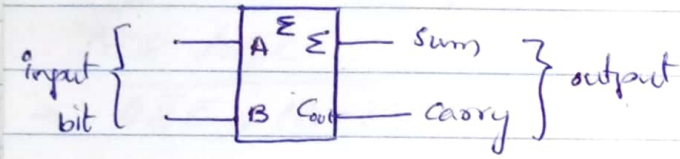
Foundation for Advancement of Science & Technology

CHAPTER # 06

(HALF ADDERS)

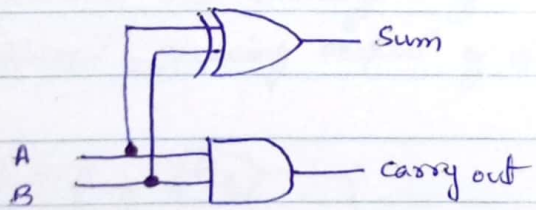
→ It accepts two binary digits on its input and produces two binary digits on its output - a sum bit & a carry bit



A	B	C_{out}	Σ_1
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	0

$C_{out} = AB$

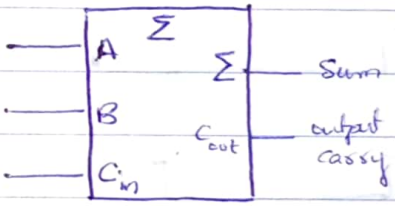
$Sum = A \oplus B$



(FULL ADDERS)

→ The full-adder accepts two input bits and an input carry & generates a sum output and a output carry.

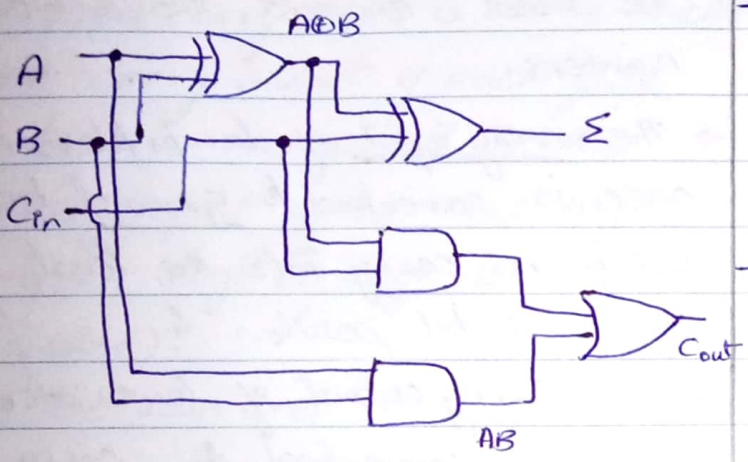
→ The only difference b/w full & half adder is of input carry.

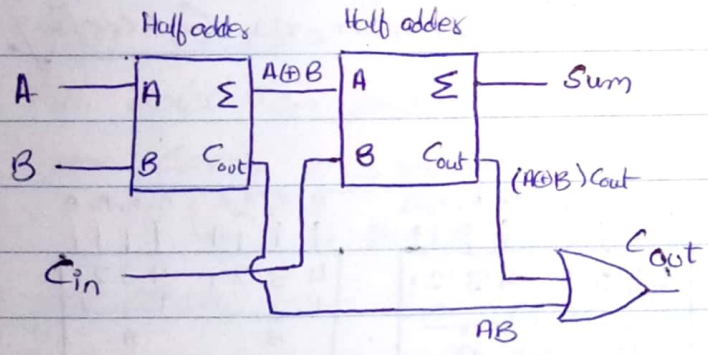


A	B	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

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

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





1
+ 1
10 → Using Half Adder

1
1
10
+1 → Cin
11 → Using Full Adder

PARALLEL BINARY ADDERS

→ Two or more full adders are connected to form paralled binary adders.

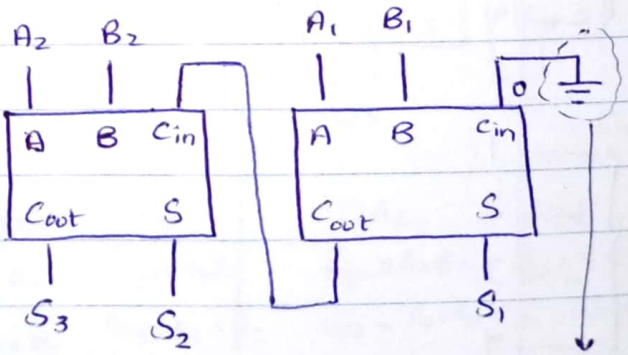
→ To add two binary no. a full adder is required for each bit
- 2-bit no. → 2 Adders (FA)
- 4-bit no. → 4 Adder (FA)

General format, addition of two-bit nos.

$A_2\ A_1$

$B_2\ B_1$

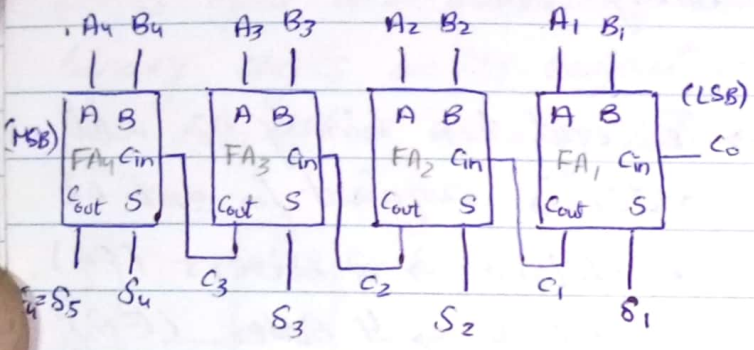
$S_3\ S_2\ S_1$



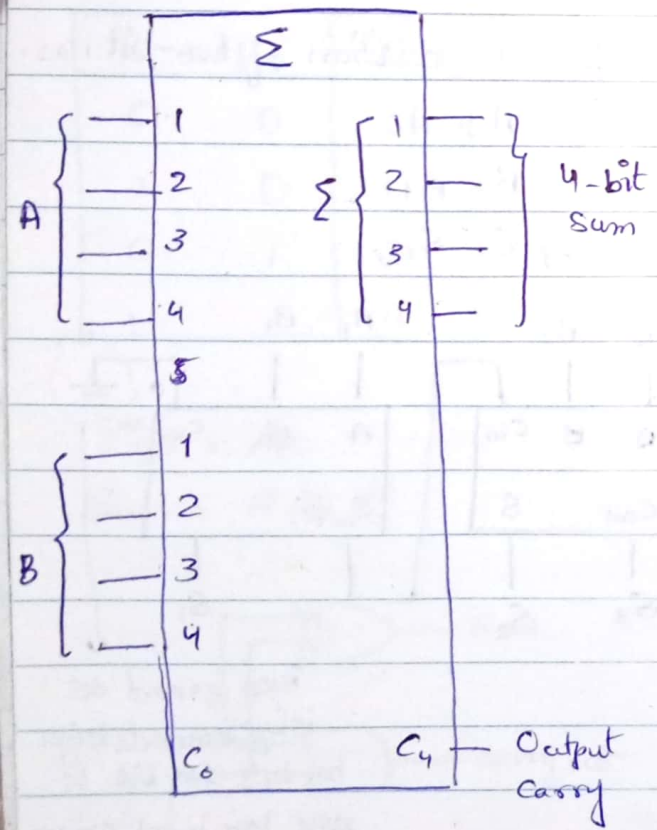
isko ground se isliye connect kiya hai bcz 1st bit ke pass koi input carry nhi hota.

(Four Bit // Adders)

→ A group of 4-bit no is called nibble.



▲ Block diagram.



▲ logic Symbol

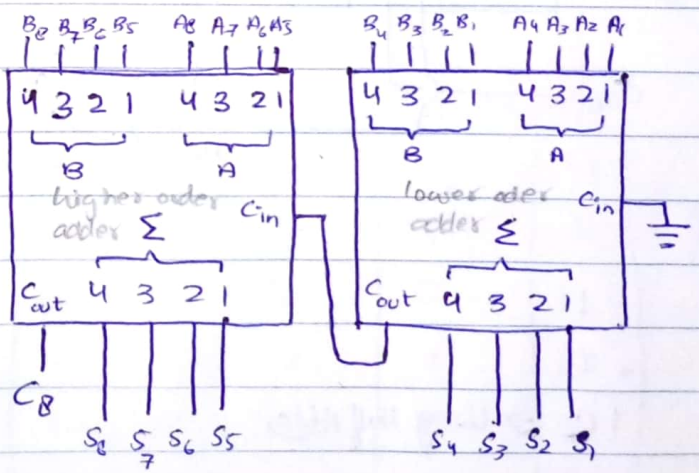
(8-Bit Addition)

→ Two 4-bit parallel adder can be used to add two 8-bit numbers.

→ The carry input of low-order adder is connected to ground bcz there is no carry into the least significant bit position &

→ the carry output of lower order adder is connected to carry input of higher order adder.

→ This is known as Cascading.



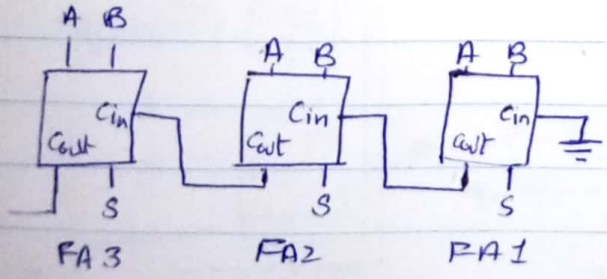
▲ Cascading of two 4-bit adders.

(RIPPLE CARRY ADDER)

A ripple carry adder is one in which the carry output of each full adder is connected to the carry input of the next high-order stage (a stage is one full adder).

→ The sum and Cout of any stage cannot be produced until the input carry occurs; this causes a time delay in the addition process called as Carry propagation delay.

↳ it is the ~~delay for each full adder~~ time from the application of the input carry until the Cout occurs.



Jab tak FA1 ka Cout FA 2 k Cin mein nhi jayega tab tak FA2 ka Sum nhi milega or Cout bhi nhi milega.

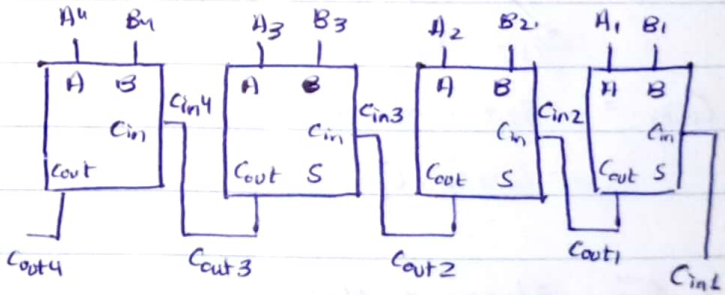
(CARRY LOOK AHEAD ADDER)

→ To solve ripple carry problem there is another adder called carry lookahead adder.

→ It predicts the carry.

$$C_{out} = \underbrace{A \cdot B}_{\text{Carry generation}} + \underbrace{(A \oplus B)}_{\text{Carry propagation}} C_{in}$$

$$C_{out} = C_g + C_p C_{in}$$



Full Adder 4: FA 3 FA 2 FA 1

$C_{g4} = A_4 B_4$ $C_{g3} = A_3 B_3$ $C_{g2} = A_2 B_2$ $C_{g1} = A_1 B_1$

$C_{p4} = A_4 + B_4$ $C_{p3} = A_3 + B_3$ $C_{p2} = A_2 + B_2$ $C_{p1} = A_1 + B_1$

→

Foundation for Advancement of Science & Technology

- **FULL ADDER #01:**

$Cout_1 = C_{g1} + C_{p1} Cin_1$

- **FULL ADDER #02:**

$Cin_2 = Cout_1$

$Cout_2 = C_{g2} + C_{p2} Cin_2$

$C_{o} = C_{g2} + C_{p2} Cout_1$

$= C_{g2} + C_{p2} (C_{g1} + C_{p1} Cin_1)$

$Cout_2 = C_{g2} + C_{p2} C_{g1} + C_{p2} C_{p1} Cin_1$

- **FULL ADDER #03:**

$Cin_3 = Cout_2$

$Cout_3 = C_{g3} + C_{p3} Cin_3$

$= C_{g3} + C_{p3} Cout_2$

$Cout_3 = C_{g3} + C_{p3} C_{g2} + C_{p3} C_{p2} C_{g1} + C_{p}$

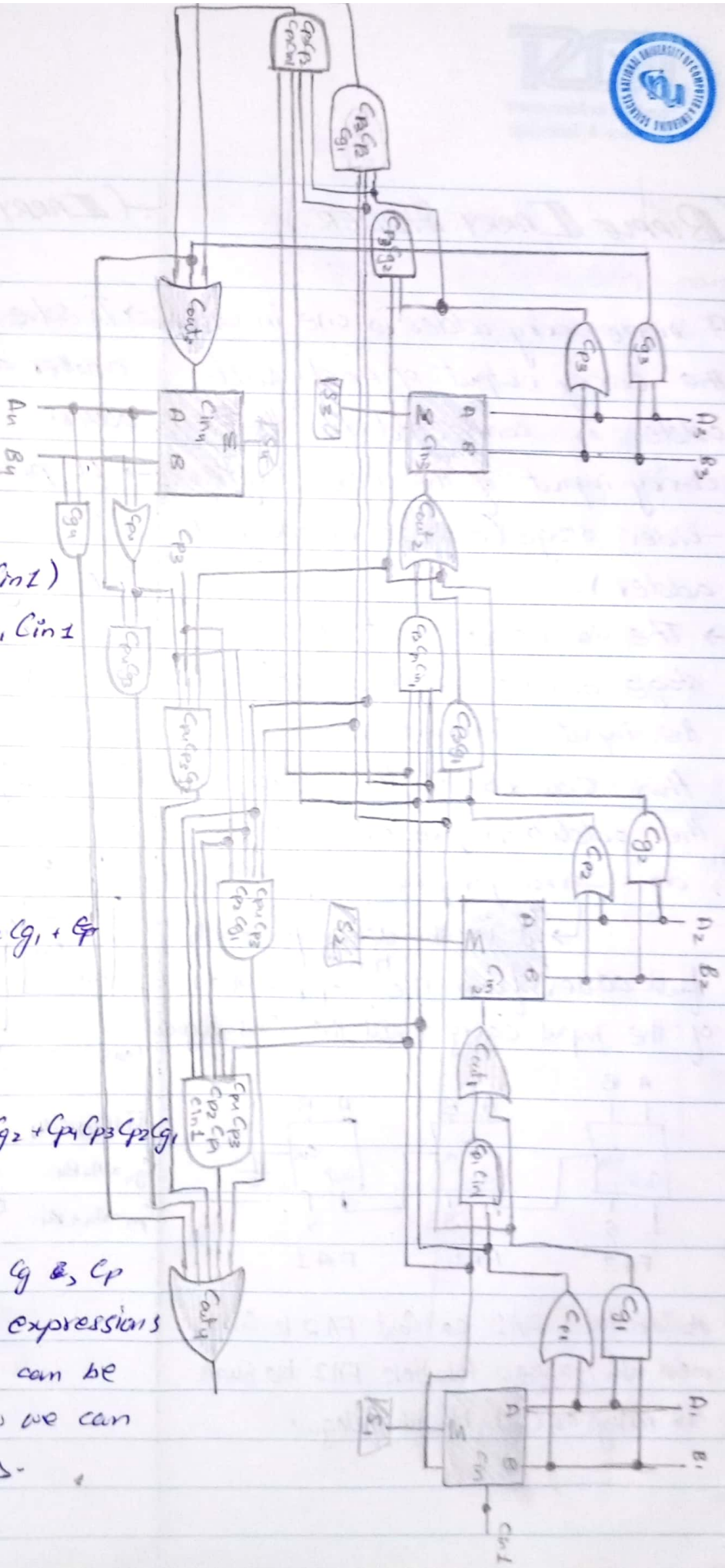
$+ C_{p3} C_{p2} C_{p1} Cin_1$

- **FULL ADDER #04:**

$Cout_4 = C_{g4} + C_{p4} C_{g3} + C_{p4} C_{p3} C_{g2} + C_{p4} C_{p3} C_{p2} C_{g1}$

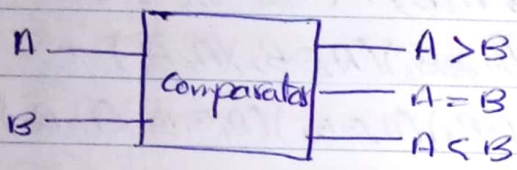
$+ C_{p4} C_{p3} C_{p2} C_{p1} Cin$

⇒ Now we have only C_g & C_p & Cin involved in the expressions of Carry output. C_g & C_p can be converted to A&B. So now we can get Cout without any delays.



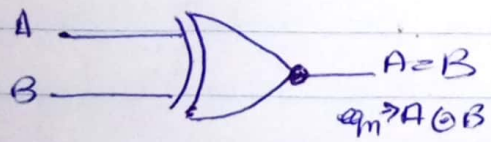
COMPARATOR

it is used to compare n-bit No. to check whether they are equal, greater or less than eachother.



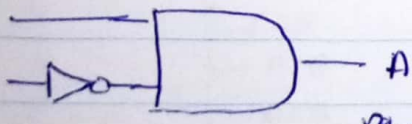
A	B	A=B	A>B	A<B
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

→ XNOR gate is used to check equality



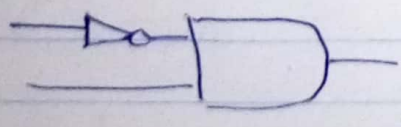
eqn → $A \odot B$

→



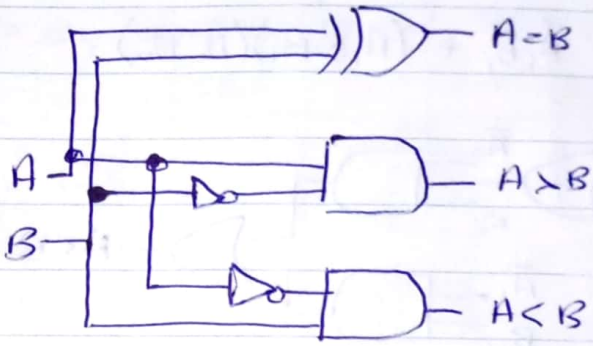
eqn → $A\bar{B}$

→



eqn → $\bar{A}B$

1-Bit Comparator

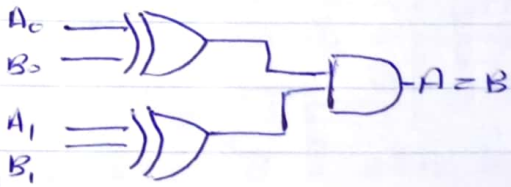


2-Bit Comparator

[EQUALITY]

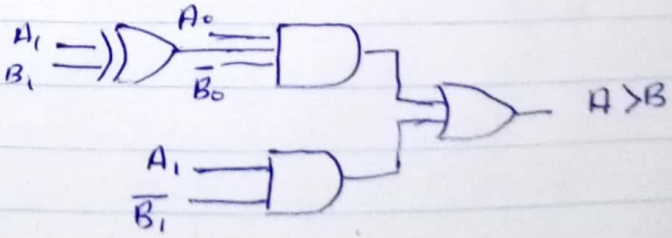
$A = A_1 A_0 \quad B = B_1 B_0$

$(A=B) = (A_1 \odot B_1)(A_0 \odot B_0)$



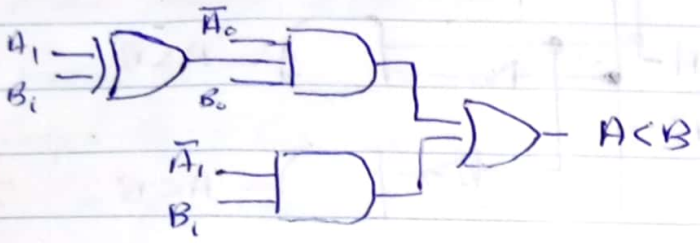
[GREATER THAN]

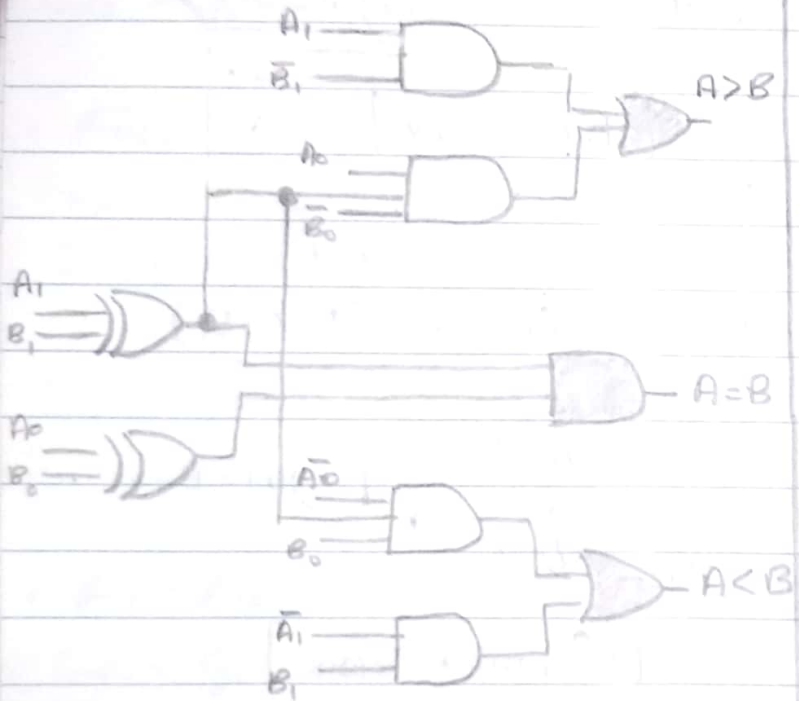
$A>B = A_1\bar{B_1} + (A_1 \odot B_1)(A_0\bar{B_0})$



[LESS THAN]

$A<B = \bar{A}_1 B_1 + (A_1 \odot B_1)(\bar{A}_0 B_0)$





▲ 2-Bit Comparator

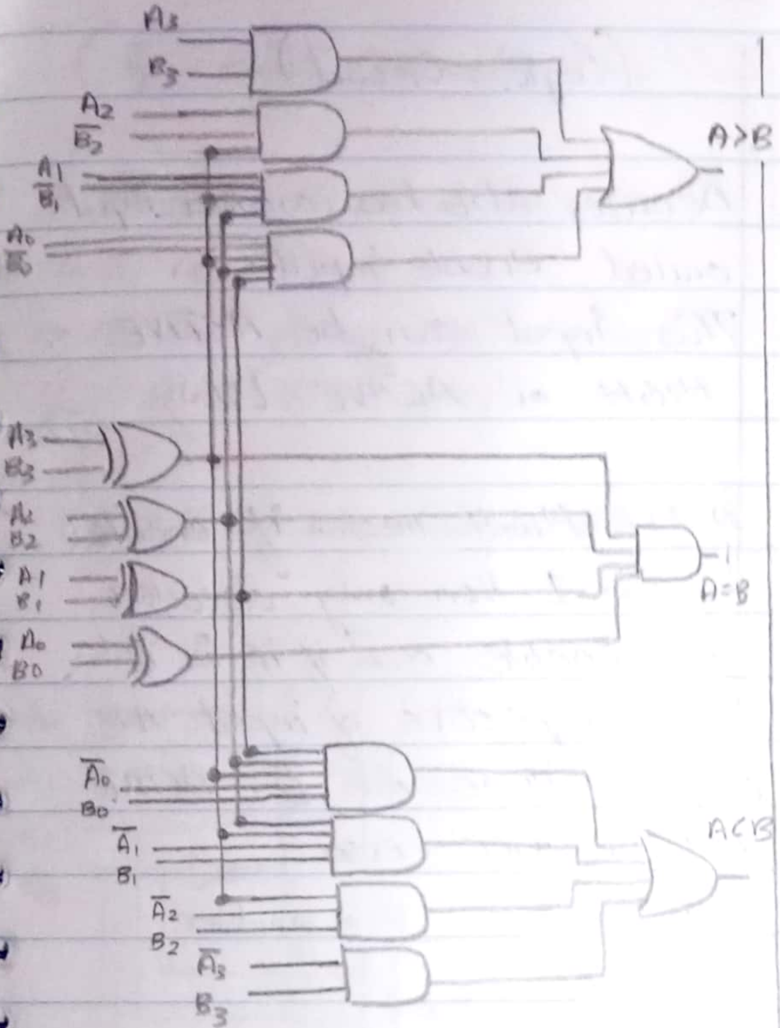
4-Bit Comparator

Equality → $(A_3 \odot B_3)(A_2 \odot B_2)$
$(A_1 \odot B_1)(A_0 \odot B_0)$

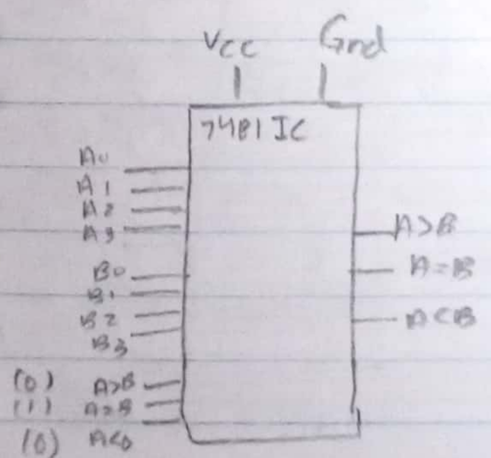
Greater

$A>B \rightarrow A_3 \bar{B}_3 + (A_3 \odot B_3)(A_2 \bar{B}_2) +$
$(A_3 \odot B_3)(A_2 \odot B_2)(A_1 \bar{B}_1) +$
$(A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \bar{B}_0)$

$A>B = \bar{A}_3 B_3 + (A_3 \odot B_3)(\bar{A}_2 B_2) +$
$(A_3 \odot B_3)(A_2 \odot B_2)(\bar{A}_1 B_1) +$
$(A_3 \odot B_3)(A_1 \odot B_1)(A_1 \odot B_1)(\bar{A}_0 B_0)$

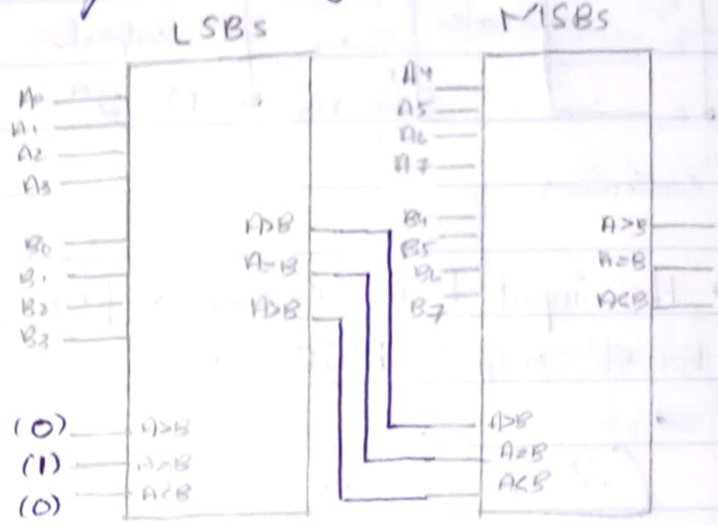


▲ 4-Bit Comparator



—(8-Bit Comparator)—

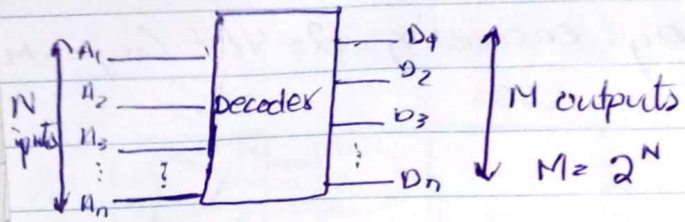
- By cascading 2-4bit Comparator



phele ~~LSBs~~ MSB check hongi agr saari bits equal hai to wo neeche waale results check karega jo LSB waali se aaate hai ~~wo aage~~ agr greater ya less hoi to results ~~aage waale mein transfer kardega~~ to wo output dedega. Isi waja se LSB waali IC ka A=B 1 rakha hai take agr saari bits equal ho or comparator neeche wale results check kare to use A=B mile waan agr use bhi zero rakh dete to result shi nhi milta.

CamScanner

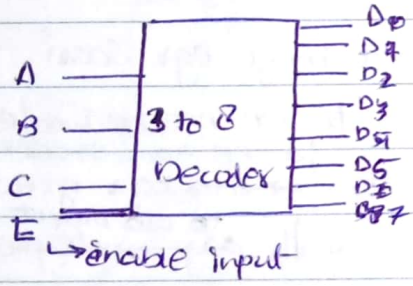
(DECODERS)



2^n combinations

- Har input k ek combination par koi ek output HIGH hoga.

(3 To 8 Decoder)



Depending on the input combination of these three bits (A, B, C) the specific output will be HIGH.

foreg: if A=0, B=1, C=0 then D_2 will be high bcz 010 = 2

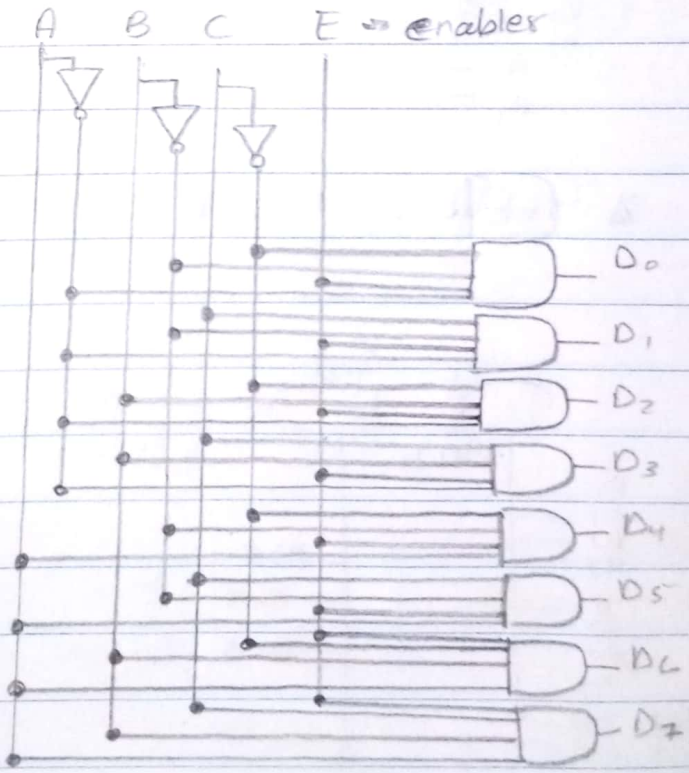
for 100 (A B C) → D_4 will be high

i.e why 3 to 8 decoder can be used as ~~Octal~~ Binary to Octal Convertor.

(logic Circuit)

Decoder also has another input called enable input. This input can be ACTIVE HIGH or ACTIVE LOW.

ACTIVE HIGH means if enable input is 1 then only decoders gets enable and if it is '0' then issespective of input the output will remain 0. ACTIVE LOW is vice versa.

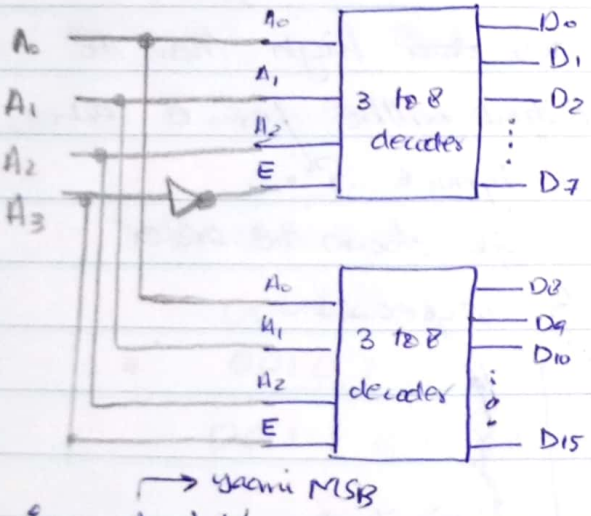


(4-Bit Decoder)

∴ there are 4 input so outputs will be $2^4 = 16$

→ 16 And gates are required for 4 to 16 decoder.

(4 to 8 Decoder Using 3 to 8 decoders)



→ yaani MSB

jab A_3 '1' hoga to upper wala decoder band hojayega & k enables 0 hojayega yaani 0–7 tak upper wala chalega or 8–15 tak neeche wala & k 0–7 mein MSB '0' hi rhegi hamesha.

(BCD To Decimal Decoder)

and an enables

it has 4 inputs but ten outputs. BCD we have only 10 outputs in BCD code.

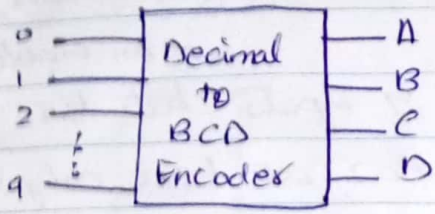
(ENCODERS)

It is a combinational circuit that performs a "reverse" decoder function.

It accepts an active level on one of its inputs representing a digit such as a decimal or octal digit and converts it to a coded output such as BCD or binary.

(Decimal To BCD)

	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1



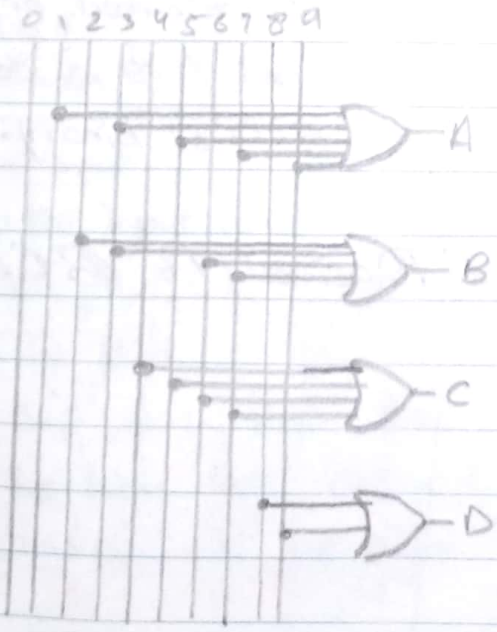
We can find expression by Truth Table for each output.

$A = 1 + 3 + 5 + 7 + 9$ → we can OR 1,3,5,7,9 to get A

$B = 2 + 3 + 6 + 7$ → we can OR 2,3,6,7 to get B

$C = 4 + 5 + 6 + 7$

$D = 8 + 9$



Priority encoders:

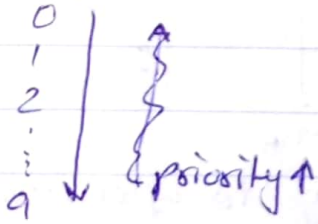
→ It performs the same encoding function. It provides additional flexibility.

The priority function means that the encoder will produce an output corresponding to the highest order input that is active.

For eg:

if 6 2 3 are inputs and ~~both~~ are ~~acti~~ high than the BCD output willbe for 6 as it will ignore 3.

As we go down the order priority increases.



(BCD TO BINARY)

Convert BCD 00100111 (27) into Binary

0010 0111

left most 4-bits ka weight 10 hai | inka weight 1 hai

0 0 1 0 0 1 1 1
80 40 20 1 8 4 2 1

Ab jo 1 wala hai unko add karenge

0 0 0 0 0 0 0 1 → 1
0 0 0 0 0 0 1 0 → 2
0 0 0 0 0 1 0 0 → 4
+ 0 0 1 0 1 0 0 → 20
0 0 1 1 0 1 1 → 27

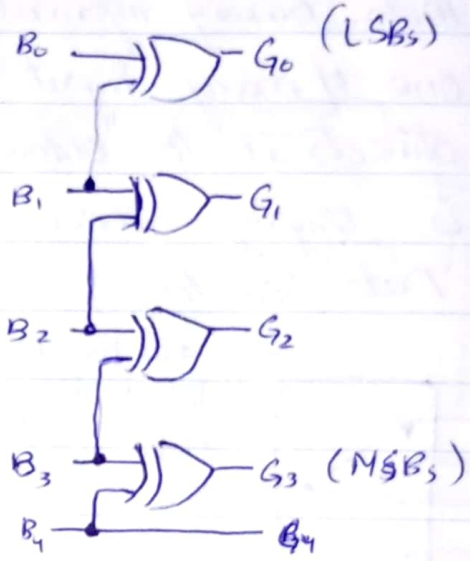
Example #02 BCD 10011000 (98)

80 40 20 10 8 4 2 1
1 0 0 1 1 0 0 0

0 0 0 1 0 0 0 → 8
0 0 0 1 0 1 0 → 10
+ 1 0 1 0 0 0 0 → 80
1 1 0 0 0 1 0 → 90

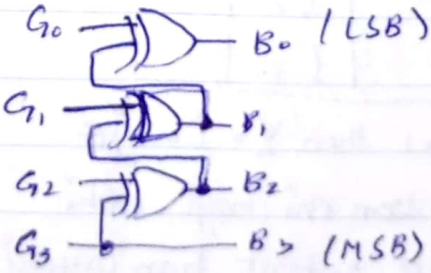
(BINARY TO GRAY)

Binar Gray



eg: 0101 → Gray (using XOR)

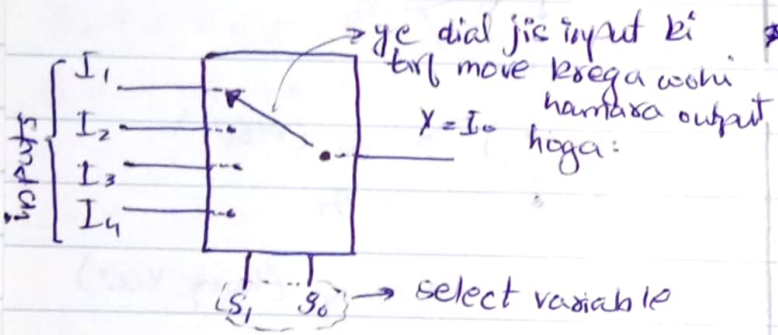
(GRAY TO BINARY)



(MULTIPLEXER)
—MUX—

"It is a combinational circuit that selects binary information from one of many input lines and directs it to output lines.

→ It is simply called as Data selector.

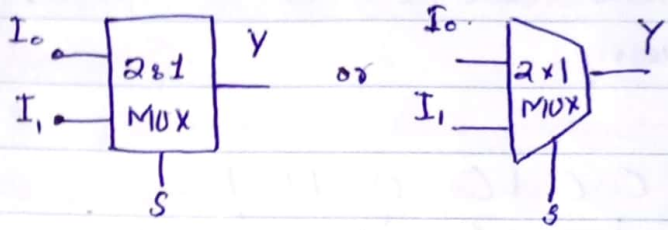


→ This dial is move by selectline or select variable.

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

jab $S_1 = 0$, $S_0 = 1$ then $Y = I_0$. jo decimal value bani rhi hogi wohi step input line output ban jayegi

Representation:



→ OUTPUT is always '1' in MUX.

* Relation b/w Input & Selector variable

$n = 2^m$ → no. of selector lines variable

n → no. of I/P

eg $n = 4 = 2^2 \therefore m = 2$

$n = 8 = 2^3 \therefore m = 3$

* ADVANTAGES :-

→ Reduces no. of wires / gate

→ Reduces circuit complexity & cost

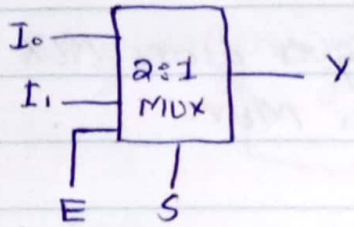
→ Various combination circuits can be implemented

* TYPES :

→ 2:1 MUX, 4:1 MUX, 8:1 MUX
16:1 MUX, 32:1 MUX

Foundation for Advancement of Science & Technology

{ 2 : 1 MUX }



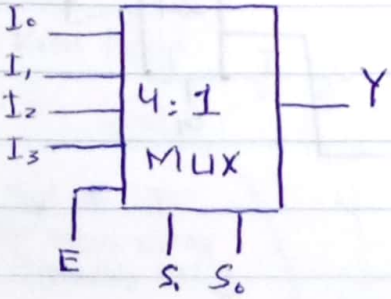
(Enable)
when it is 0' circuit will not work.

E	S	Y
0	X	0
1	0	I_0
1	1	I_1

$Y = E \cdot \bar{S} \cdot I_0 + E \cdot S \cdot I_1$

$Y = E(\bar{S} I_0 + S I_1)$

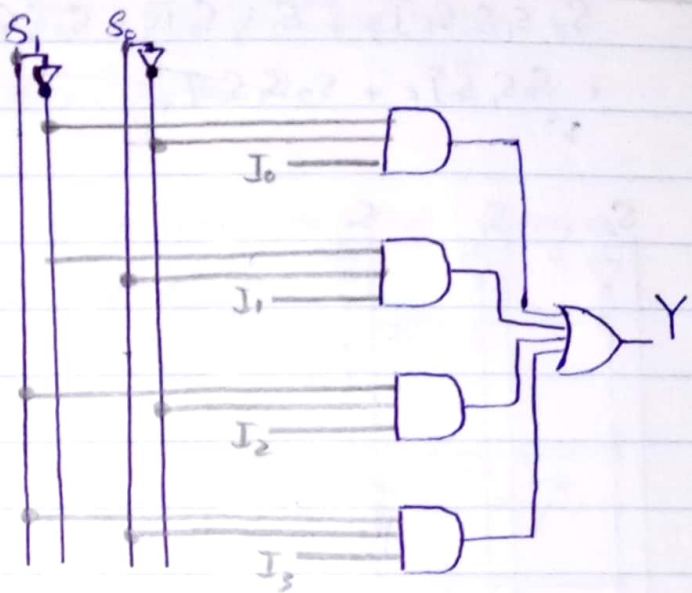
[4 : 1 MUX]



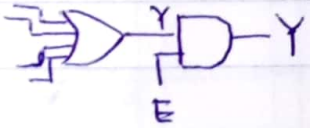
Enabler ko abhi humne ignore kardiya hai

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

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

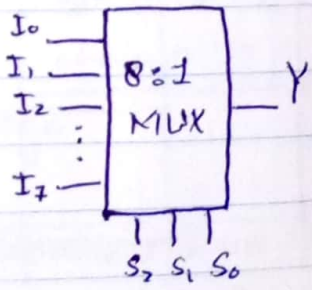


ago is mein enable dalna hai to hum AND output wale OR k age k AND gate lgadenge or usko enable k saath connect kardenge →



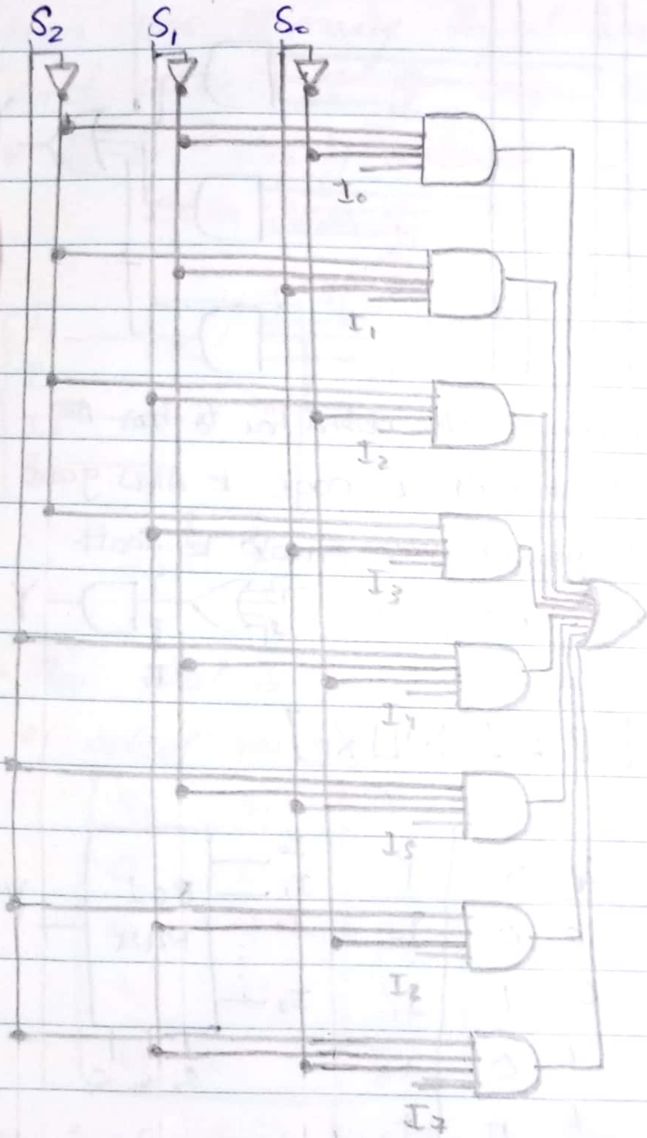
[8 : 1 MUX]

S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7



Foundation for Advancement of Science & Technology

$Y = \bar{S_2}\bar{S_1}\bar{S_0}I_0 + \bar{S_2}\bar{S_1}S_0I_1 + \bar{S_2}S_1\bar{S_0}I_2$
$\bar{S_2}S_1S_0 S_2 I_3 + S_2\bar{S_1}\bar{S_0}I_4 + S_2\bar{S_1}S_0I_5$
$+ S_2S_1\bar{S_0}I_6 + S_2S_1S_0I_7$



enabler ke liye pichle wale ki hi tarha kardenge.

(MUX → TREE)

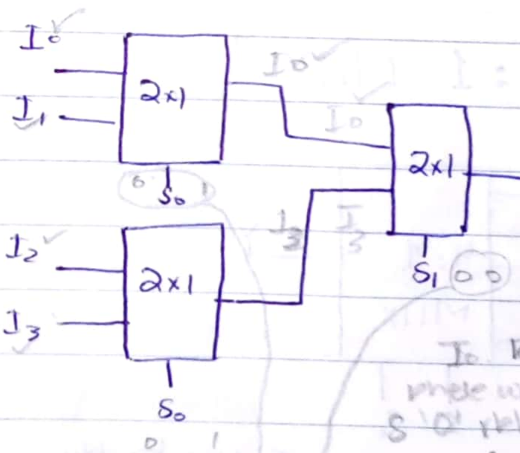
→ Obtaining higher order MUX using lower order MUX.

—(4 : 1 MUX Using 2 : 1 MUX)—

$n = 4 \Rightarrow \frac{4}{2} = (2) \Rightarrow \frac{2}{2} = (1)$

(+) ↳ = 3 → we will use 3 2×1 MUX to make 4:1 MUX

→ order bhi same hoga phele 2 2×1MUX use karenge phir 1 use karenge



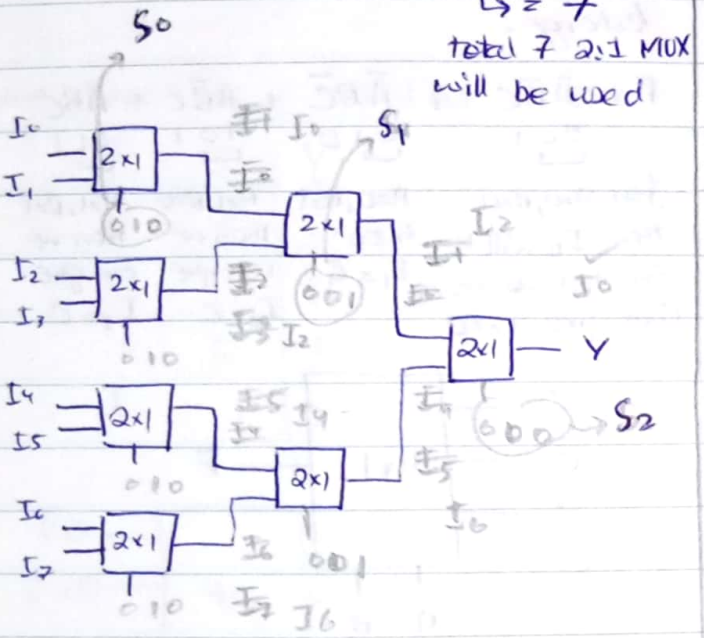
See NESO lec # 121

(8:1 MUX Using 2:1 MUX)

$n=8 \Rightarrow \frac{8}{2} \Rightarrow (4) \Rightarrow \frac{4}{2} \Rightarrow (2) \Rightarrow \frac{2}{2} \Rightarrow (1)$

$\hookrightarrow = 7$

total 7 2:1 MUX will be used



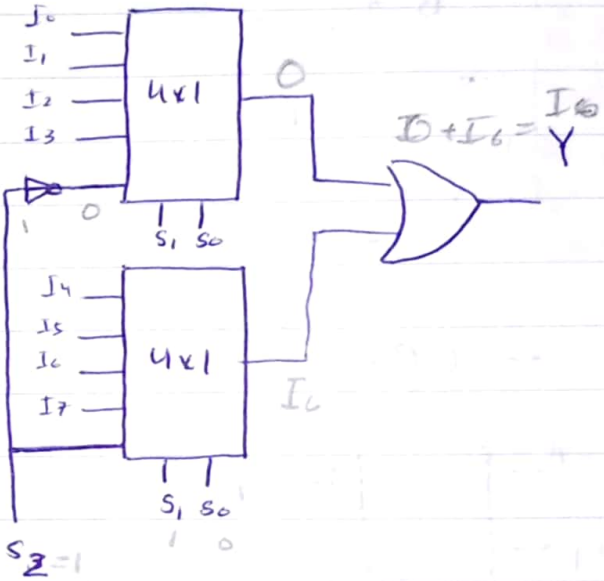
Select lines humein behud table se find out krni hongi.

(8:1 MUX Using 4:1 MUX)

— Special Case —

$n=8, \quad \frac{8}{4} \Rightarrow 2$

hum yaha enables use krenge apne S_2 k liye.



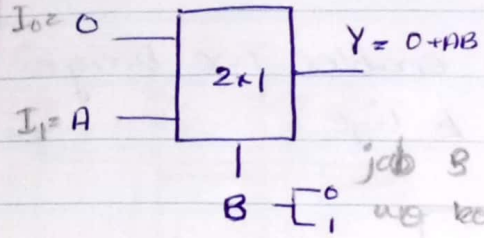
If we want I_6 then $S_2=1, S_1=1, S_0=0$ to cab neeche wala chalega or upper wala nhi chalega

- MUX tree ko implement krne k liye Truth table boht imp hai.

Foundation for Advancement of Science & Technology

(IMPLEMENTING BASIC GATES)

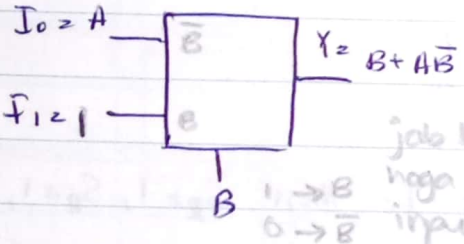
— AND —



jab B (0) hoga to we ko select krega or jab 1 hoga to A ko.

B → 1 par aya hum $I_1 = A$ kadein to humein aya → AB miljayegi.

— OR —

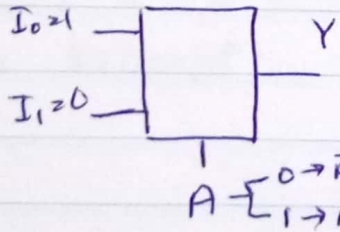


jab B=1 yani B hoga to hum 1 input dedenge take B=1 = B or agr B=0 yani $\bar{B}$ hoga to A input dedenge take $A\bar{B}$ miljaye.

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

$\bar{A}B + A\bar{B} + AB$
$B(\bar{A}+A) + A\bar{B}$
$B + A\bar{B}$ ✓

— NOT —



to ab jab A=0 hoga to $Y = I_0 = 1$ output miljega.

(IMPLEMENTING FONCTION)

$F(A,B,C) = \Sigma(1,2,5,7)$

kisi do variables ko S_0, S_1 choose krenge.

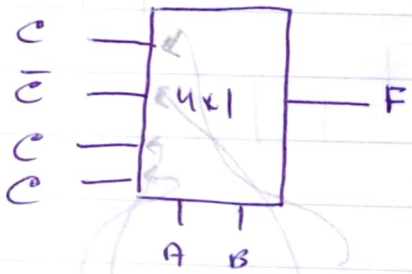
$F = \underbrace{\bar{A}\bar{B}C}_{001} + \underbrace{\bar{A}B\bar{C}}_{010} + \underbrace{A\bar{B}C}_{101} + \underbrace{ABC}_{111}$

jab A=0, B=0 then I_0 will be selected so we can give $I_0 = C$

A=0, B=1 then $I_1 = \bar{C}$

A=1, B=0 then we can give $I_2 = C$

A=1, B=1 then we can give $I_3 = C$



A B
0 0 → $\bar{A}\bar{B}C$
0 1 → $\bar{A}B\bar{C}$
1 0 → $A\bar{B}C$
1 1 → ABC

Q: $F(A,B,C,D) = \Sigma m(1,4,5,7,9,12,13)$

$\bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD$
0001 0100 0101 0111

$A\bar{B}\bar{C}D + A\bar{B}C\bar{D} + A\bar{B}CD$
1001 1100 1101

→

$\bar{C}\bar{D} + \bar{C}D + CD$
$\bar{C}(\bar{D}+D) + CD$
$\bar{C} + CD$ $\bar{C}+D$

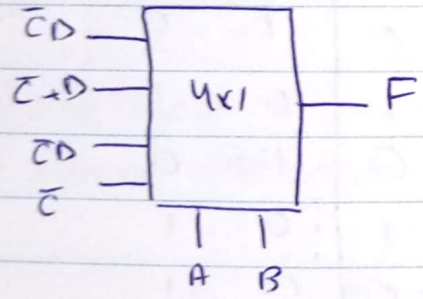
Foundation for Advancement of Science & Technology

~~ARG~~ (ANOTHER METHOD)

AB \ CD	00	01	11	10	
00		1			→ $C\bar{D} = I_0$
01	1	1	1		→ $\bar{C}+D = I_1$
10	1	1			→ $\bar{C} = I_2$
10		1			→ $\bar{C}D = I_3$

we have choose A & B as S_1 & S_0

S_1	S_0	Y
0	0	$I_0 = \bar{C}D$
0	1	$I_1 = B\bar{C}+D$
1	0	$I_2 = \bar{C}D$
1	1	$I_3 = \bar{C}$



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

no. of variable = n = 3

no. of selection line = n-1 = 3

Multiplexer = $2^{n-1} = 4:1$

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

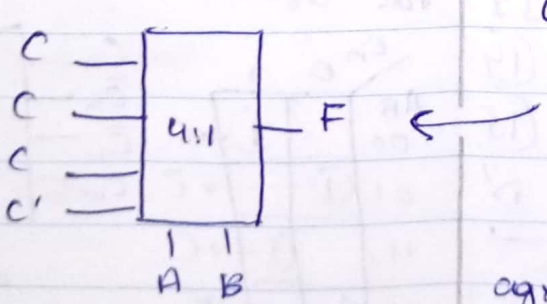
kisi bhi do variable ko selector k liye use kr sakte hai.

	I_0	I_1	I_2	I_3
C'	0	2	4	(6)
C	(1)	(3)	(5)	7
	C	C	C	C'

jis jaha output 1 hai us number ko circle krdenge

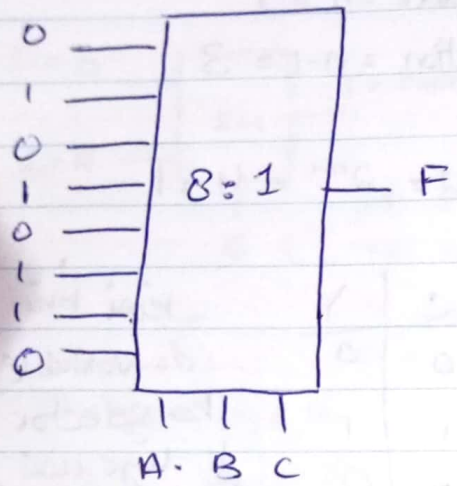
jis variable par circle hai usko likhdenge aur C wali line par ~~to~~ C or agr C' wali par ~~to~~ C'

agr dono mein circle nhi ~~to~~ '0' or dono mein circle ~~to~~ '1'.



(ANOTHER METHOD)

→ isi saaad ko 8:1 se bhi kr sakte hai.



is method mein direct bus Truth table hi implement krna hota hai.

Q: $F(A,B,C,D) = \Sigma(0,1,2,3,4,9,13,14,15)$

$n = 4$

no. of selector = 3

Multiplexer = $2^3 = 8$

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
D'	(0)	(2)	(4)	6	8	10	12	(14)
D	(1)	(3)	5	7	(9)	11	(13)	(15)
	1	1	D'	0	D	0	D	D'

hum bs ye input $I_1, I_2, \dots I_{15}$ mein denge or A, B, C selector honge.

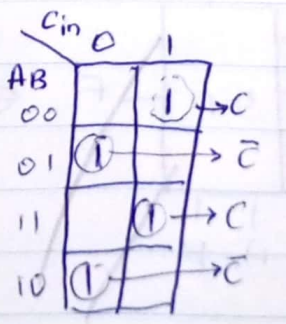
→ isi saaad ko 16:1 se krte to direct truth table hi map krna parta sirf.

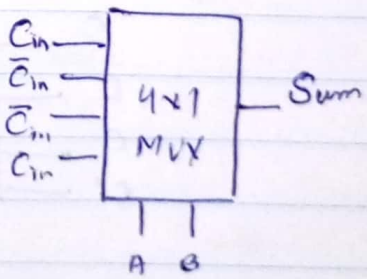
→ Agr isi saaad ko 4:1 se krna hai to Karnaugh map waala method use krna hoga.

(1-BIT FULL ADDER)

A	B	C_{in}	S	C_o
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

For S:-

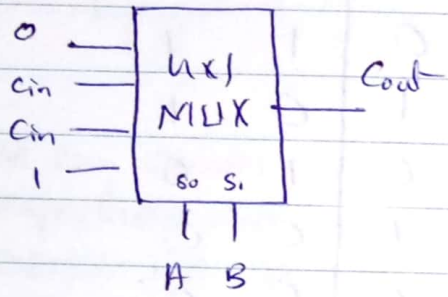




For Cout

	I_0	I_1	I_2	I_3
C_{in}'	0	2	4	(6)
C_{in}	1	(3)	(5)	(7)

0 C_{in} C_{in} 1



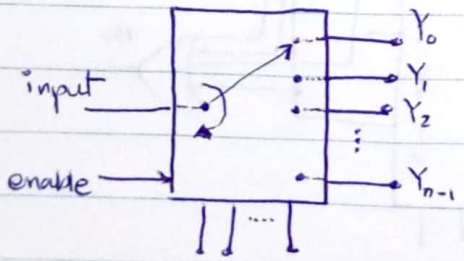
(DEMULTIPLEXER)

— DEMUX —

→ One input & many output

→ Reverse operation of multiplexer

→

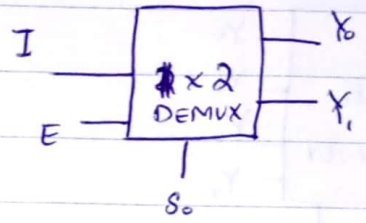


n → o/p lines

m → select lines

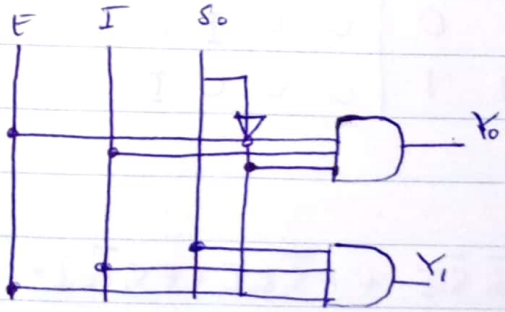
$n = 2^m$

(1:2 DEMUX)



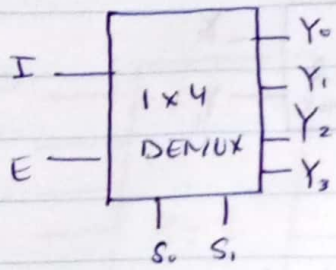
E	S_0	Y_0	Y_1
0	0	0	0
0	1	0	0
1	0	I	0
1	1	0	I

$Y_0 = E\bar{S_0}I$ $Y_1 = ES_0I$



Foundation for Advancement of Science & Technology

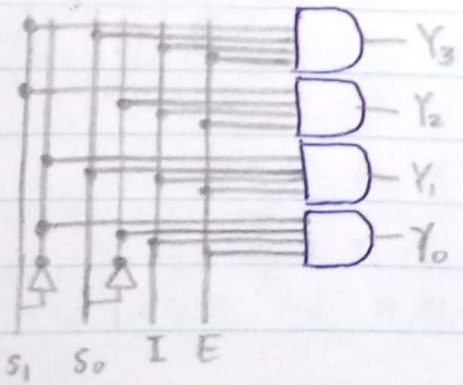
(1 : 4 DEMUX)



E	S0	S1	Y0	Y1	Y2	Y3
~~0~~	~~X~~	~~X~~	~~0~~	~~0~~	~~0~~	~~0~~
~~0~~	~~X~~	~~X~~	~~0~~	~~0~~	~~0~~	~~0~~
~~0~~	~~X~~	~~X~~	~~0~~	~~0~~	~~0~~	~~0~~
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

Enable high

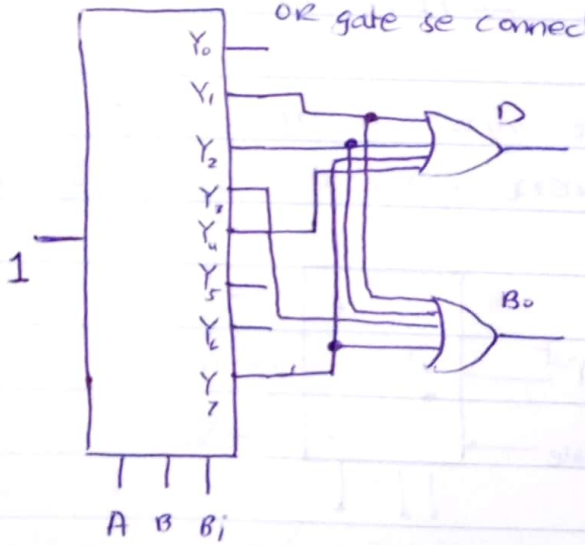
$Y_0 = E\bar{S}_1\bar{S}_0I + E\bar{S}_1S_0I + ES_1\bar{S}_0I + ES_1S_0I$



(FULL SUBTRACTOR USING 1:8 DEMUX

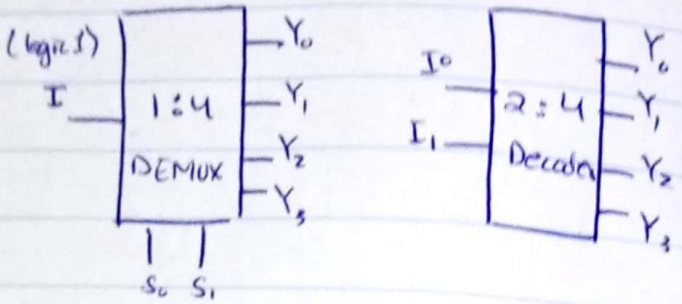
	A	B	Bi	D	Bo
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

jis jis k liye D '1' hai usko hum 'D' wale OR gate se connect krenge



Foundation for Advancement of Science & Technology

(DEMUX As DECODER)



We can consider S_0 & S_1 as I_0 & I_1 respectively and will provide I of DEMUX with permanent logic 1.

$\therefore S_0 = I_0$

$S_1 = I_1$

$I = 1$

By doing this our 1:4 DEMUX will work as 2:4 Decoder.