & so on, so there is a delay for this propogation.
The delay is from left to right. (←)
This propogation delay is → carry propagation delay, so that's why it is also → Ripple Carry Adder.

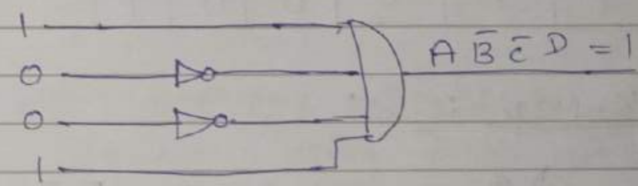
→ Decorder:

* It is combinational logic circuit.
* The basic () of decorders is detects a presnce of some input combinations & indicates specific output levels.
* If there is a n input combination then max m ≤ 2^n output combination.
* eg → 2x4, 3x8, 4x16
(2 input - 4 outpt)

a) Basic Binary Decorder :

eg → [1001 [illegible] combination [illegible] → 1 [illegible] → 0 produce]

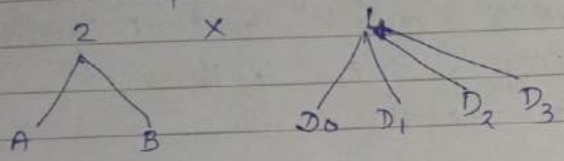
logical circuit:



~~Here are the using AND gate~~
for eg, take the combination 1001 = 1
and for all othr combination it produces 0.
In this eg we are using AND gate
(0110 produces 0)
([illegible] input [illegible] 1 [illegible] AND gate used)

b) 2x4 line decorder :

* 2 inputs & 4 o/T



* 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	$A\bar{B}$ D_2	AB D_3

$\bar{A}\bar{B} \rightarrow D_0 \rightarrow 1$

$\bar{A}B \rightarrow D_1 \rightarrow 1$

$A\bar{B} \rightarrow D_2 \rightarrow 1$

$AB \rightarrow D_3 \rightarrow 1$

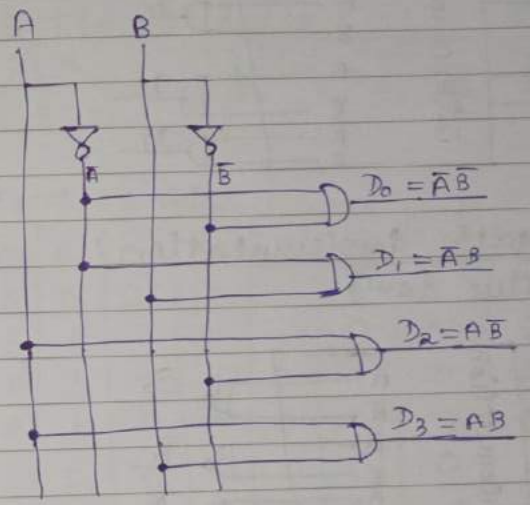
$D_0 = \bar{A}\bar{B}$

$D_1 = \bar{A}B$

$D_2 = A\bar{B}$

$D_3 = AB$

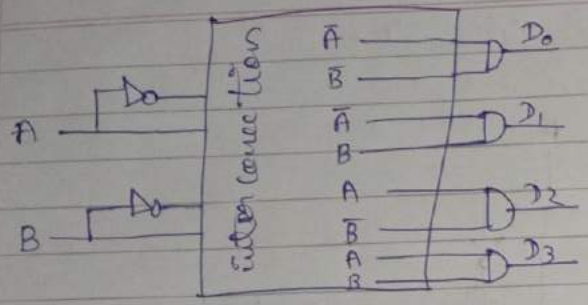
* Logical circuit :



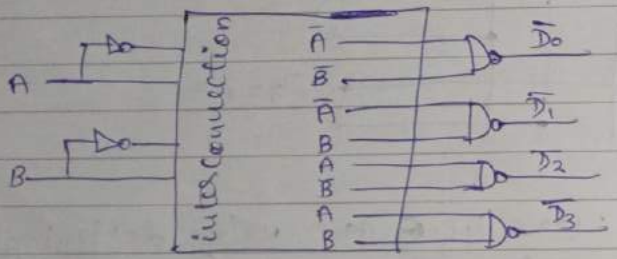
Here we are designing decorder logic circuit using AND gate. So it is in active high.
Suppose we are designing with NAND gate, then it is active low.

* AND gate implementation :

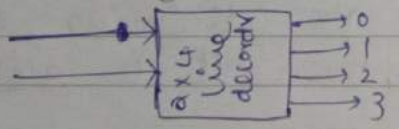
(active high)



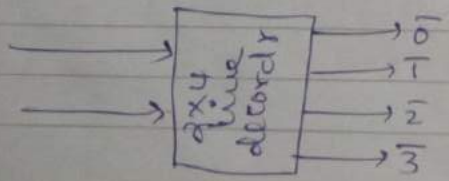
* NAND gate implementation: (Active low)



* AND logic symbol:

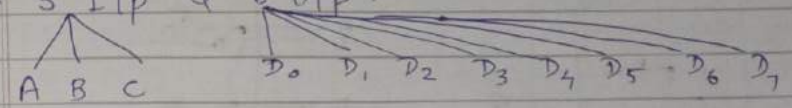


* NAND logic symbol:



c) 3x8 line decorder:

* 3 I/p & 8 O/p.



* 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

* Kmap

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	$B\bar{C}$ 10	BC 11
$\bar{A}$ 0	D_0	D_1	D_2	D_3
A 1	D_4	D_5	D_6	D_7

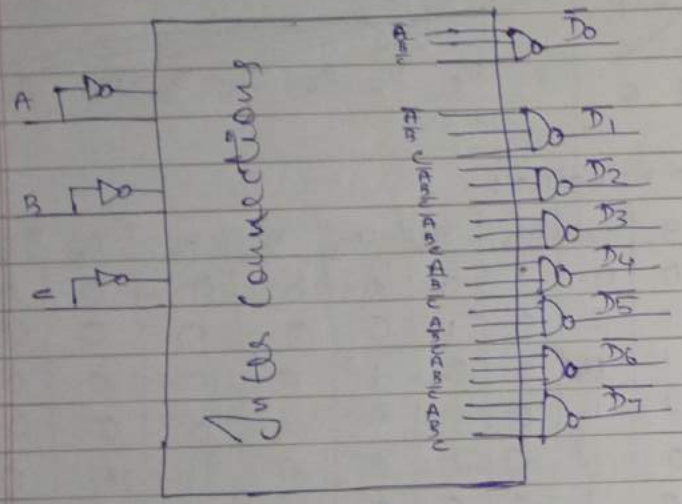
eq→ $D_0 = \bar{A}\bar{B}\bar{C}$ $D_4 = A\bar{B}\bar{C}$

$D_1 = \bar{A}\bar{B}C$ $D_5 = A\bar{B}C$

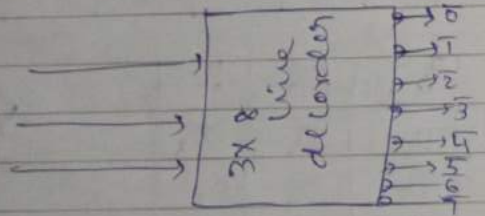
$D_2 = \bar{A}B\bar{C}$ $D_6 = AB\bar{C}$

$D_3 = \bar{A}BC$ $D_7 = ABC$.

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

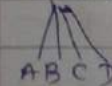


* Logic Symbol:



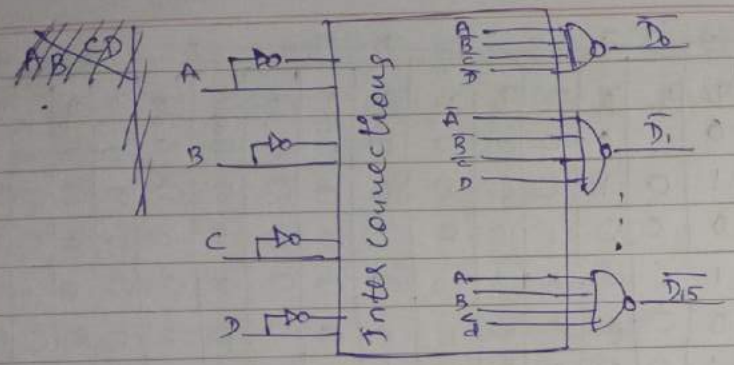
d) 4x16 line decoder:

* 4 input & 16 o/p.
  ↳ $D_0 - D_{15}$

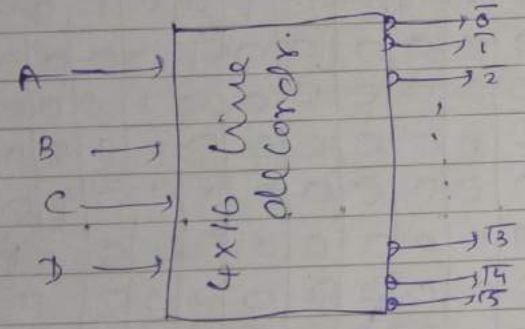


A	B	C	D	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	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

eq → $D_0 = \bar{A}\bar{B}\bar{C}\bar{D}$
$D_1 = \bar{A}\bar{B}\bar{C}D$
$D_2 = \bar{A}\bar{B}C\bar{D}$
$D_3 = \bar{A}\bar{B}CD$
$D_4 = A\bar{B}\bar{C}\bar{D}$
$D_5 = A\bar{B}\bar{C}D$
$D_6 = \bar{A}\bar{B}C\bar{D}$
$D_7 = \bar{A}\bar{B}CD$
$D_8 = A\bar{B}\bar{C}\bar{D}$
$D_9 = A\bar{B}\bar{C}D$
$D_{10} = A\bar{B}C\bar{D}$
$D_{11} = A\bar{B}CD$
$D_{12} = AB\bar{C}\bar{D}$
$D_{13} = AB\bar{C}D$
$D_{14} = ABC\bar{D}$
$D_{15} = ABCD$



* Logic circuit:



→ B.C.D to 7 Segmnt decorder:

B C D → 0 - 9 (decimal)

eg- calculator (0-9)

In cal, 8 is displayed as

(7 LED's)

* when a decimal no. is represnting in a digital screen it ~~xxxxx~~ needs 7 Segments / 7 LED's.

* Here we using the input as BCD & we get a particular o/T

(def) * It is combinational logic circuit which has input binary code usually BCD & convrt the input into 7 Segmnt o/T LED'S.

Decimal no = 0 to 9.

(this order is imp)

Here, each ▯ is a LED (a, b, c, d, e, f, g)
So there are 7 LED's. So that's it is → 7 Segmnt decorders.

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

No. of inputs = 9

BCD code upto 9 will be the o/T.

The relationship b/e I/p & o/T:

Decimal	A (8)	B (4)	C (2)	D (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	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	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	1	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	0	1	1

* look col-wise ← $a \rightarrow \Sigma m(0, 2, 3, 5, 6, 7, 8, 9)$

(a-[illegible] col-[illegible] 1 [illegible], others avoid [illegible] non-[illegible])

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

[a to g [illegible] take e only. for drawing logical circuit]

De → 2^n → 2^n

En → 2^n → n.

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

$0 \rightarrow \bar{A}\bar{B}\bar{C}\bar{D}$.

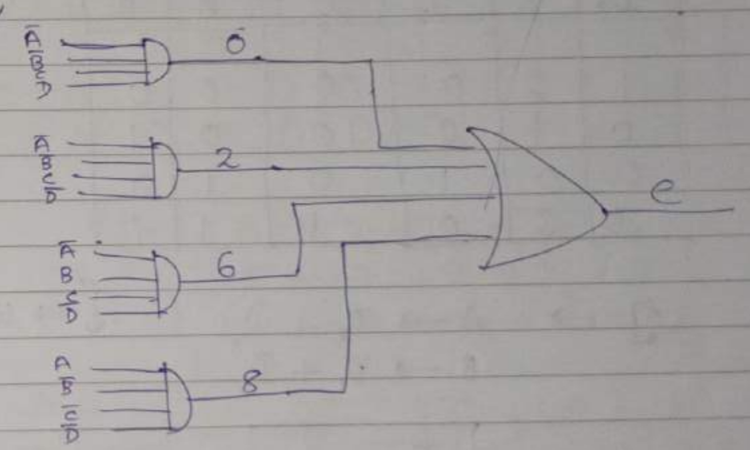
$2 \rightarrow \bar{A}\bar{B}C\bar{D}$

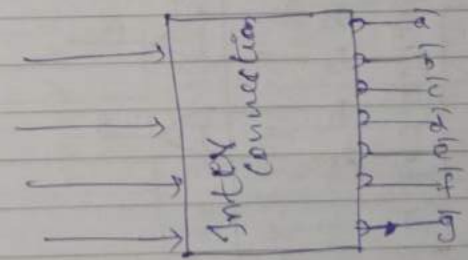
$6 \rightarrow \bar{A}BC\bar{D}$

$8 \rightarrow A\bar{B}\bar{C}\bar{D}$

$e \rightarrow \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + \bar{A}BC\bar{D} + A\bar{B}\bar{C}\bar{D}$

L circuit





→ Encoders :

De → 2 × 4 → opp
En → 4 × 2 ←

I 4 to 2 Encoder :

* 4 I/p & 2 o/p.

4 → D_0, D_1, D_2, D_3
2 → A, B.

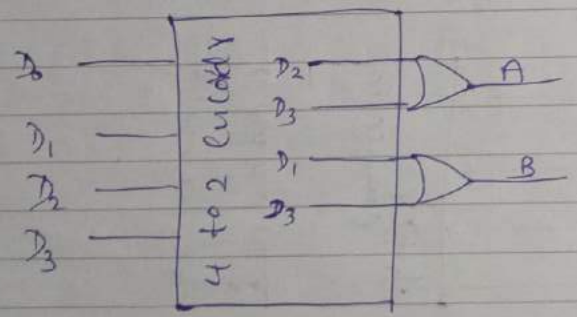
* Truth table :

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

(col. check like this)

eq → $A \rightarrow D_2 + D_3$

$B \rightarrow D_1 + D_3$



II 8 to 3 Encoder :

8 → $D_0 - D_7$

3 → A, B, C

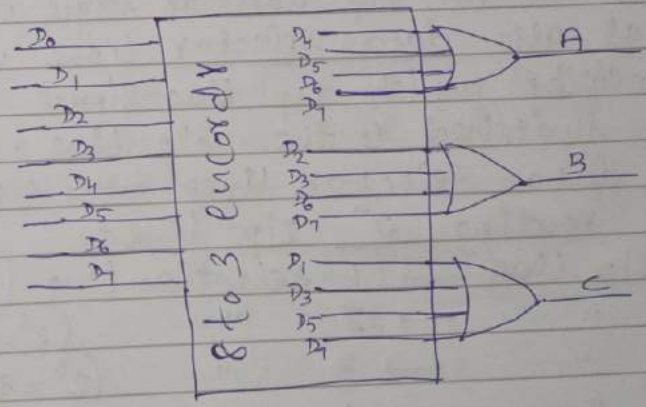
A, B, C → 2^3 → 8 rows

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

eq → $A \rightarrow D_4 + D_5 + D_6 + D_7$

$B \rightarrow D_2 + D_3 + D_6 + D_7$

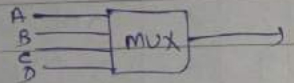
$C \rightarrow D_1 + D_3 + D_5 + D_7$



⇒ Multiplexers & De-multiplexers :

→ Multiplexers (MUX) :

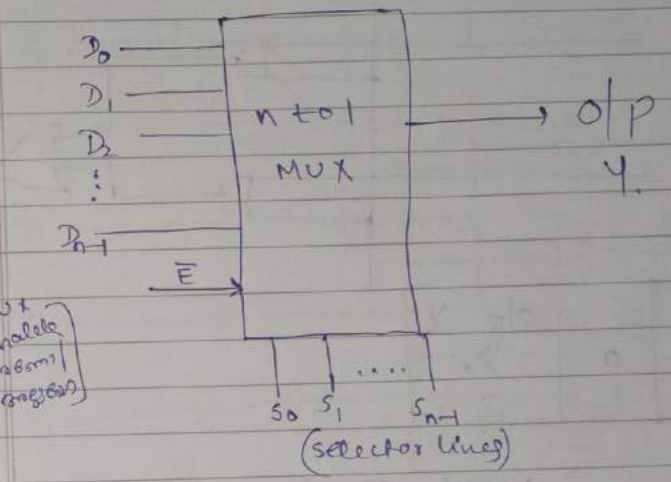
n ⟶ 1 [n' nos of I/p ಇದ್ದಾಗ we get only 1 o/p]



[n' input pins ಇರುವುದರಿಂದ ಅದರ I/p ಲೈನ್ಸ್ ಕೊಡುವುದಕ್ಕೆ ಮತ್ತು Select ಮಾಡುವುದಕ್ಕೆ selector ಪಿನ್ MUX ಗೆ ಕೊಡಲು we get only o/p.]

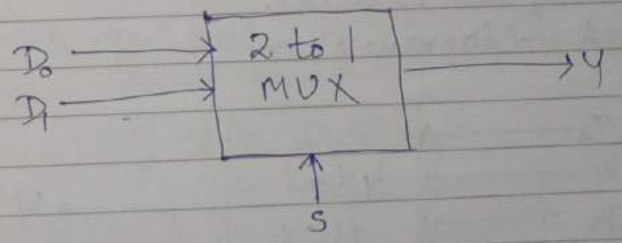
* MUX is a device that allows digital signals from several sources to be routed onto a single line of o/p.
* Has several I/p lines & single o/p line.
* Has also data selector lines which specifies which I/p ~~line~~ signal has to be switched to the o/p line.
* 'n' data selector lines are required for routing 2^n I/p lines.
  - 2 I/p lines ⟶ 1 selection line ($2^1 = 2$)
  - 4 " ⟶ 2 " ($2^2 = 4$)
  - 8 " ⟶ 3 " ($2^3 = 8$)
  - 16 " ⟶ 4 " ($2^4 = 16$)

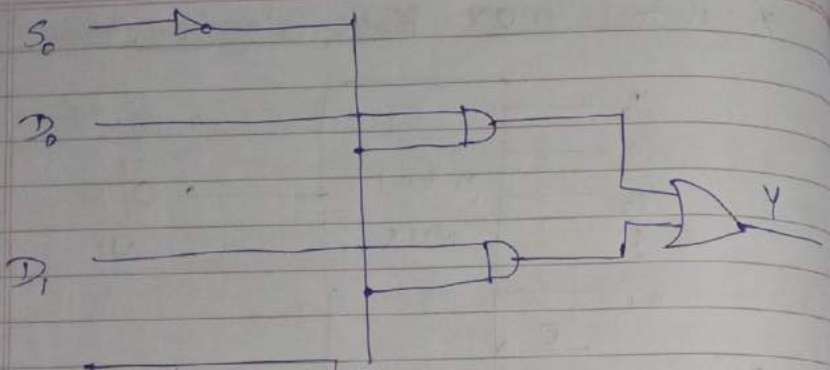
* n to 1 MUX Block diagram :



I 2 to 1 MUX :

* 2 I/p data line and 1 o/p line. & 1 selection line (2^1)
* Block diagram :



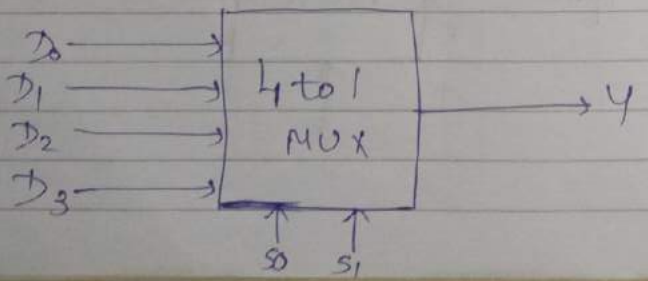


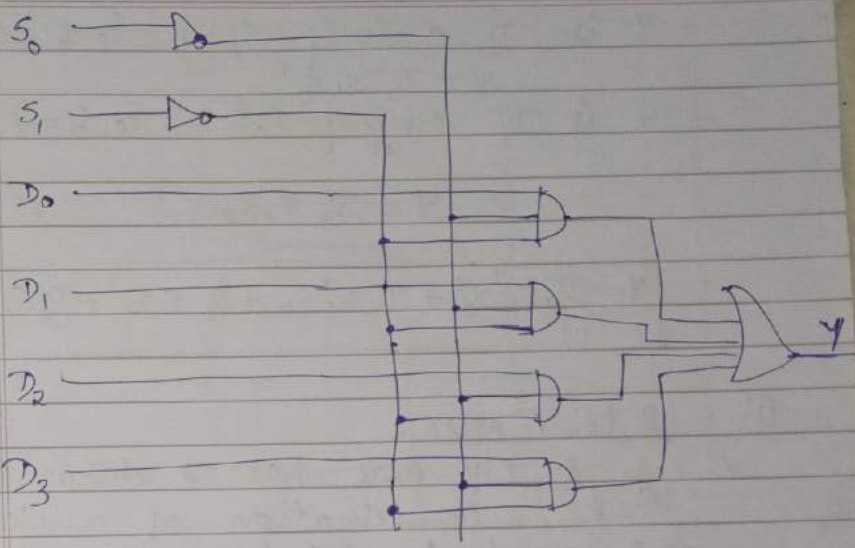
S	O/p Y
0	D_0
1	D_1

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

II 4 to 1 MUX :

* A 4 I/p MUX has 2 data selector line ($2^2 = 4$), a combination of which are used to select any of the 4 data I/p 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 D_0 only if $S_1 = 0$ & $S_0 = 0$.

$\therefore Y = D_0 S_1' S_0'$

- Y is D_1 only if $S_1 = 0$ & $S_0 = 1$

$Y = D_1 S_1' S_0$

- Y is D_2 only if $S_1 = 1$ & $S_0 = 0$.

$$Y = D_2 S_1 S_0'$$

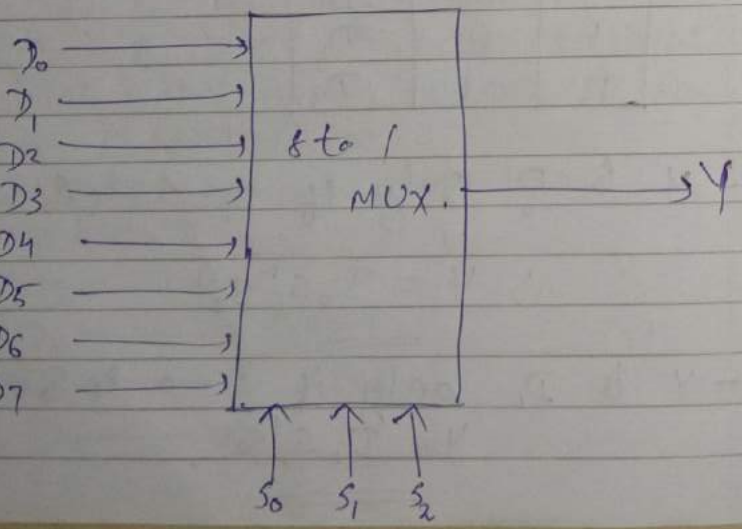
- Y is D_3 only if $S_1 = 1$ & $S_0 = 1$

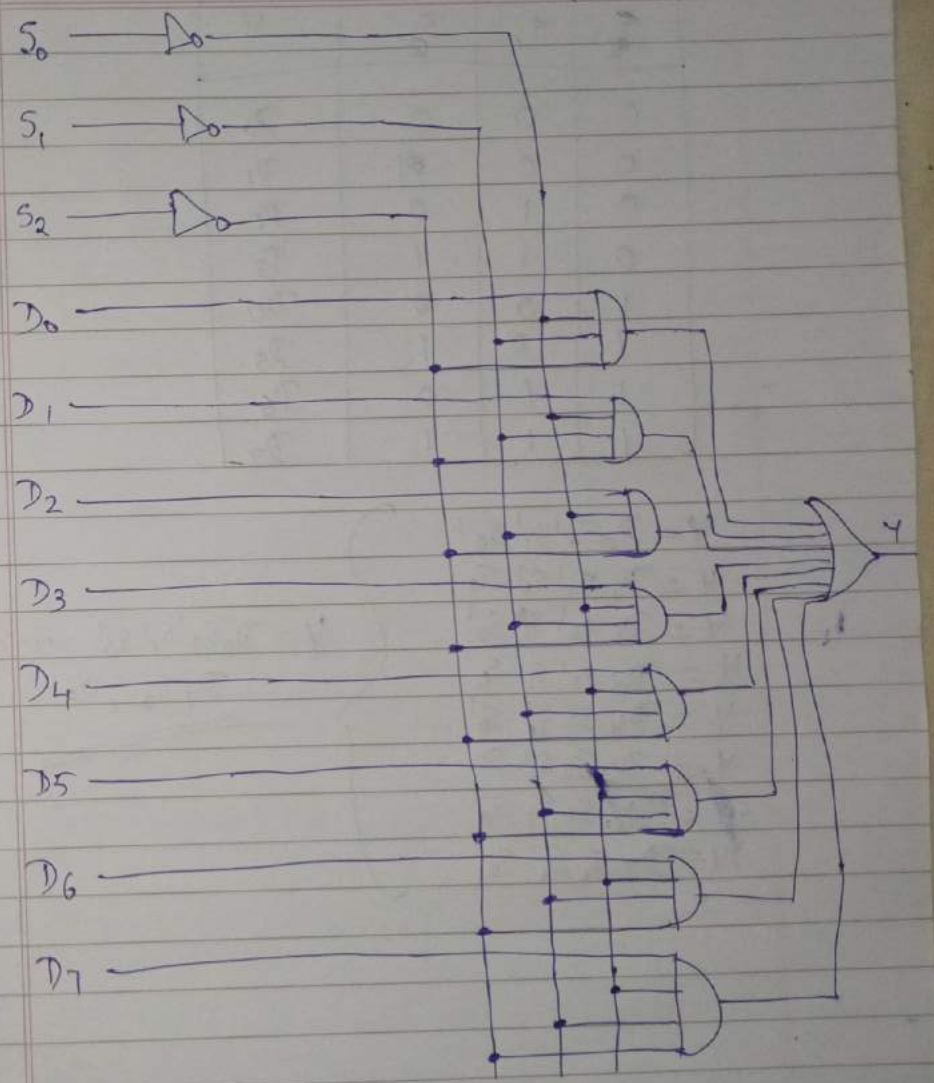
$$\therefore Y = D_3 S_1 S_0$$

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

III **8 to 1 MUX:**

* A 8 I/p 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

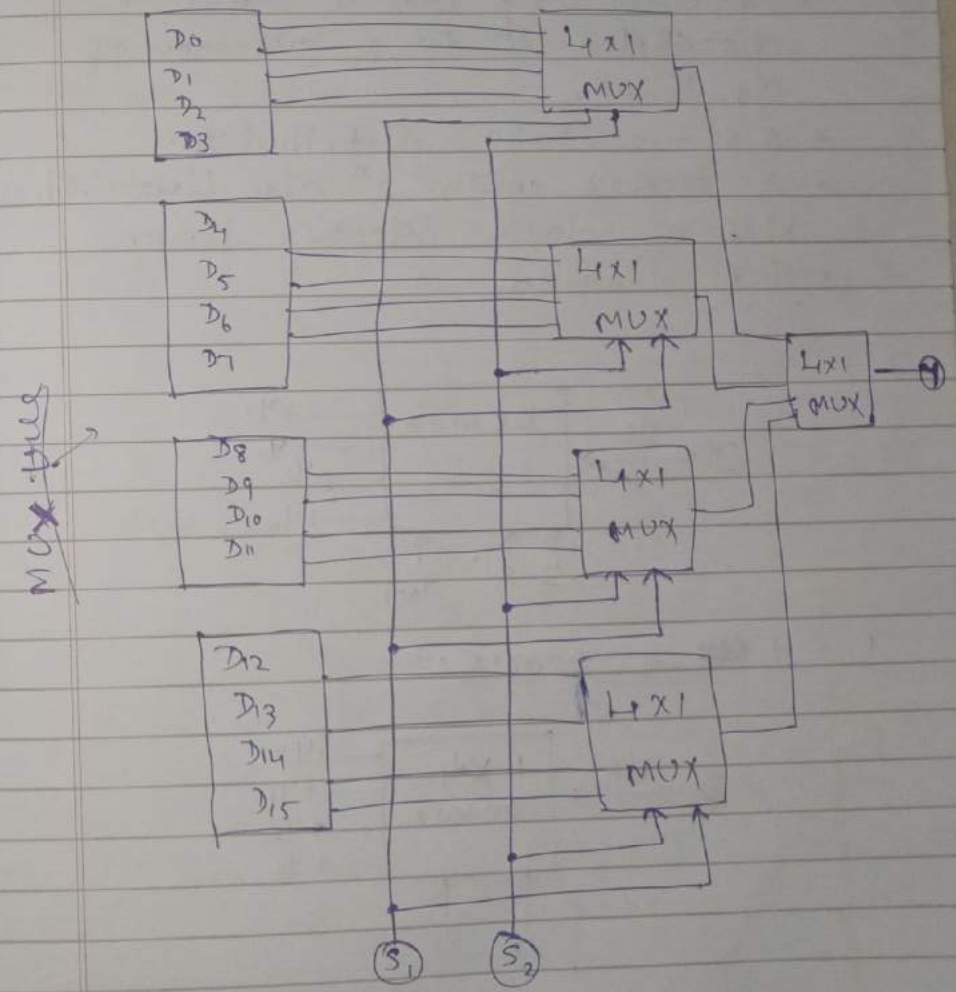
$Y = D_0 S_2' S_1' S_0'$
$Y = D_1 S_2' S_1' S_0$
$Y = D_2 S_2' S_1 S_0'$
$Y = D_3 S_2' S_1 S_0$
$Y = D_4 S_2 S_1' S_0'$
$Y = D_5 S_2 S_1' S_0$
$Y = D_6 S_2 S_1 S_0'$
$Y = D_7 S_2 S_1 S_0$

$Y = D_0 S_2' S_1' S_0' + \dots + D_7 S_2 S_1 S_0$

→ MUX tree :

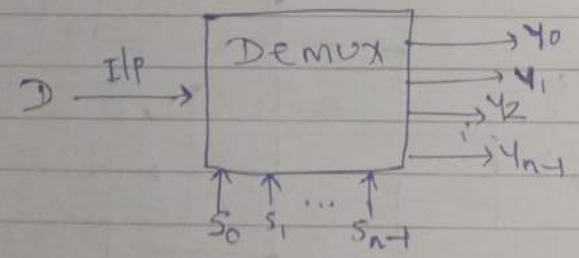
* Bigger MUX obtained by combining smaller MUX.
* MUX with more no of I/p can be obtained cascading 2 / more MUX with less no of I/p.

eg → for 16x1 MUX, we need 4 MUX.

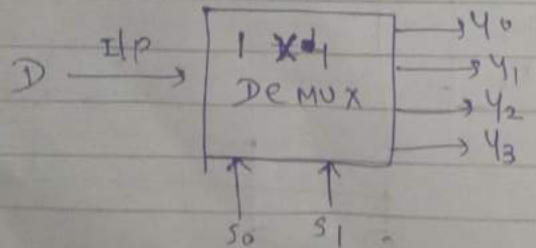


→ DEMUX :

* 1 I/p line & 'n' nos of o/p.
* A DEMUX performs the reverse of a MUX.
* It takes I/p from 1 line & distributes it to a given nos of o/p lines.
* Also → Data distributer.
* A DEMUX with 2^n o/p line should have 'n' data Selector line.
* Blog diagram →



1 1 X 4 DEMUX :

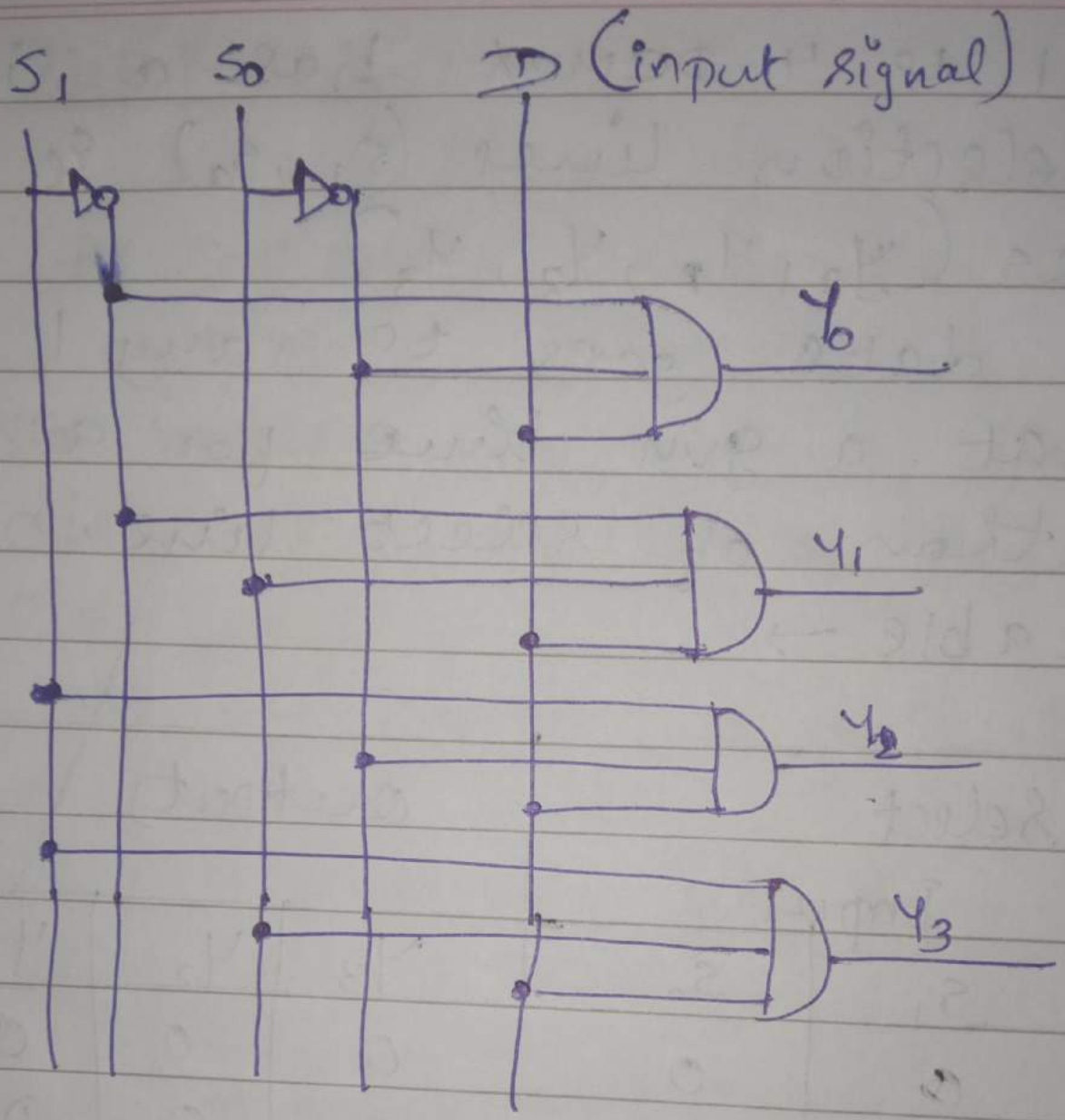


* A 1 to 4 DEMUX has a single I/p (D), 2 selection lines (S_1, S_0) & 4 o/p lines (Y_0, Y_1, Y_2, Y_3)
* The I/p data goes to any 1 of the 4 o/p at a given time for a particular combination of select line.
* Truth table →

Data Input (D)	Select Input		Outputs			
	S_1	S_0	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0

eqs →

$$Y_0 = \overline{S_1}\,\overline{S_0}$$
$$Y_1 = \overline{S_1}\,S_0$$
$$Y_2 = S_1\,\overline{S_0}$$
$$Y_3 = S_1\,S_0$$


S1
S0
D (input signal)
Y0
Y1
Y2
Y3