[UNIT-2]

Camlin Page
Date / /

LOGIC GATES AND COMBINATIONAL LOGIC

→ Logic gates are the basic building blocks for forming digital electronic circuitry.

→ A logic gate has one o/p terminal and one or more i/p terminals.

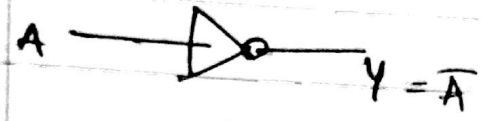
→ Its o/p would be high (1) or low (0) depending on the digital levels at the i/p terminals.

→ Logic gates can be used to design digital systems that will evaluate digital i/p levels and produce a specific o/p response based on that particular logic circuit design.

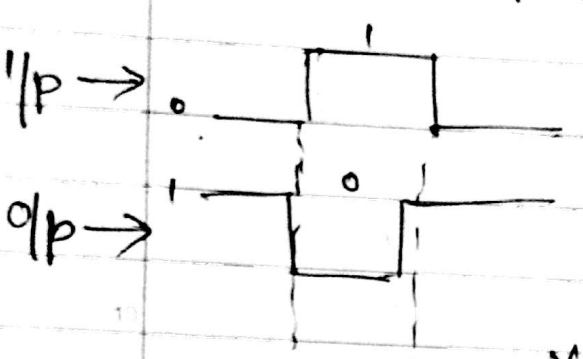
→ The Inverter, AND and OR gates are considered to be the basic logic gates. The NAND and NOR gates are known as universal gates. bcoz, of this reason either NAND or NOR gates are sufficient for the realization of any logical expression.

The Inverter -

Known as NOT gate that perform inversion (complementation).

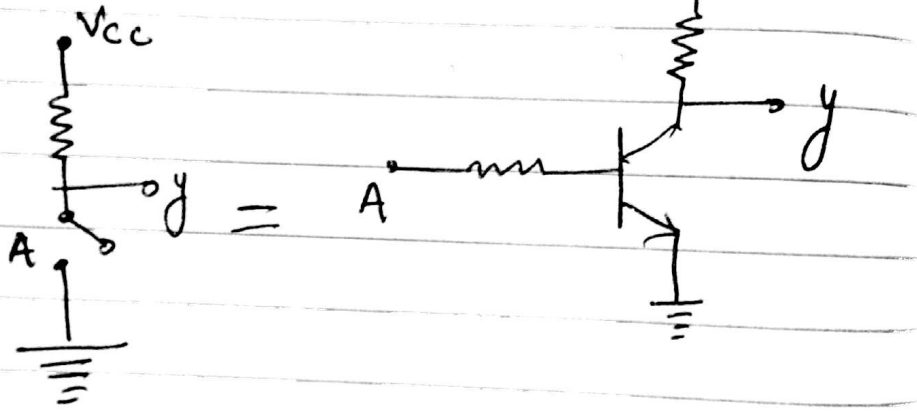


A	Y
0	1
1	0



Boolean expression

$Y = \overline{A}$



Open switch = off switch = logic 0
closed switch = on switch = logic 1

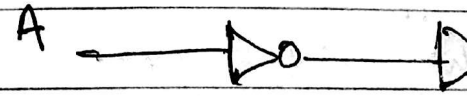
→ when inverter i/p is low (0), the o/p Y is high (1), on the other hand, if the i/p is high (1), the o/p is low (0)

Date / /

The AND gate –

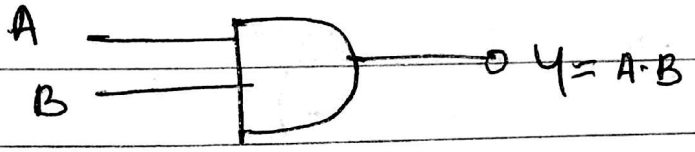
Buffer → produces the transfer function but does not produce any particular logic operation

This ckt is used merely for power amplification of the signal and is equivalent to 2 inverters connected in cascade.

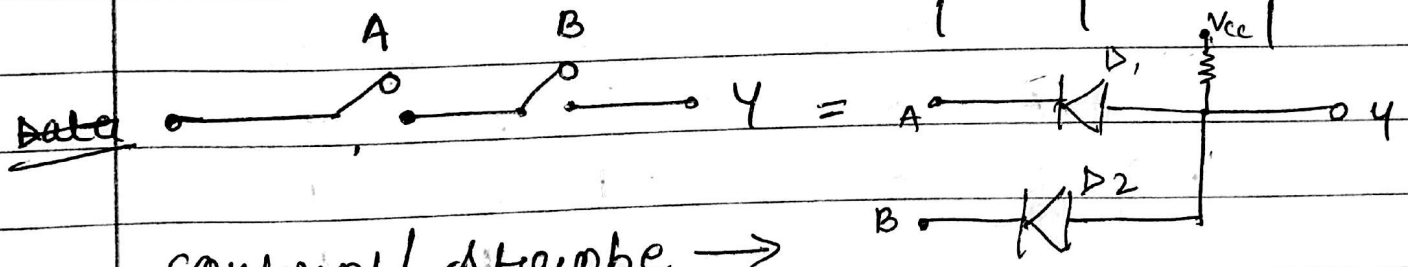


A	Y
0	0
1	1

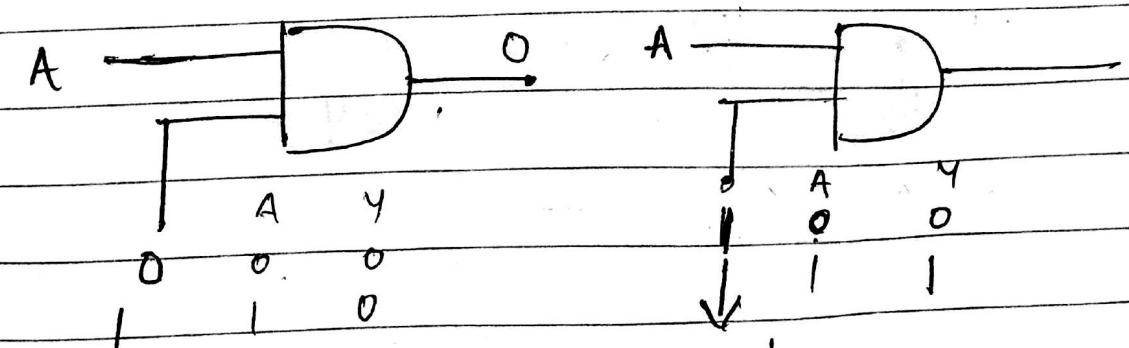
The AND gate –



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



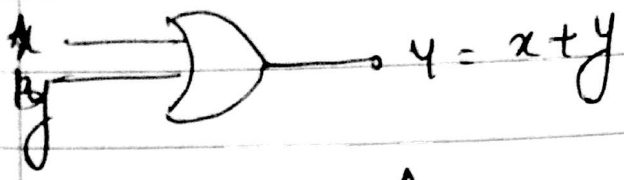
control / strobe →



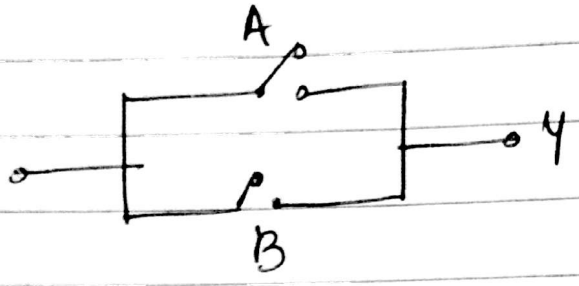
~~Disenable~~
Disable

enable

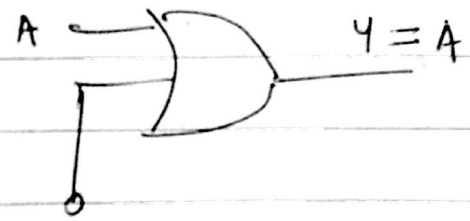
The OR gate –



x	y	f
0	0	0
0	1	1
1	0	1
1	1	1

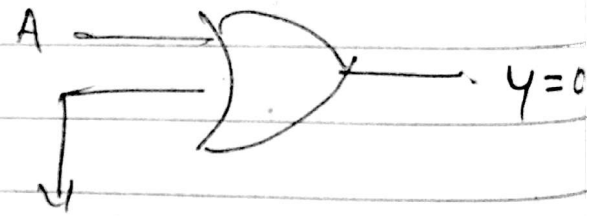


If either of the i/p signal is high the o/p of the OR gate is high. When both the i/p's are low then o/p of the gate is low.



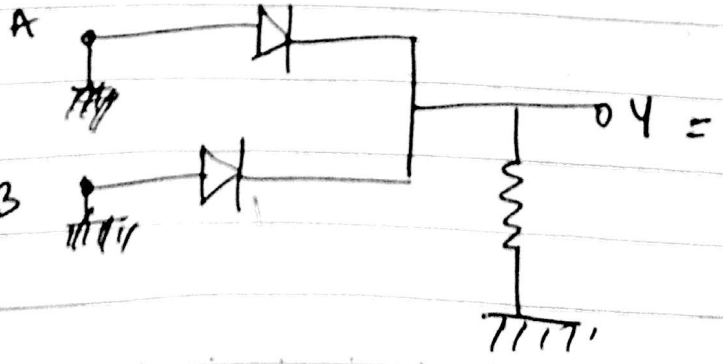
0 → enable

A	Y
0	0
1	1



1 → Disable

A	Y
0	1
1	1

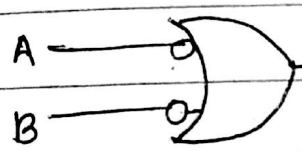


The NAND gate —
NOT- AND operation
Bubbled OR gate.

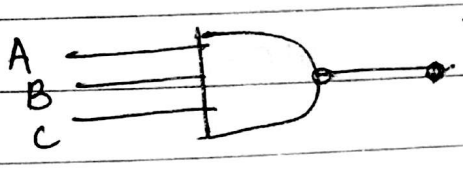
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

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

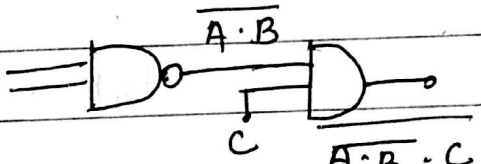
$= \overline{A} + \overline{B}$



follow commutative but not associative.

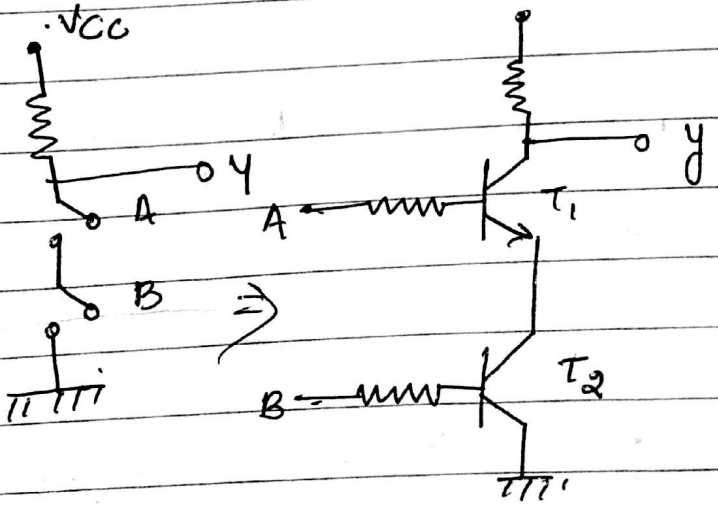


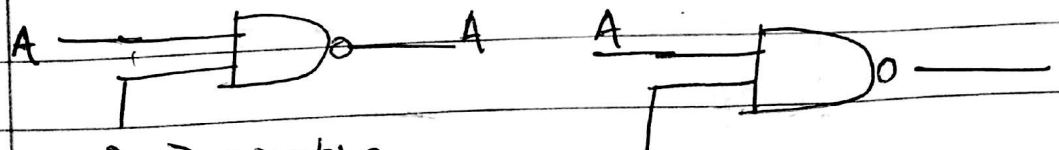
$\overline{A \cdot B \cdot C}$ =



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

$= AB + \overline{C}$





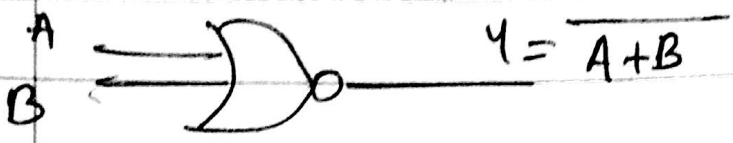
0 → Disable

1 → enable

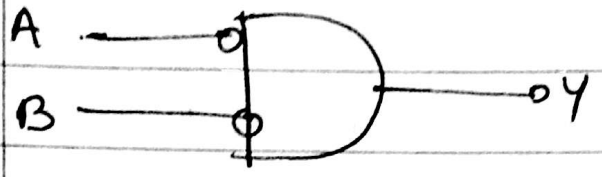
A	Y
0	0
	1
1	0

A	
1	1
0	1

The NOR gates - Bubbled AND



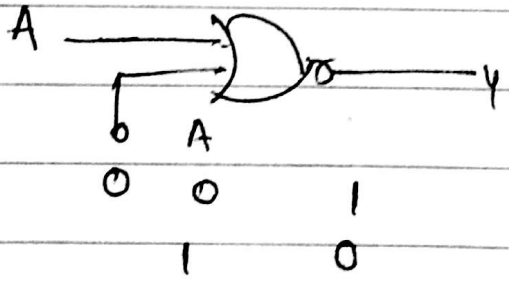
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

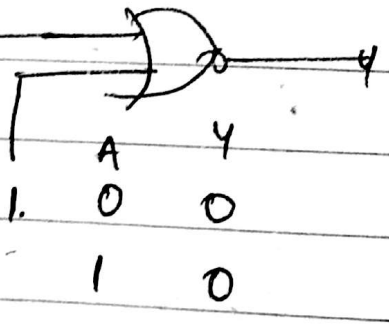


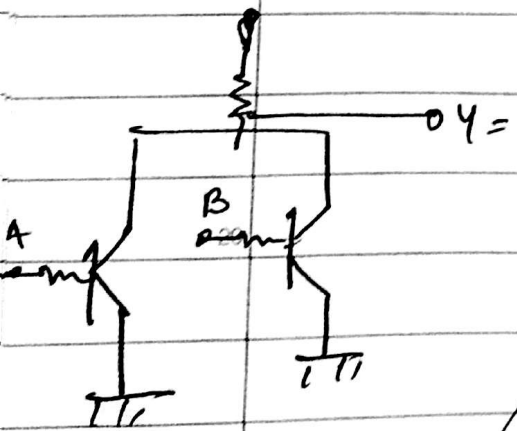
~~Bub~~

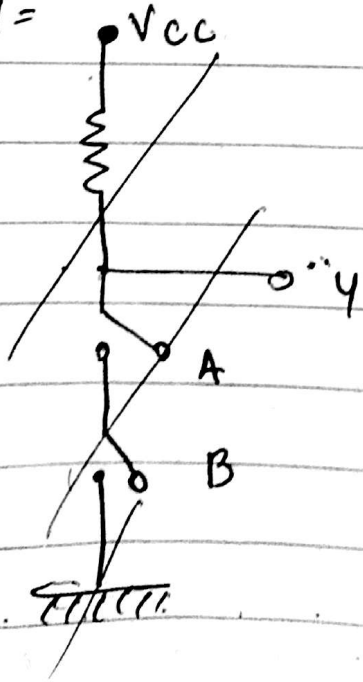
Not followed associative law.

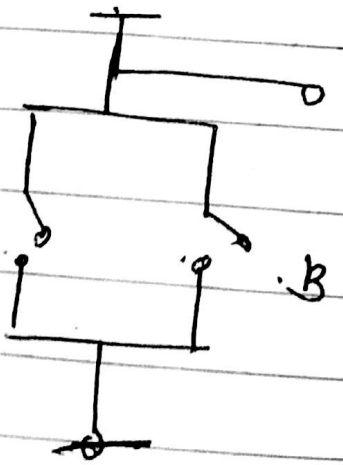
OR] 0 → enable
NOR] 1 → Disable.



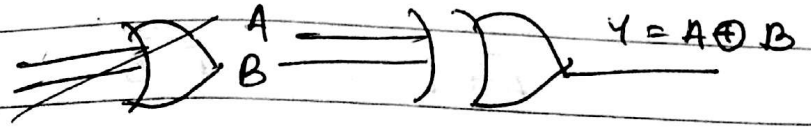








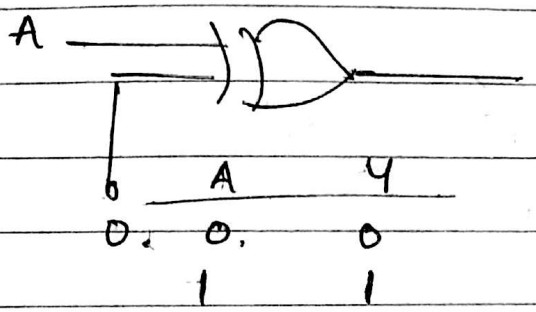
Ex-OR Gate –



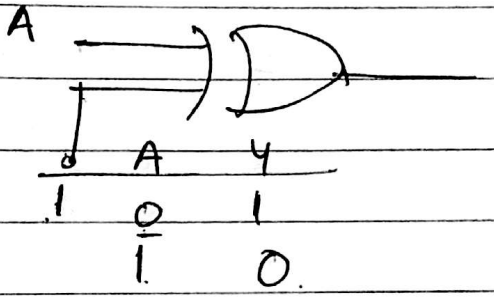
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

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

$= A \oplus B$



Acting as Buffer



Acting as Inverter.

In ex-OR both 0 & 1 is for enable i/p.

$A \oplus A = 0$

$A \oplus \bar{A} = 1$

$A \oplus 0 = A$

$\bar{A}A + A\bar{A} = 0$

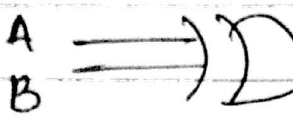
$\bar{A}\bar{A} + A \cdot \bar{A} = 1$

$\bar{A} \cdot 0 + 0 \cdot \bar{A} \cdot 1 = A$

In ExOR gate O/p is 1 when no. of 1's in the i/p is odd no. hence it is known as odd no. of 1's detector.

Camlin Page
Date / /

Ex-NOR gate -



$Y = \overline{A \oplus B} = AB + \bar{A}\,\bar{B}$

$= A \odot B$

$Y = \overline{A \oplus B}$

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

$= \overline{(\bar{A}B)} \cdot \overline{(A\bar{B})}$

$= (\bar{\bar{A}} + \bar{B}) \cdot (\bar{A} + \bar{\bar{B}})$

$= (A + \bar{B}) \cdot (\bar{A} + B)$

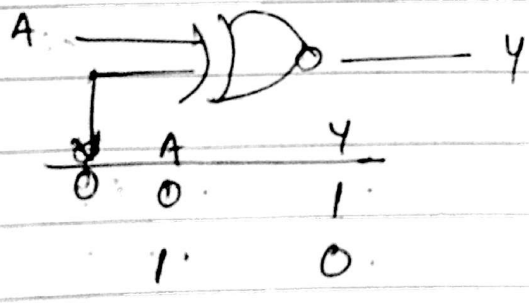
$= A\bar{A} + AB + \bar{A}\bar{B} + B\bar{B}$

$Y = AB + \bar{A}\bar{B}$

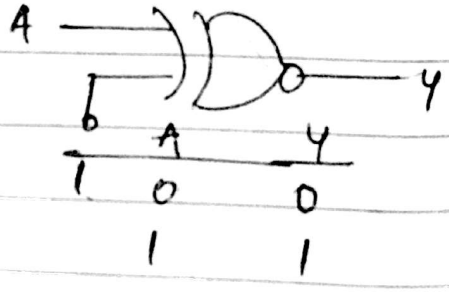
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

If $A = B$ then $Y = 1$

$A \neq B$ then $Y = 0$



Acting as Inverter



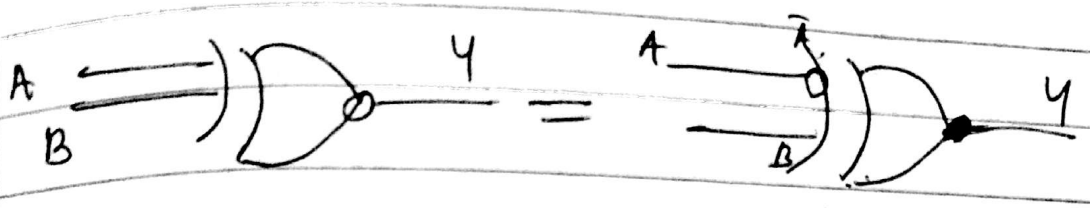
Acting as Buffer

$A \odot A = 1$

$A \odot \bar{A} = 0$

$A \odot 0 = \bar{A}$

$A \odot 1 = A$



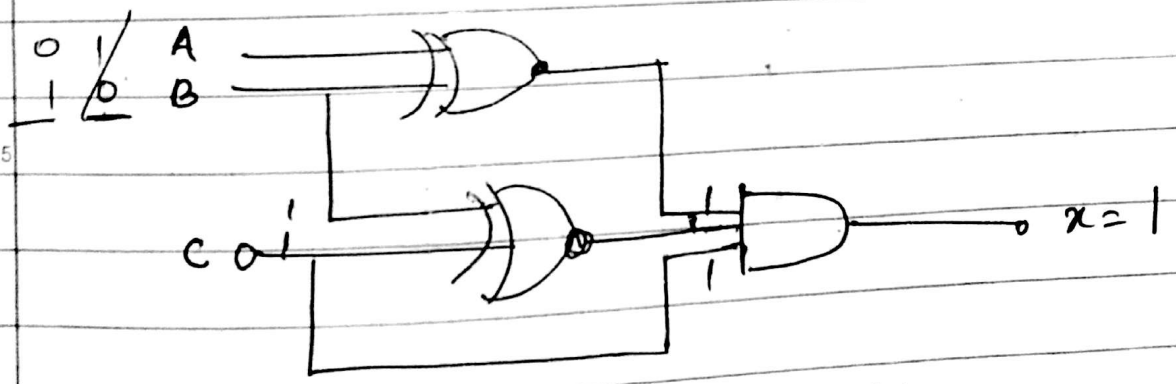
$Y = \overline{A}, B$

$Y = \overline{\overline{A}B} + \overline{A}\,\overline{B}$

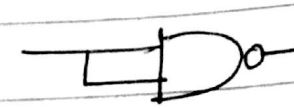
$= AB + \overline{A}\,\overline{B}$

$Y = A \odot B$

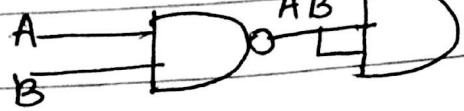
In the ckt shown in fig. HO provide O/P $x = 1$ I/Ps A, B & C be respectively — are —



NAND as universal gate-

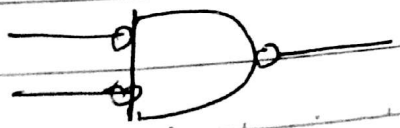
NOT → 

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

AND → 

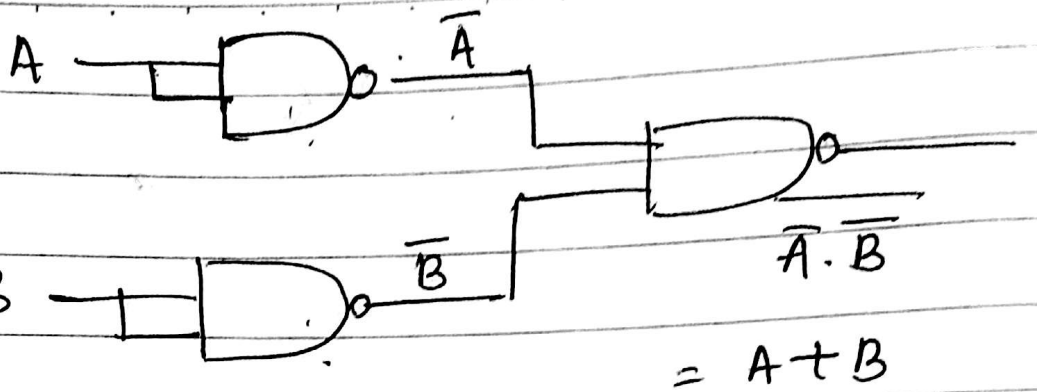
$\Rightarrow \overline{\overline{AB} \cdot \overline{AB}} = AB + AB$

AB

OR → 

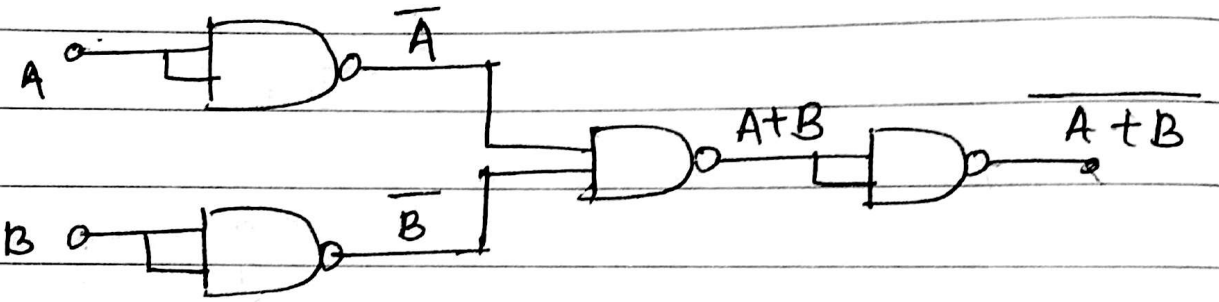
$\overline{A+B}$

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

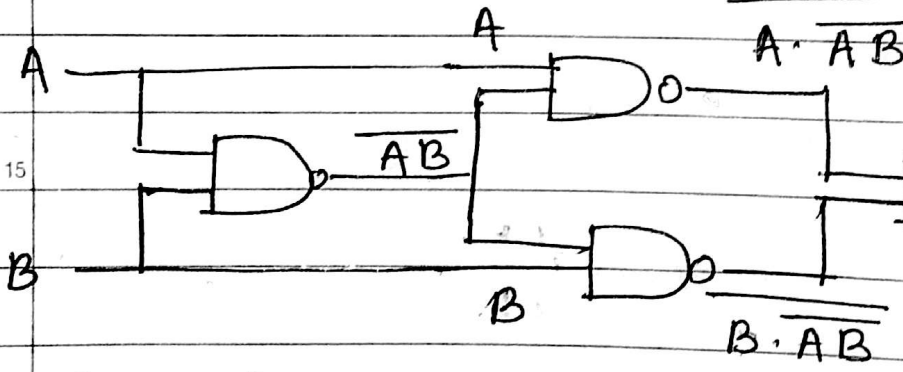


$= A + B$

NOR-



EX-OR



$\overline{A \cdot \overline{AB}} = (\overline{A} + AB) \cdot (\overline{B} + AB)$

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

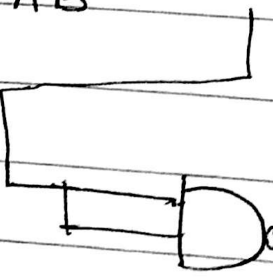
$\overline{A}\,\overline{B} \cdot \overline{AB}$

$\overline{(A+B)} \cdot \overline{(\overline{A}+B)} = \overline{AB}$

$\overline{\overline{AB}}$

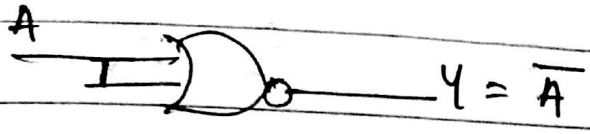
$A \oplus B$

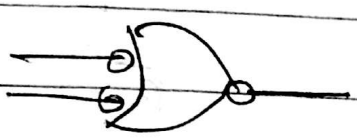
EX-NOR

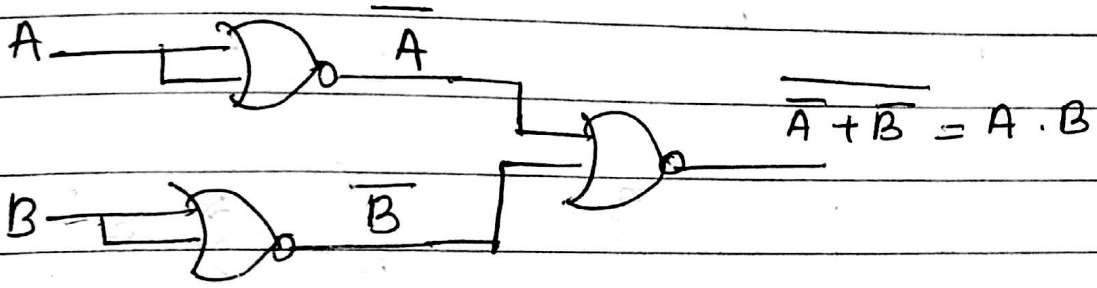


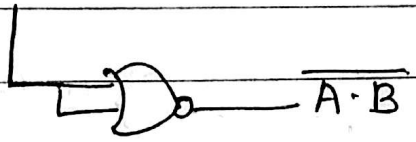
$A \odot B$

NOR as universal -

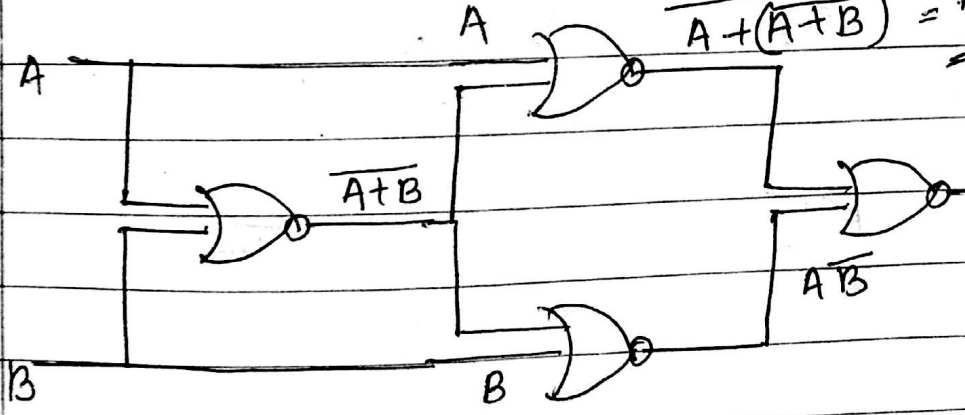
NOT → 

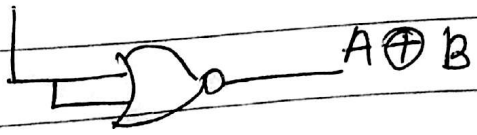
AND → 

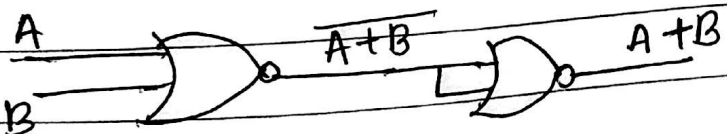


NAND → 

EX-NOR →

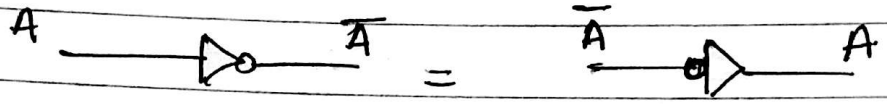


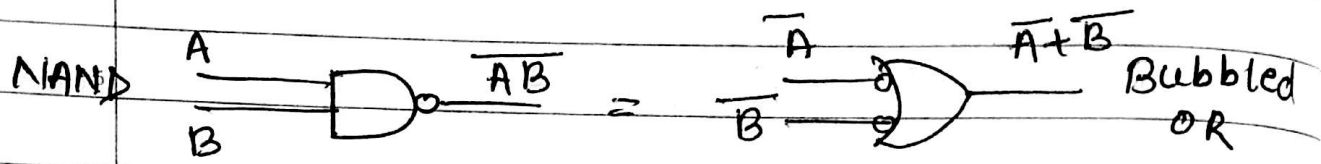
EX-OR. 

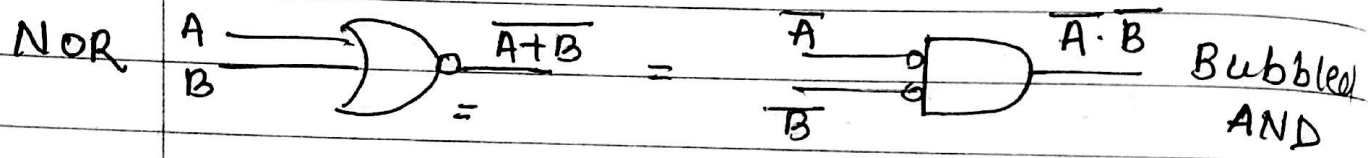
OR → 

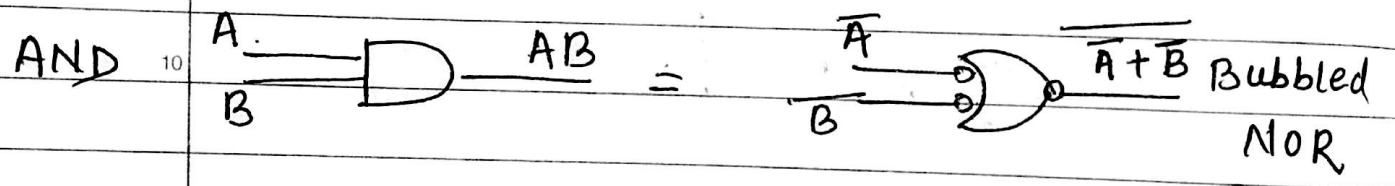
Camlin Page
Date / /

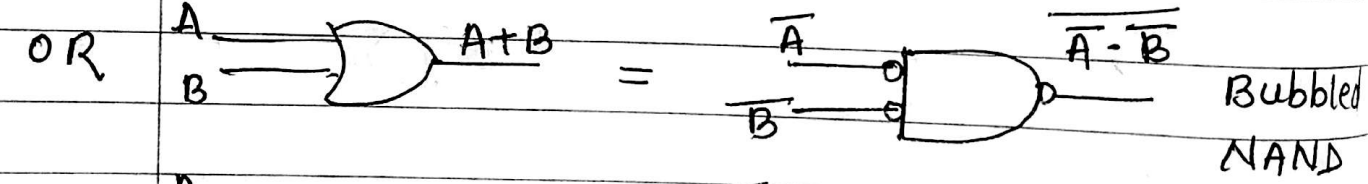
Alternate Symbol -

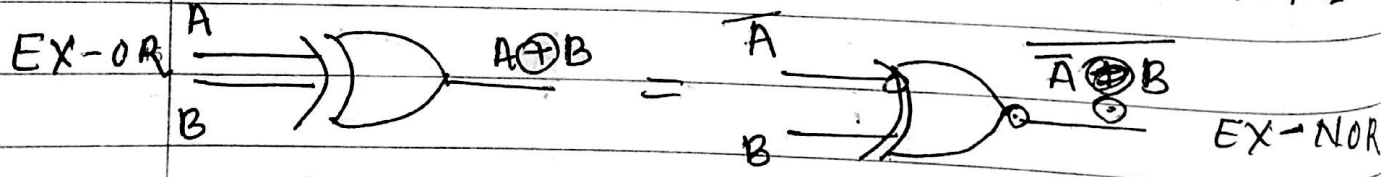


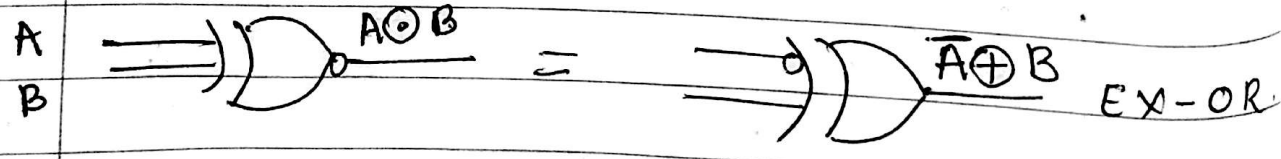












Q→ use De-Morgan's theorem to convert the boolean expression -

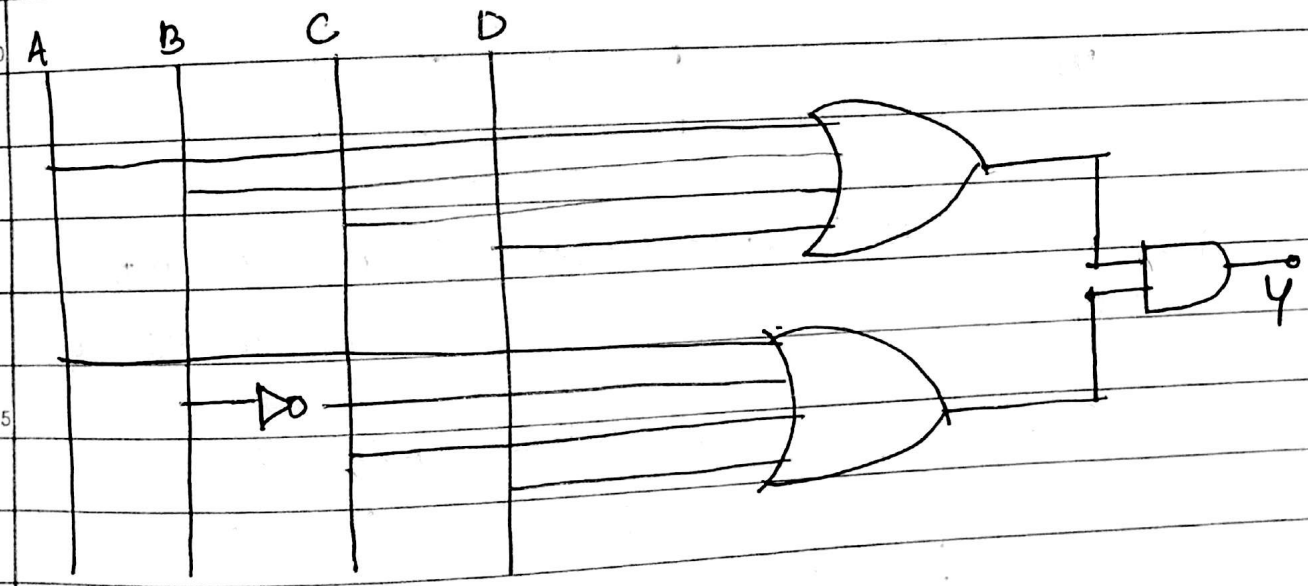
$Y = \overline{\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,B\,\overline{C}\,\overline{D}}$ to its maxterms form and draw the logic diagram.

$$Y = \left(\overline{\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,B\,\overline{C}\,\overline{D}}\right)$$

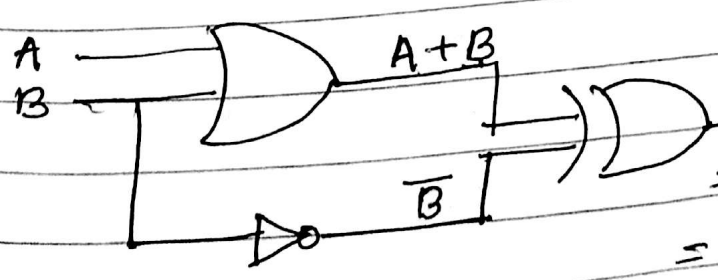
$$= \left(\overline{\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}}\right) \cdot \left(\overline{\overline{A}\,B\,\overline{C}\,\overline{D}}\right)$$

$$= \left(\overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}} + \overline{\overline{D}}\right) \cdot \left(\overline{\overline{A}} + \overline{B} + \overline{\overline{C}} + \overline{\overline{D}}\right)$$

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



Q what would be the pulse train at the O/P of the XOR gate -



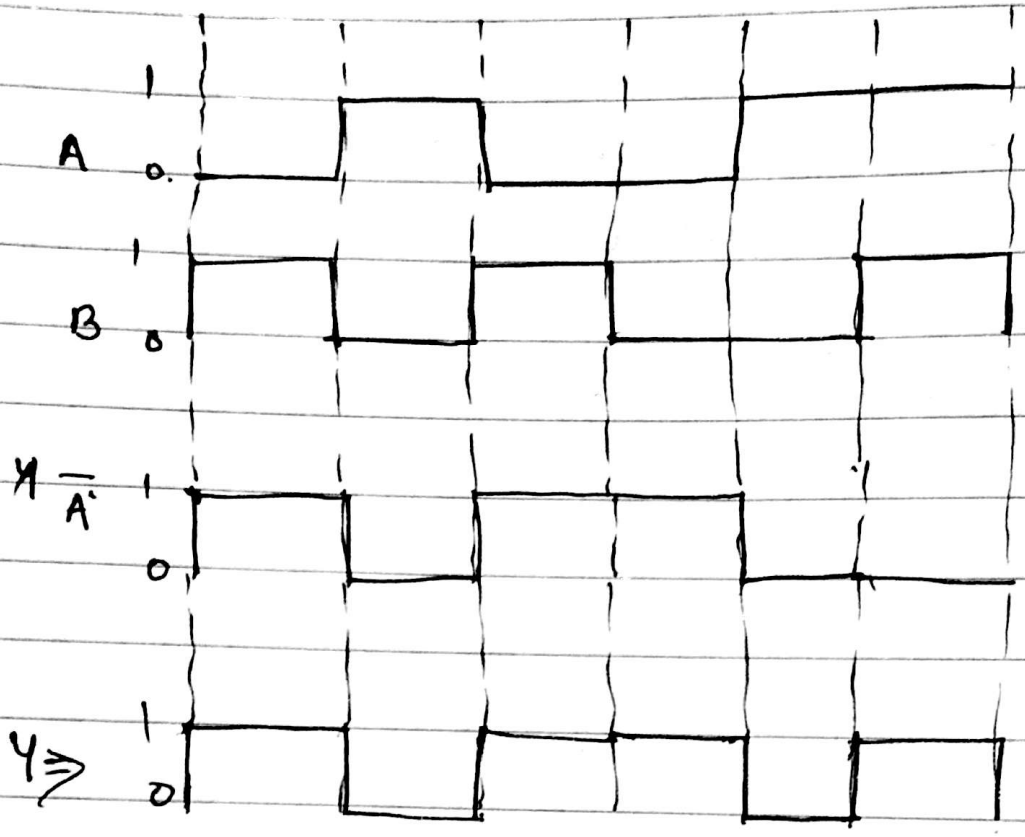
$\overline{A+B} \cdot \overline{B} + (A+B) \cdot \overline{\overline{B}}$

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

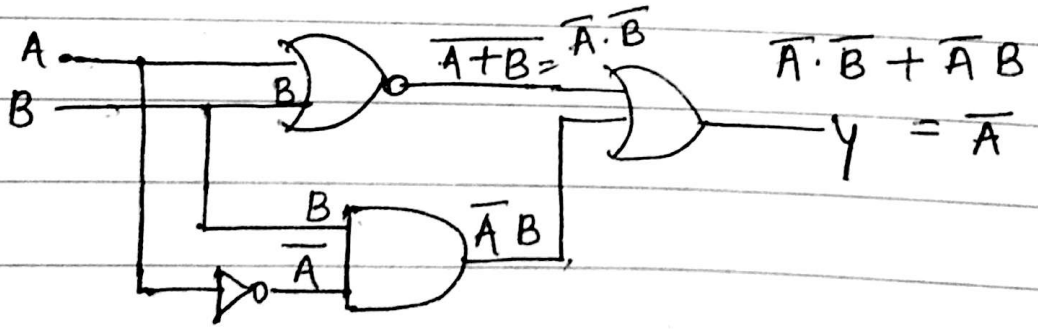
$= \overline{A}\,\overline{B} + AB + B$

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

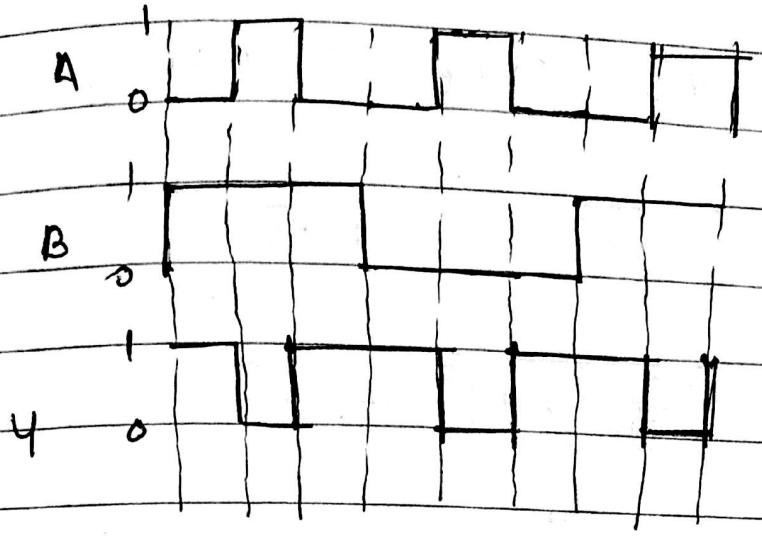
$Y = \overline{A} + B$



Q→ Write the expression for the O/P of figure ⇒, use it to determine TT.



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



$AB + A\bar{C}$

BC

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

Q Given the logical equation

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

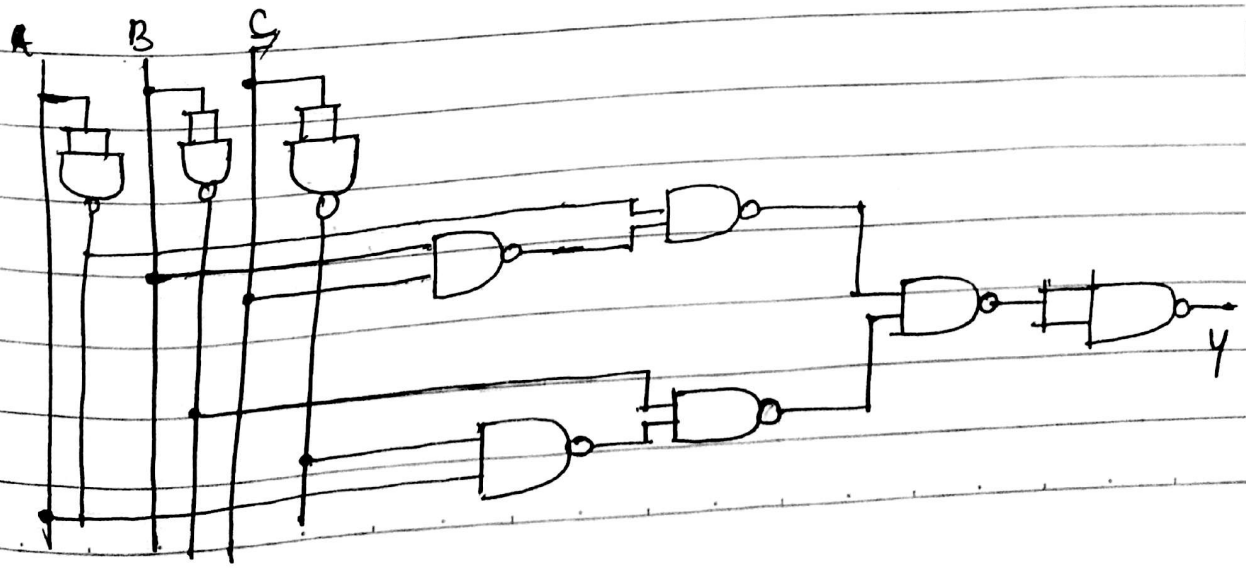
find out whether it is possible to design the ckt only one type of gate? if yes design the ckt

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

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

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

$$= \bar{A} \cdot \overline{BC} \cdot \overline{B} \cdot \overline{\bar{C}A}$$

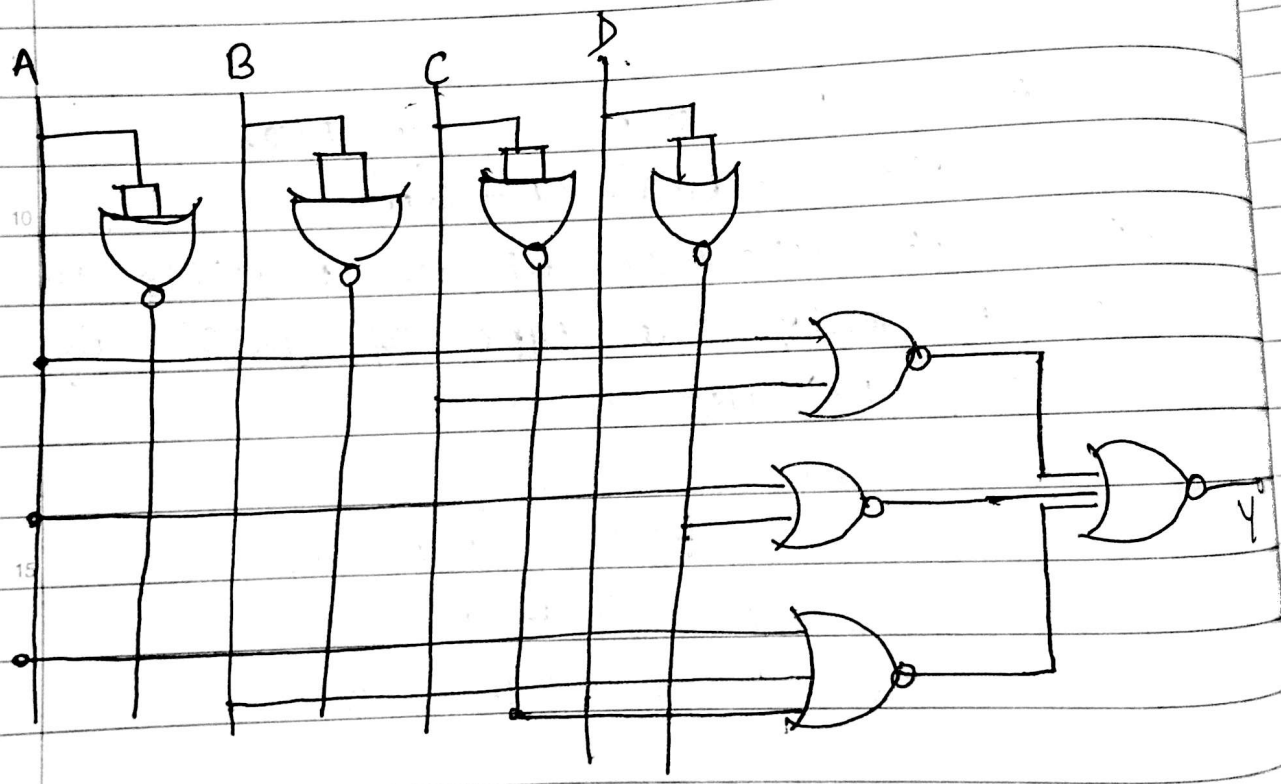


Realise:

$Y = (A+C)\cdot(A+\overline{D})\cdot(A+B+\overline{C})$ using NOR gate

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

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



Digital ckt

combinational ckt — Sequential ckt

→ present o/p depends on only present i/p.
→ NO feedbak.
→ NO memory.
→ HA, FA, Mux, Decoder

ADDERS –

A combinational ckt which performs the additions of 2 ~~binary digits~~ bits is called a half adder. while that performs the addition of 3 bits is a full adder.

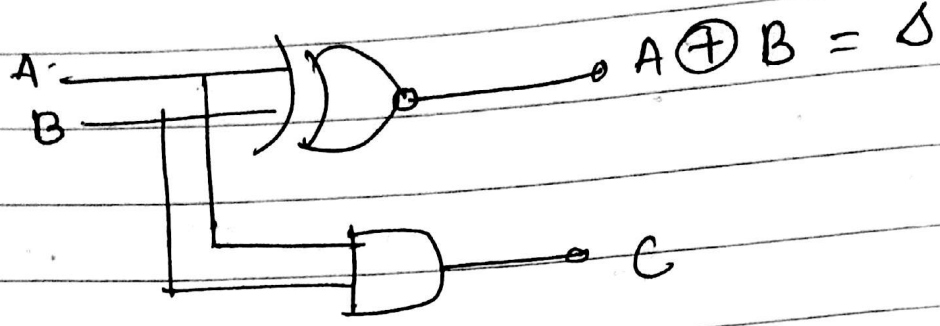
HA →

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$Sum = A'B + AB' = A \oplus B$

$carry = A \cdot B$

Logic ckt →



HA using NAND →

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

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

Full Adder -

it add three bit binary information and gives 2 o/p sum and carry.

A	B	C	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

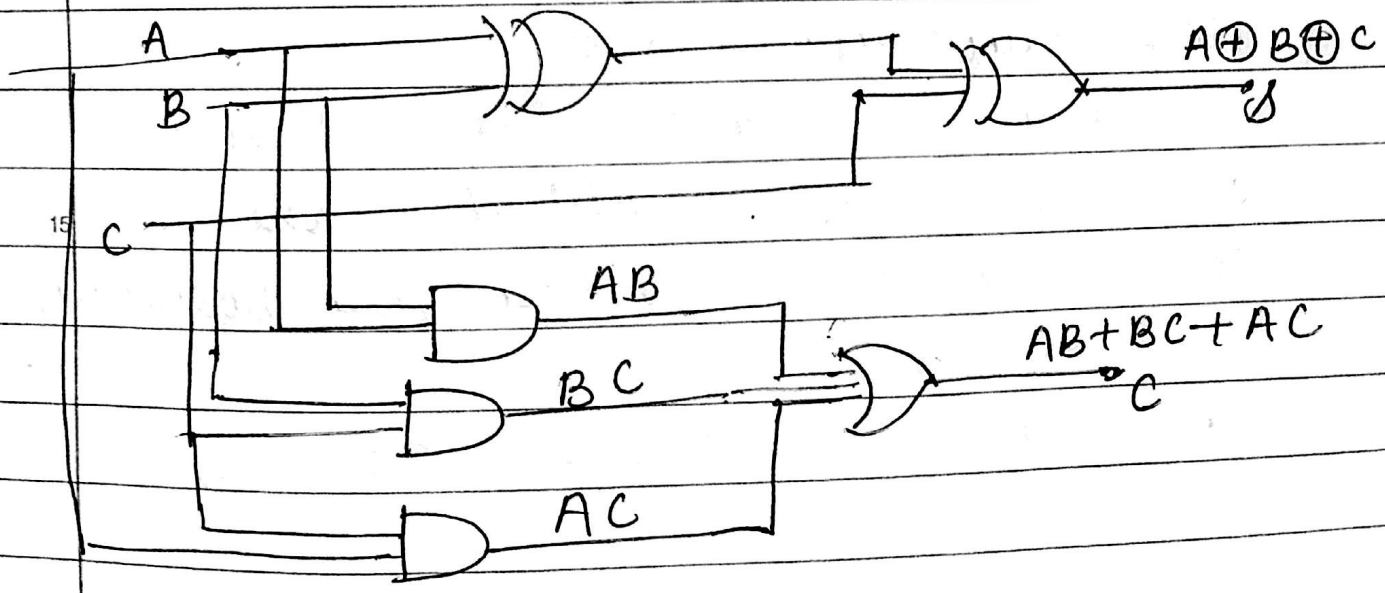
A \ BC	00	01	11	10
0	0	1	0	1
1	1	0	1	0

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

$A \oplus B \oplus C$

A \ BC	00	01	11	10
0			1	
1	1	1	1	1

Carry = $AC + AB + BC$

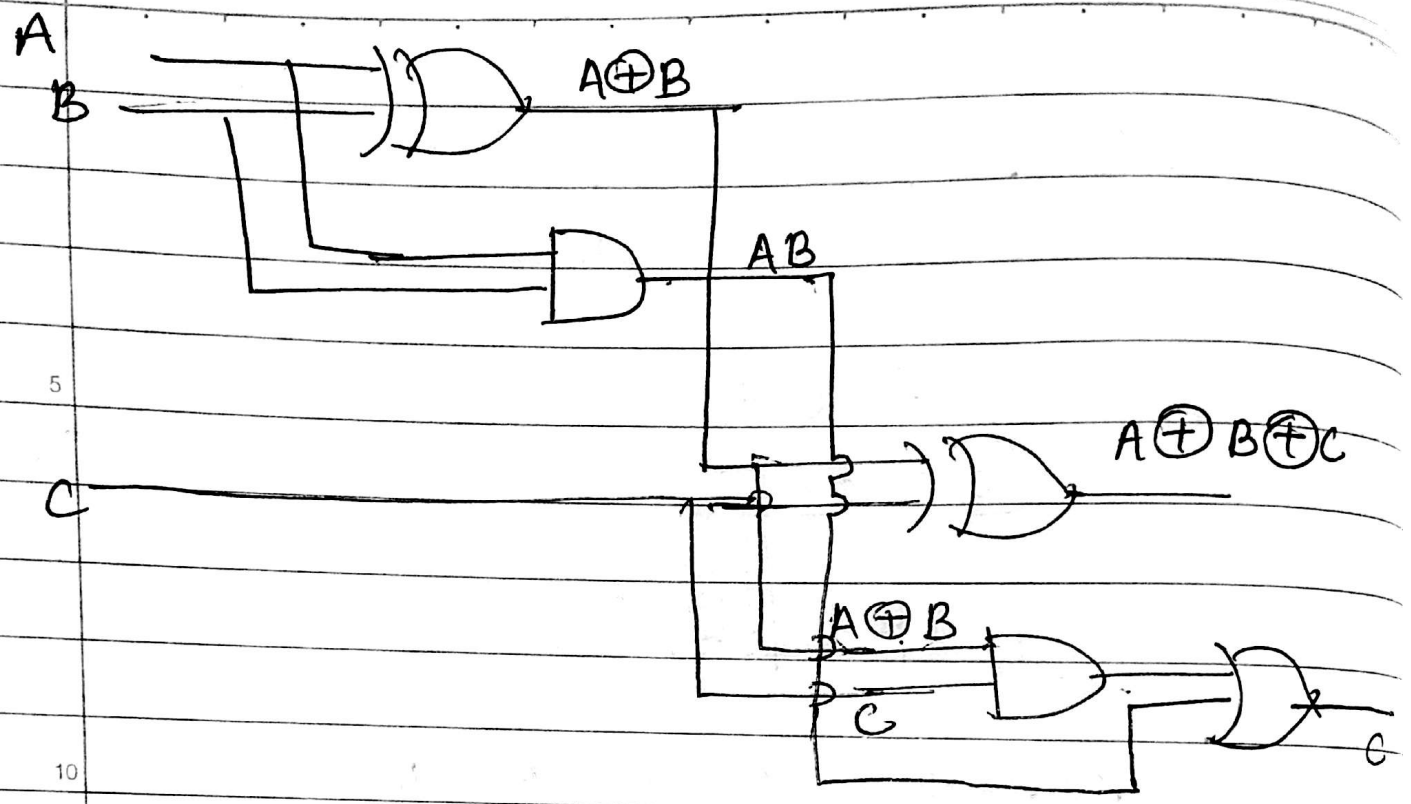


Carry = $\bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

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

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

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



2 HA & 1 OR gate → full Adder.

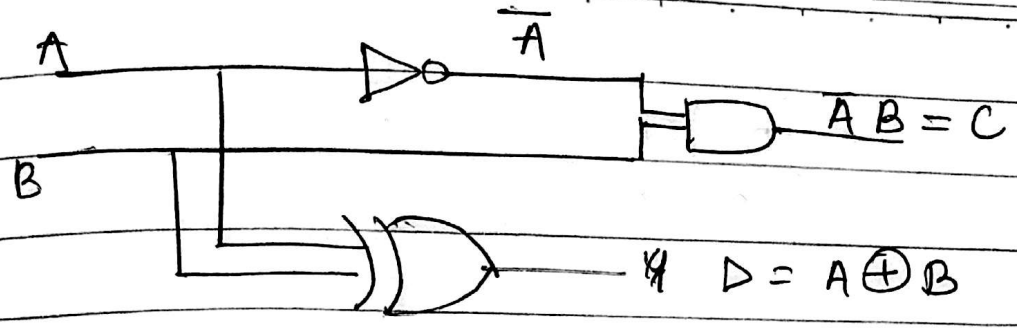
Half Subtractor –

it is a combination ckt having 2 inputs and 2 outputs, one for their difference other for borrow.

A	B	Diff.	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

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

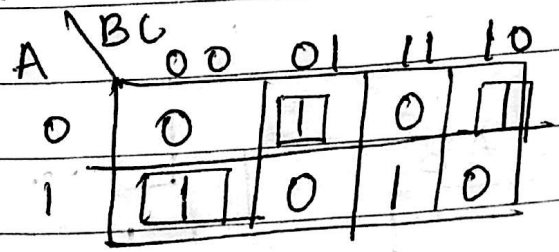
$Borrow = \bar{A}B$



Full Subtractor -

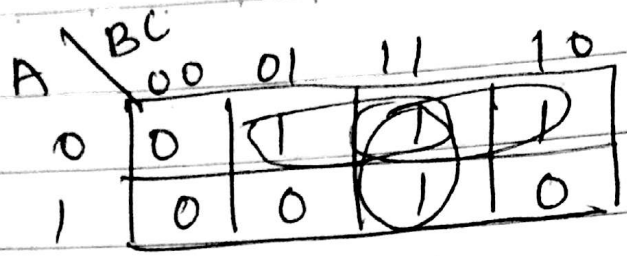
Subtract three bit binary information and produces 2 outputs, Difference & borrow.

A	B	C	Diff.	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

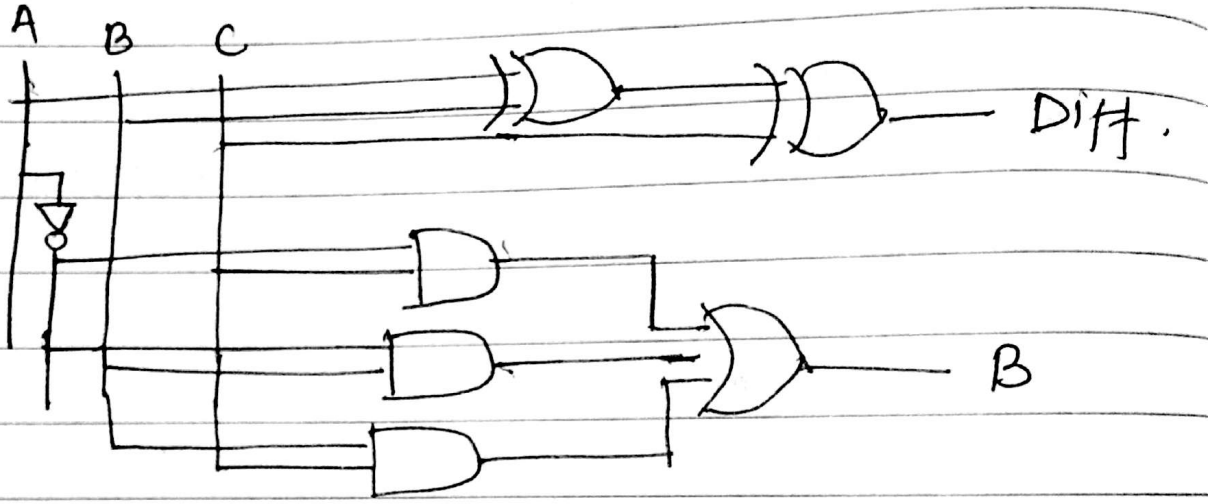


$Y = \bar{A}B'C + A'BC' + AB'C' + ABC$

$= A \oplus B \oplus C$



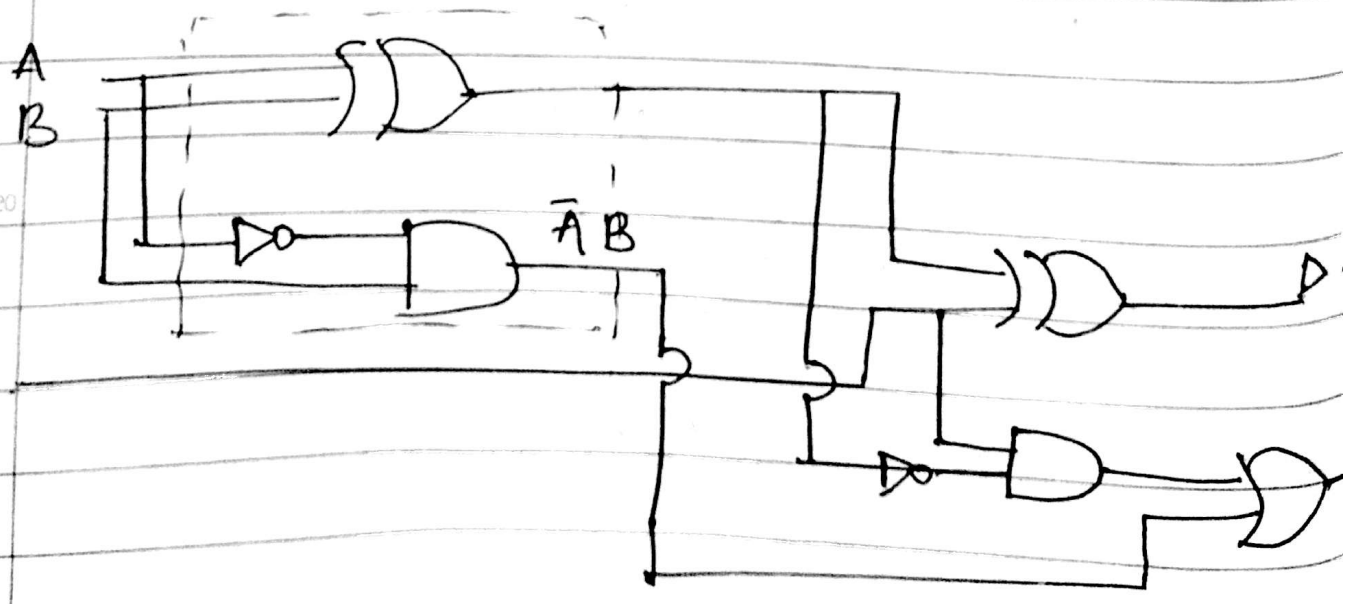
Borrow - $A'B + A'C + BC$



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

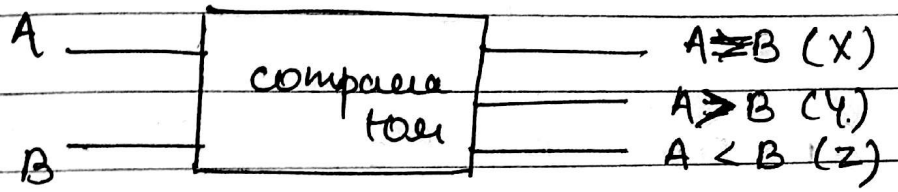
$= \bar{A}B + \bar{A}\bar{B}C + ABC$

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



Camlin | Page
Date / /

Magnitude comparator -
it is used to know whether a binary no. A is greater than, equal to, or less than another no. B.



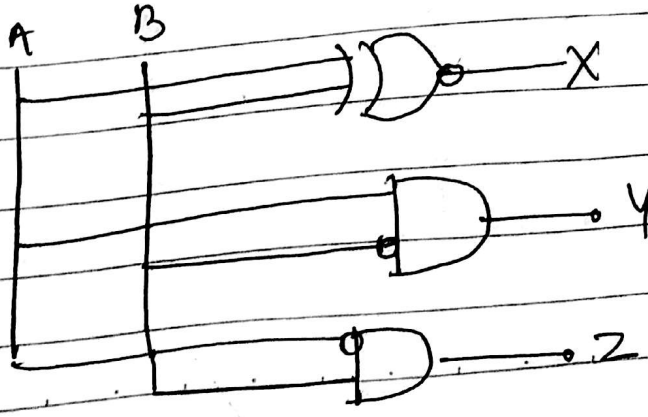
A	B	X	Y	Z
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

1 bit condition comparator → equal condition - 2
greater - 1
less - 1

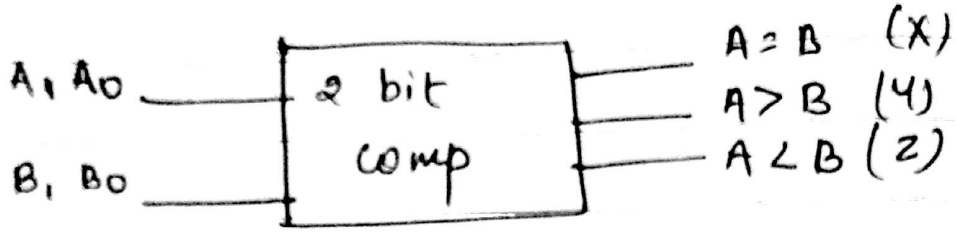
$X = \overline{A}\,\overline{B} + AB = A \odot B$

$Y = A\overline{B}$

$Z = \overline{A}B$



2 bit comparator -



A_1	A_0	B_1	B_0	X	Y	Z
0	0	0	0	1	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	0	0	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	0	1	0
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	0	1	1	0	0	1
1	1	0	0	0	1	0
1	1	0	1	0	1	0
1	1	1	0	0	1	0
1	1	1	1	1	0	0

Camlin | Page
Date / /

1 bit comparator → $2^1 = 2$ i/p line.
2 bit comparator → $2^{2^1} = 16$ i/p lines

greater condition = 6
lesser condition = 6
equal condition = 4
$\Rightarrow \frac{2^{2n} - 2^n}{2}$
$\Rightarrow 2^2 = 2^n$

2 bit

$X = (A_1 \odot B_1) \cdot (A_0 \odot B_0)$

$Y = (A_1 \overline{B_1}) + (A_1 \odot B_1) \cdot (A_0 \overline{B_0})$

$Z = (\overline{A_1} B) + (A_1 \odot B_1) \cdot \overline{A_0} B_0$

3 bit

$X = (A_2 \odot B_2) \cdot (A_1 \odot B_1) \cdot (A_0 \odot B_0)$

$Y = A_2 \overline{B_2} + (A_2 \odot B_2) A_1 \overline{B_1} + (A_2 \odot B_2) \cdot (A_1 \odot B_1) \cdot A_0 \overline{B_0}$

$Z = \overline{A_2} B_2 + (A_2 \odot B_2) \overline{A_2} B_1 + (A_2 \odot B_2) \cdot (A_1 \odot B_1) \overline{A_0} \overline{B_0}$

4 bit

$X = (A_3 \odot B_3) \cdot (A_2 \odot B_2) \cdot (A_1 \odot B_1) (A_0 \odot B_0)$

$Y = A_3 \overline{B_3} + (A_3 \odot B_3) \cdot A_2 \overline{B_2} + (A_3 \odot B_3) \cdot (A_2 \odot B_2) \cdot A_1 \overline{B_1} + (A_3 \odot B_3) (A_2 \odot B_2) (A_1 \odot B_1) A_0 \overline{B_0}$

$Z =$

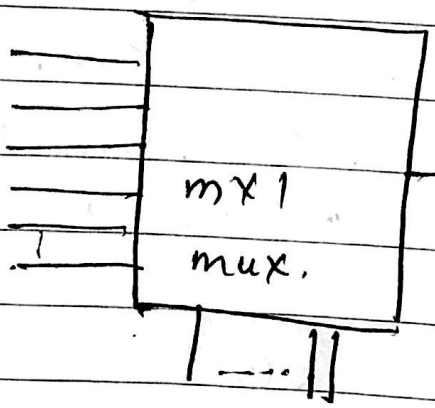
Multiplexers –

it is a combinational ckt which select binary inf. ∅ from one of the many i/p lines and directs it to a single o/p line, thats why it is called a data selector.

Selection line – n

i/p lines – 2^n

Selection lines bit combination determine which i/p is selected.



Block diagram

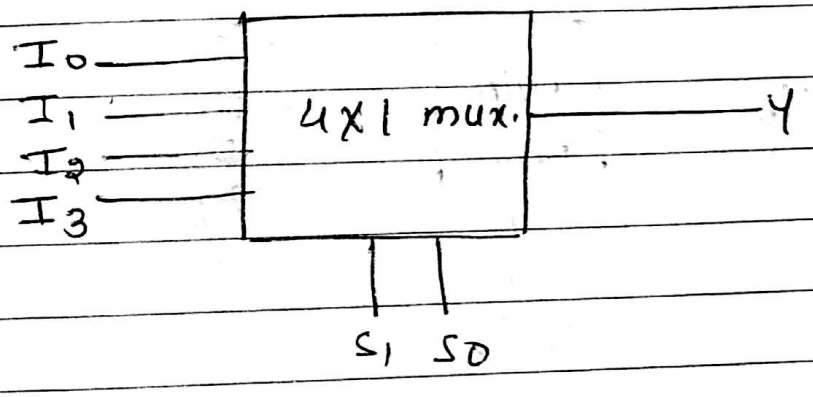
n = 2

i/p = 4

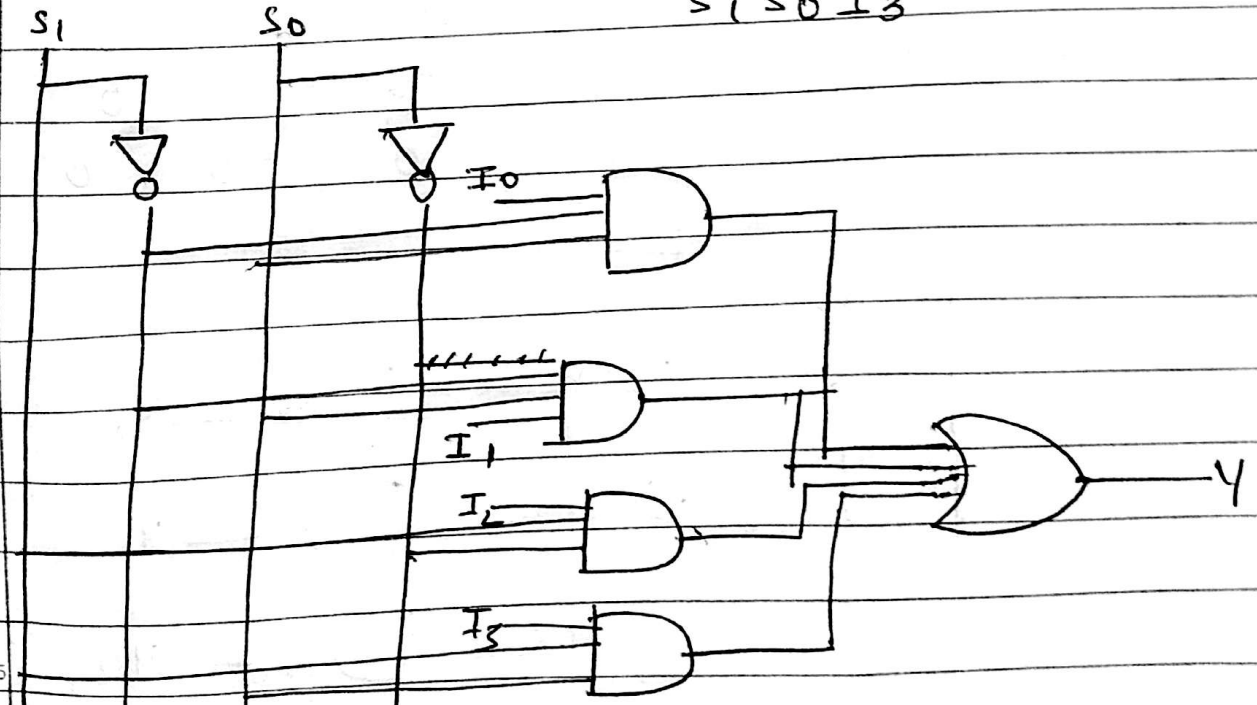
Camlin Page
Date / /

TT for 4x1 Mux—

S_1	S_0	O/P
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



$Y = \overline{S_1}\,\overline{S_0}\,I_0 + \overline{S_1}\,S_0 I_1 + S_1\,\overline{S_0} I_2 + S_1 S_0 I_3$



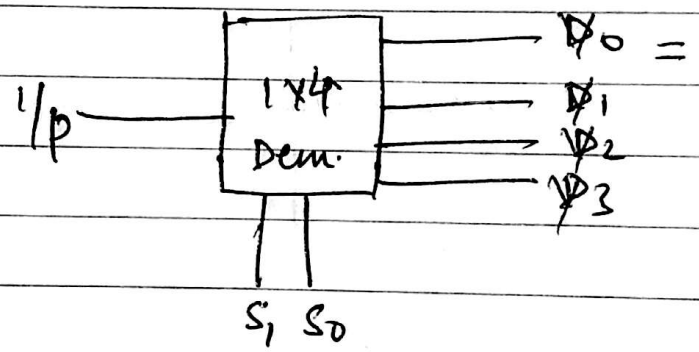
Logic diagram

Camlin Page
Date / /

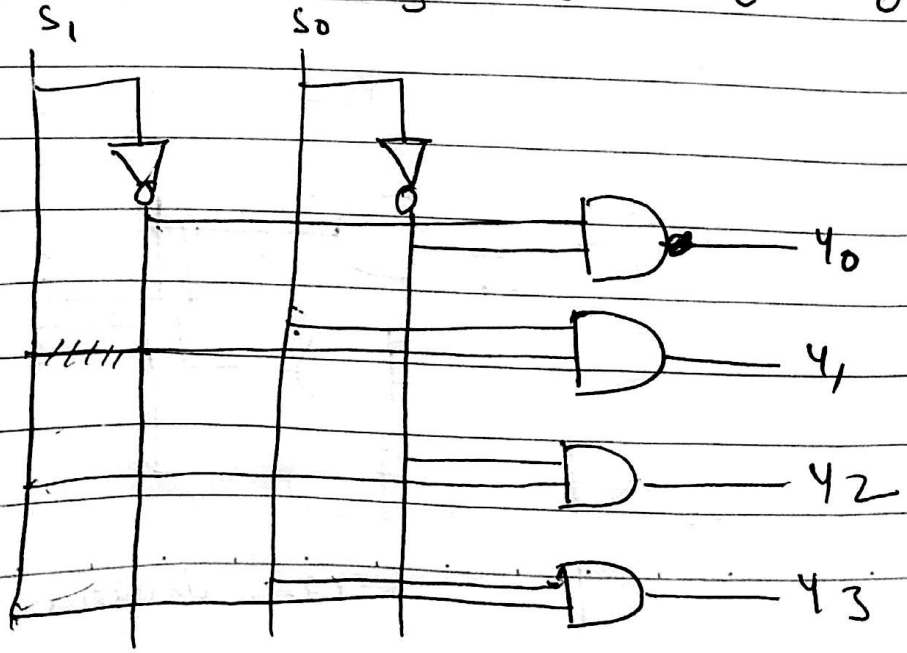
Demultiplexer -

which transmit single i/p inf. to one of the many o/p line selected by select lines is called demultiplexer.

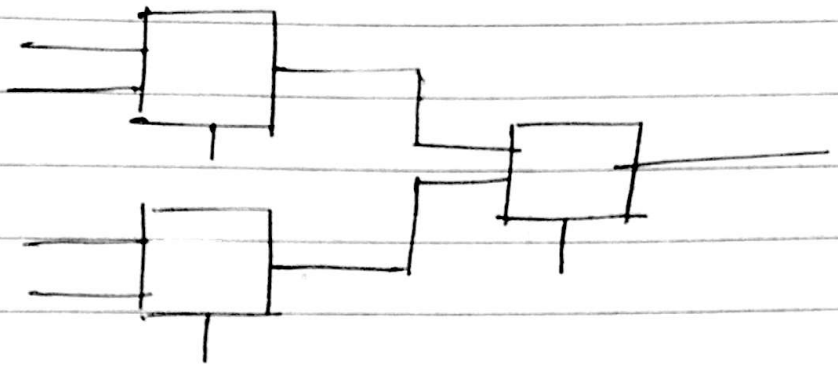
1 x 4 Demux -



Y	S_1	S_0	Y_3	Y_2	Y_1	Y_0
Y_0	0	0	0	0	0	I_0
Y_1	0	1	0	0	I_1	0
Y_2	1	0	0	I_2	0	0
Y_3	1	1	I_3	0	0	0



4 X 1 $\xrightarrow{3}$ 2 X 1 mux



8 X 1 $\xrightarrow{7}$ 2 X 1

256 X 1 $\xrightarrow{255}$ 2 X 1

2^n X 1 $\xrightarrow{2^n - 1}$ 2 X 1

16 X 1 $\xrightarrow[\frac{16}{4} = 4+1]{5}$ 4 X 1

20+1 = 21

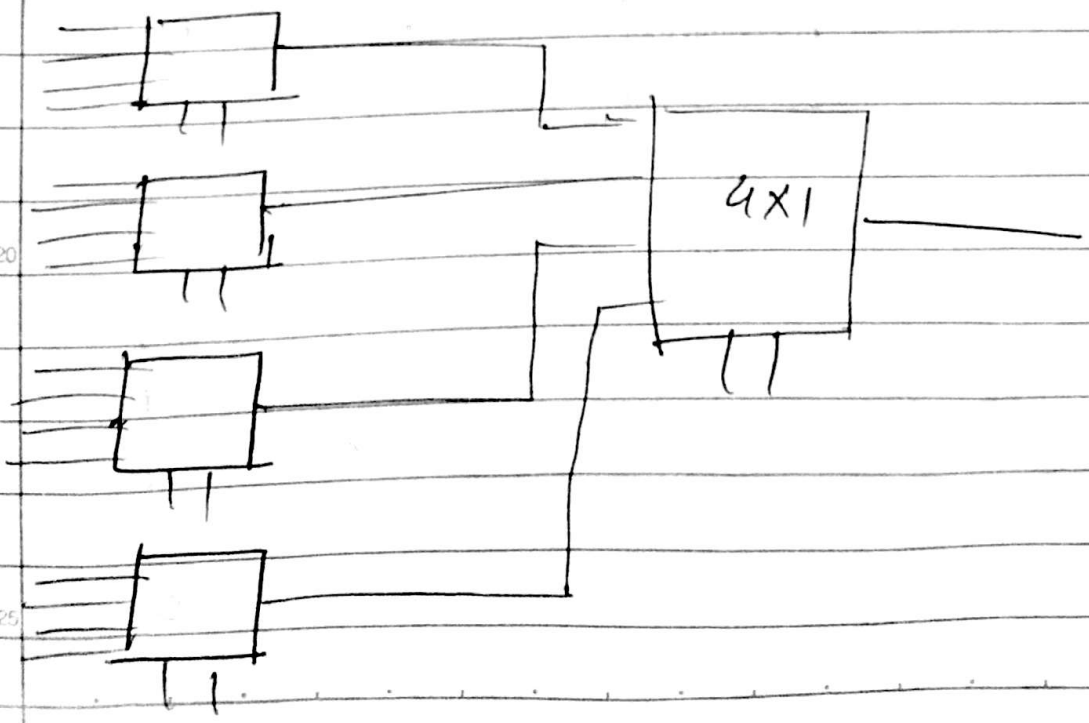
64 X 1 $\longrightarrow$ 4 X 1

$\frac{64}{4} = 16 + 4 + 1$

64 X 1 $\xrightarrow[8+1]{}$ 8 X 1

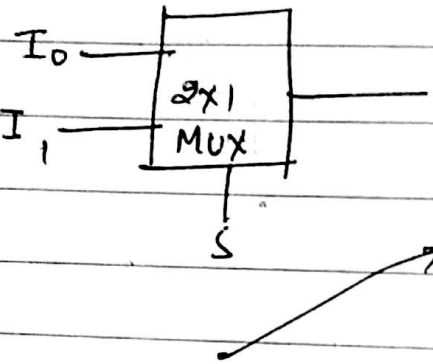
64 X 1 $\xrightarrow[4+1]{}$ 16 X 1

256 X 1 $\xrightarrow[16+1]{}$ 16 X 1



Mux as universal logic gate -

NOT →



	I_0	I_1
$\bar{B}$	0	1
B	2	3
	0	B

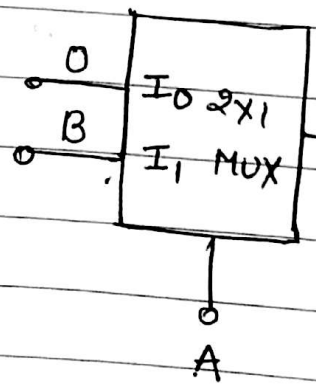
AND →

B	Y
0	0
1	A

A	Y
0	0
1	B

A	B	Y	
0	0	0	0
0	1	0	1
1	0	0	2
1	1	1	3

S=0 (A)	I_0	I_1
$\bar{A}$	0	1
A	?	3
	0	A



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

$= S$

$= \bar{A} \cdot 0 + AB$

$= AB$

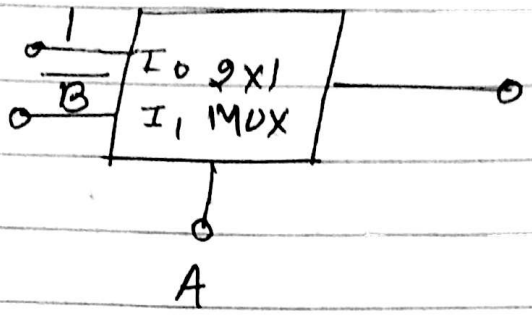
NAND →

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

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

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

	I_0	I_1
$\overline{A}$	0	1
A	1	B

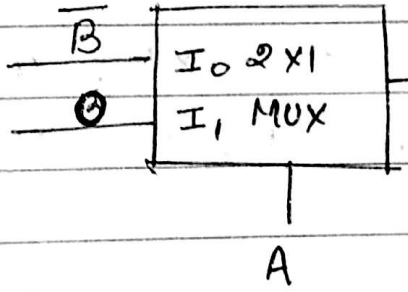


$$Y = \overline{A} + A\overline{B}$$
$$= \overline{A} + \overline{B}$$
$$= \overline{A \cdot B}$$

NOR-

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

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

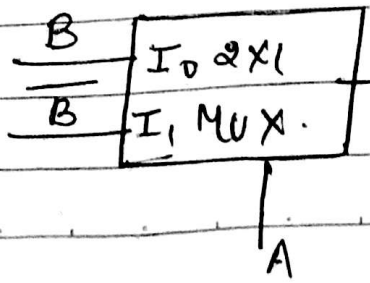


$$Y = \overline{A}\,\overline{B} + A \cdot 0$$
$$= \overline{A}\,\overline{B}$$
$$= \overline{A+B}$$

EX-OR

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

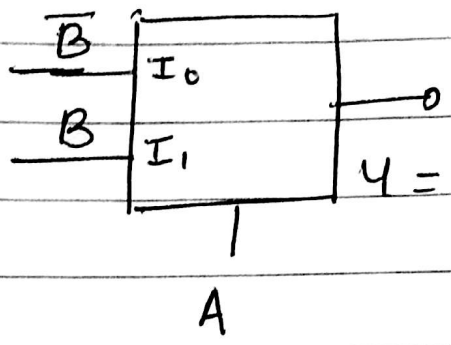
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0



$$Y = \overline{A}B + A\overline{B}$$
$$= A \oplus B$$

EX-NOR -

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



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

$$Y = \overline{A}\,\overline{B} + AB$$

$$= A \odot B$$

Q10 Implement the following function using MUX

$$f(x,y,z) = \Sigma(1,2,6,7)$$

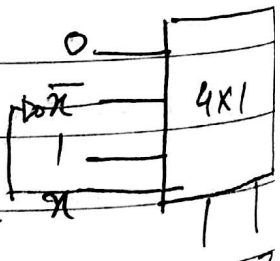
1) y z as select line.

$n = 3$

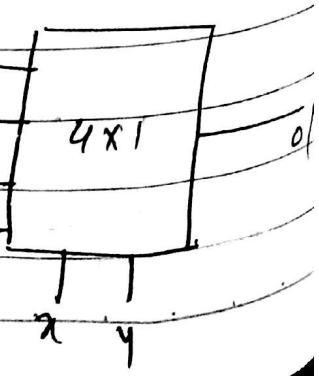
Mux required – $2^{3-1} \times 1 = 4 \times 1$

least significant y, z → select lin.

	I_0	I_1	I_2	I_3
$\overline{x}$	0	(1)	(2)	3
x	4	5	(6)	(7)
	0	$\overline{x}$	1	x



	I_0	I_1	I_2	I_3	
$\overline{z}$	0	(2)	4	(6)	$\overline{z}$
z	(1)	3	5	(7)	0
	z	$\overline{z}$	0	1	1



Camlin | Page
Date / /

4 o/p = $\bar{y}\bar{z}x + \bar{x}\bar{y}z + xy\bar{z} + xyz$

$= \bar{x}\bar{y}\bar{z} + x\bar{y}\bar{z} + \bar{x}\bar{y}z + xy\bar{z} + xyz + \bar{x}yz$

$= 0 \quad 4 \quad 1 \quad 6 \quad 7 \quad 2$

o/p = $\bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xy$

$= \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + x\bar{y}z + xy\bar{z} + xyz$

$= 001 \quad 010 \quad 100 \quad 101 \quad 110 \quad 111$

$= 1 \quad 2 \quad 6 \quad 7$

	I_0	I_1	I_2	I_3
$\bar{z}$	0	①	②	3
z	4	5	⑥	⑦
	0	$\bar{z}$	1	z

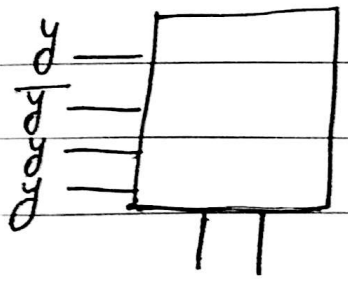
$y = \bar{x}y\bar{z} + x\bar{y}(z+\bar{z}) + xyz$

$= \bar{x}y\bar{z} + x\bar{y}z + x\bar{y}\bar{z} + xyz$

$= 010 \quad 101 \quad 100 \quad 111$

$= 2 \quad 5 \quad 4 \quad 7$

	I_0	I_1	I_2	I_3
$\bar{y}$	0	①	4	5
y	②	3	⑥	⑦
	y	$\bar{y}$	y	y



o/p = $\bar{x}y\bar{z} + \bar{x}\bar{y}z + xy\bar{z} + xyz$

$= 010 \quad 001 \quad 110 \quad 111$

$= 2 \quad 1 \quad 6 \quad 7$

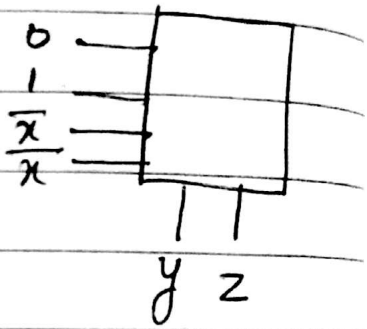
Q Implement the following function using M

$$f(x,y,z) = \pi(0,4,6,7)$$

$2^{3-1} \times 1$ MUX = 4×1 MUX.

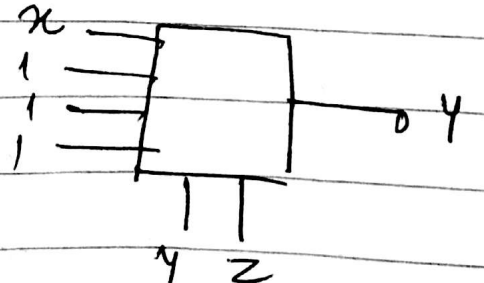
$$f(x,y,z) = \Sigma(1,2,3,5)$$

	I_0	I_1	I_2	I_3
$\bar{x}$	0	1	2	3
x	4	5	6	7
	0	1	$\bar{x}$	$\bar{x}$



$$o/p = \bar{y}\bar{z}x0 + \bar{y}z(x+\bar{x}) + \bar{x}y\bar{z} + \bar{x}yz$$
$$= x\bar{y}z + \bar{x}\bar{y}z + \bar{x}y\bar{z} + \bar{x}yz$$
$$101 \quad 001 \quad 010 \quad 011$$
$$5 \quad 1 \quad 2 \quad 3$$

Q The combinational logic ckt shown has an o/p Y find o/p Y(x,y,z)



	I_0	I_1	I_2	I_3
$\bar{x}$	0	1	2	3
x	4	5	6	7
	x	1	1	1

$y(x, 4, 2) = \Sigma (1, 2, 3, 4, 5, 6, 7)$

Q⇒ 4X1 - convert 4X1 MUX to 16X1 MUX.

$\frac{16}{4} + 1 = 5$

16X1 ⟶ 4X1 (4+1)

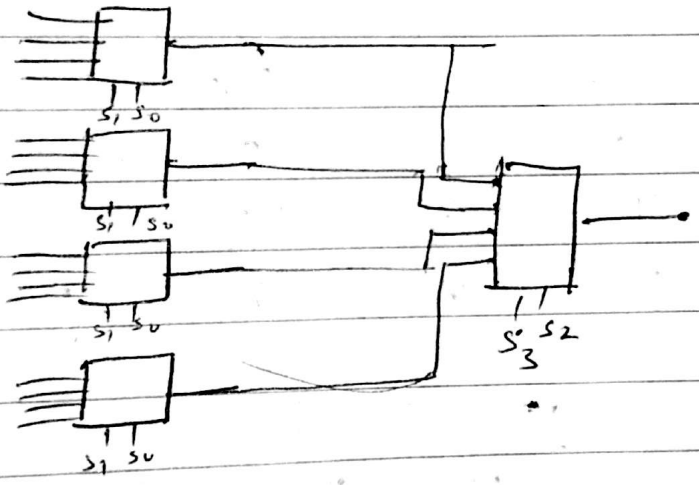
64X1 ⟶ 4X1 (16+4+1)

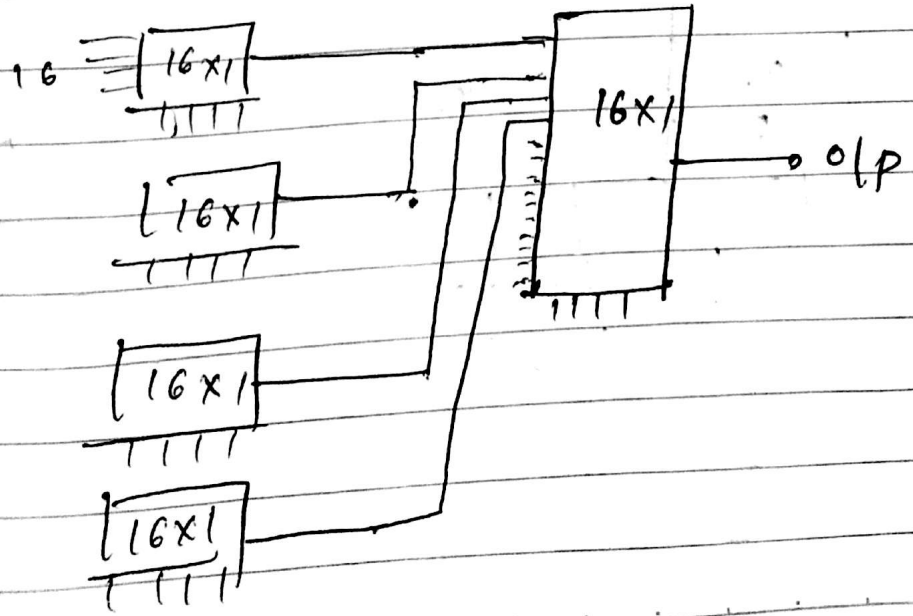
64X1 ⟶ 16X1

$\frac{64}{16} = 4 + 1 = 5$

64X1 ⟶ 8X1 (8+1)

256X1 ⟶ 16X1 (16+1)





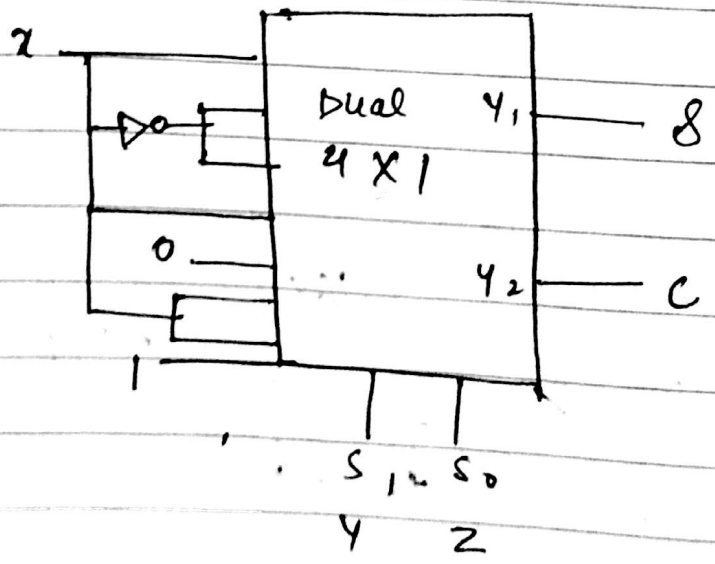
Implement full adder using MUX -

$S(x,y,z) = \Sigma m(1,2,4,7)$

$C(x,y,z) = \Sigma m(3,5,6,7)$

Sum	I_0	I_1	I_2	I_3
$\bar{x}$	0	(1)	(2)	3
x	(4)	5	6	(7)
	x	$\bar{x}$	$\bar{x}$	x

carry	I_0	I_1	I_2	I_3
	0	1	2	(3)
	4	(5)	(6)	(7)
	0	x	x	1



Implement using all 4x1 mux.

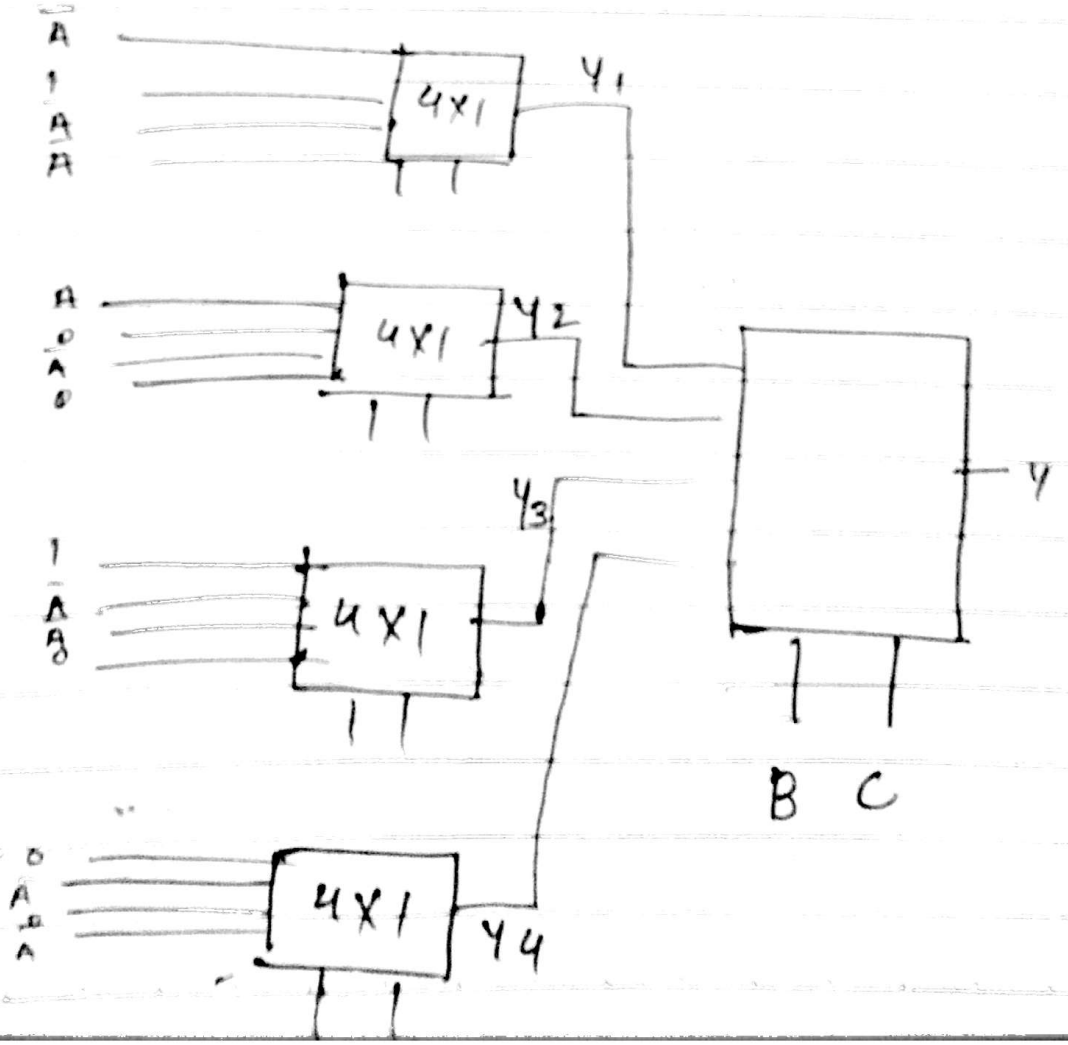
$f(A,B,C,D,E) = \Sigma m(0,1,2,3,6,8,9,10,13,15,17,20,24)$

$n = 5$

Mux $\longrightarrow 2^{5-1} =$ 16x1 MUX.

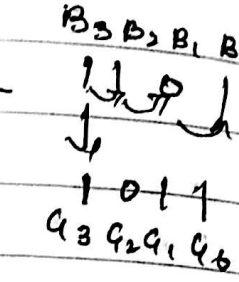
16x1 $\xrightarrow{5}$ 4x1

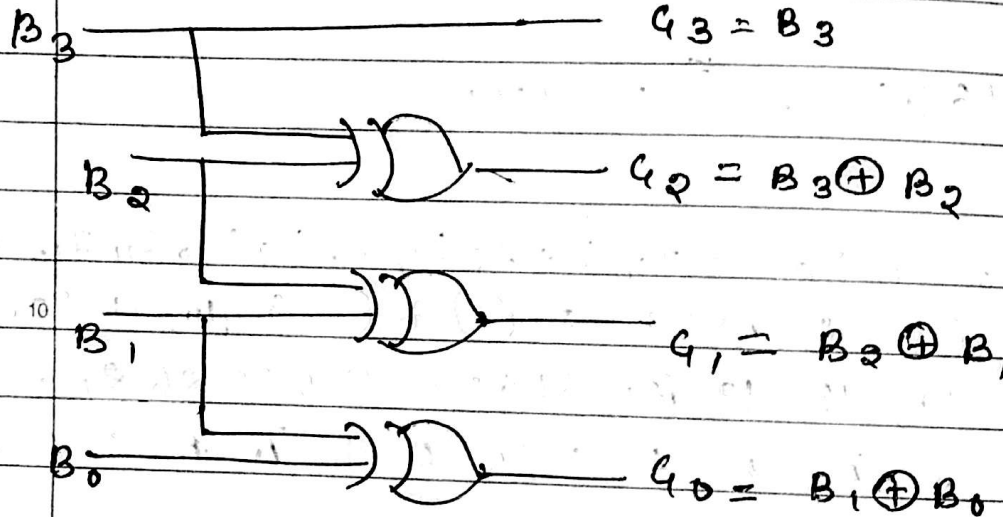
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
$\bar{A}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	$\bar{A}$	1	$\bar{A}$	$\bar{A}$	A	0	$\bar{A}$	0	1	$\bar{A}$	$\bar{A}$	0	0	$\bar{A}$	0	$\bar{A}$



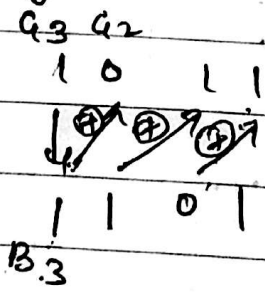
CODE CONVERSION–

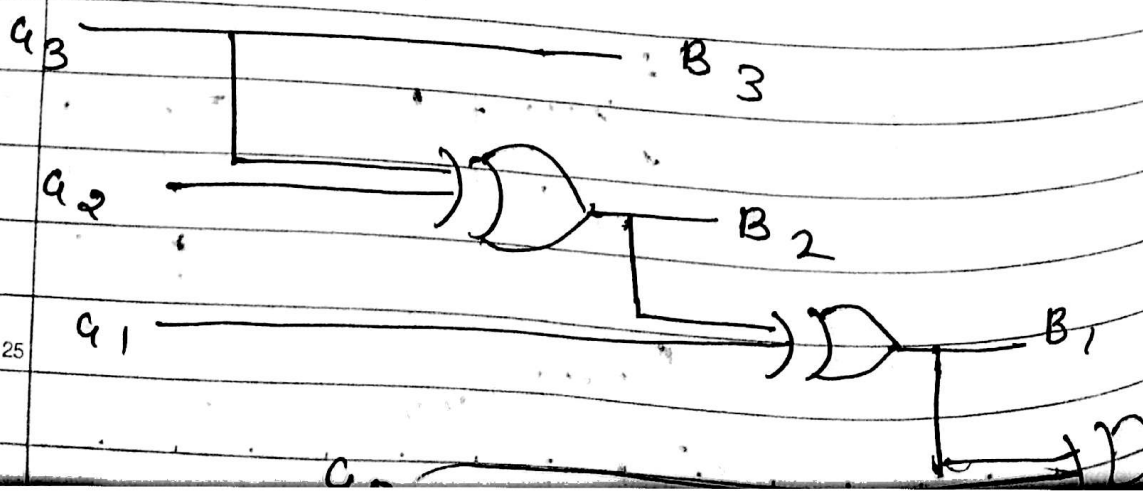
Binary to Gray convertar –





Gray to Binary –

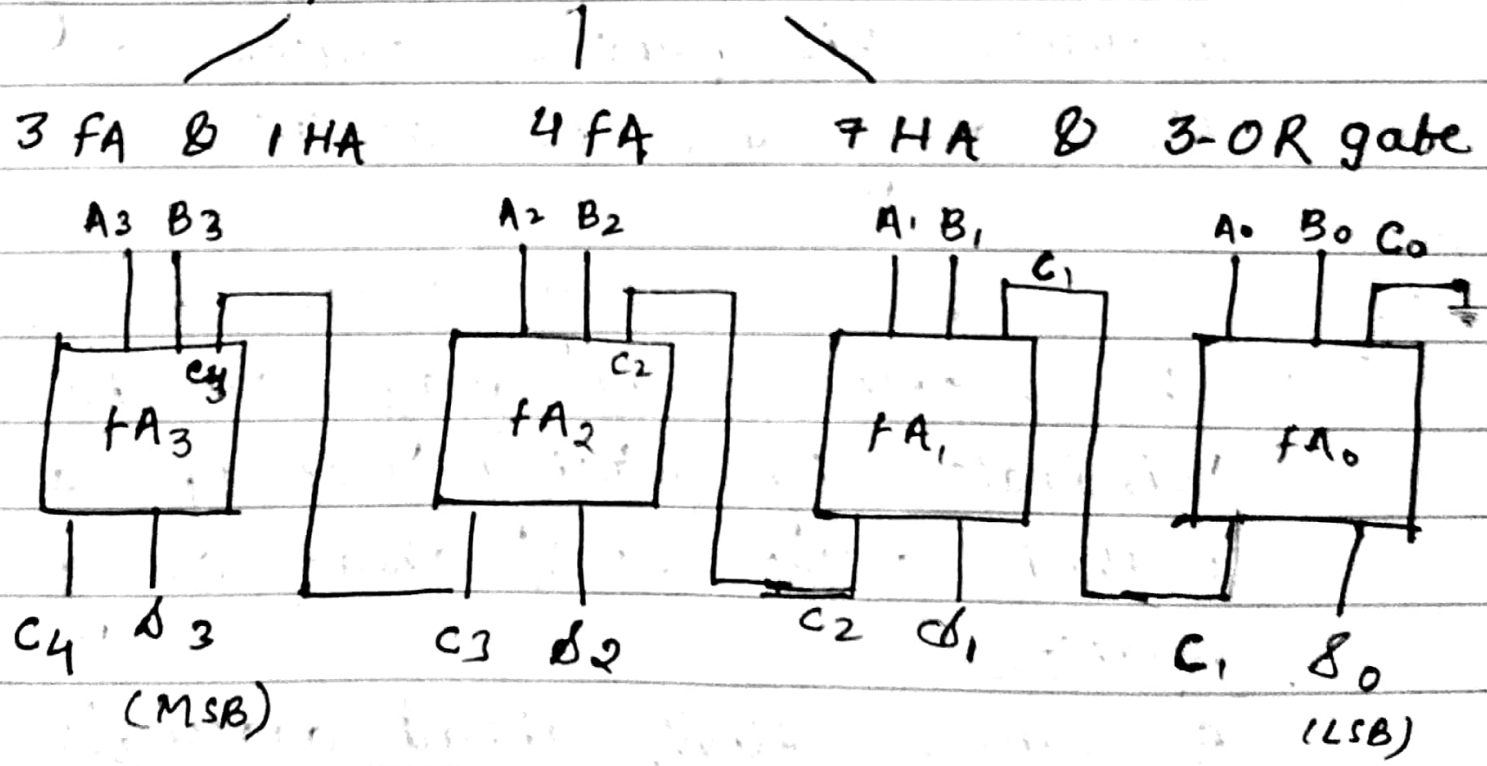




parallel Binary Adders -

A single full adder is capable of adding two 1-bit numbers and an i/p carry. If we need to add binary numbers with more than one bit we require additional full adders. for the addition of 2-bit numbers we need two full adders. Similarly for the addition of 4-bit numbers we need 4 full adders and so on.

Four-bit parallel Adder -



it consist of 4-bit full adders connected together.

right most full adder whereas the higher order bits are applied as shown to the successive higher order adders.

→ The MSB of the 2 no.s are applied to the left most full adder.

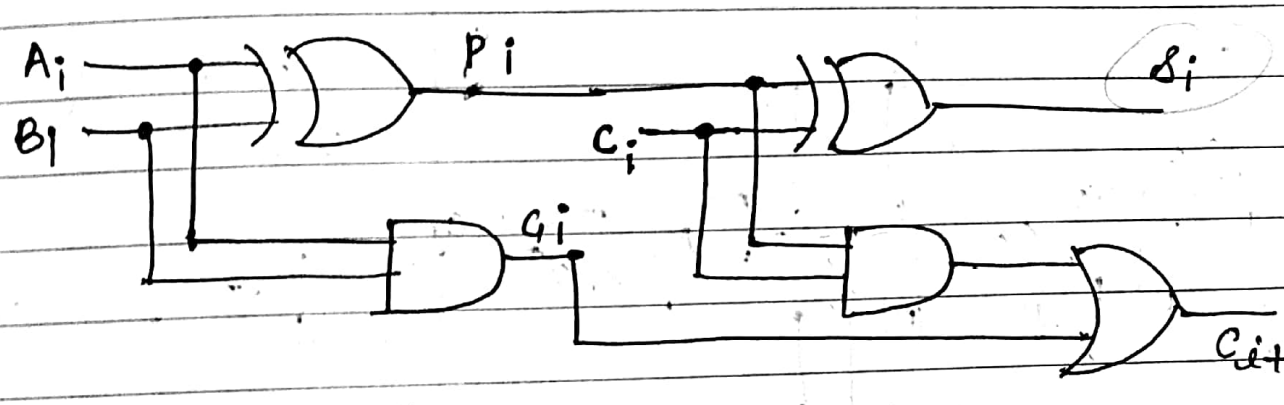
→ It is also known as ripple carry adder. In this the carry o/p of each full adder is connected to the carry i/p of the next higher order stage. The sum and the o/p carry of any stage cannot be produced untill the i/p carry occurs. This process causes a time delay in the addition.

LOOK ahead carry Adder -

→ The ripple carry delay can be eliminated by making use of a method called look ahead carry addition.

→ A carry look-ahead adder is a high speed adder which follows a special strategy for quick carry generation at each stage without waiting for the carries of the previous stages.

Let us consider a full adder ckt in block diagram.



$C_0 \rightarrow$ i/p carry

for stage 'i' →

$$P_i = A_i \oplus B_i$$
$$G_i = A_i \cdot B_i$$
$$S_i = P_i \oplus C_i = A_i \oplus B_i \oplus C_i$$
$$C_{i+1} = G_i + P_i C_i$$

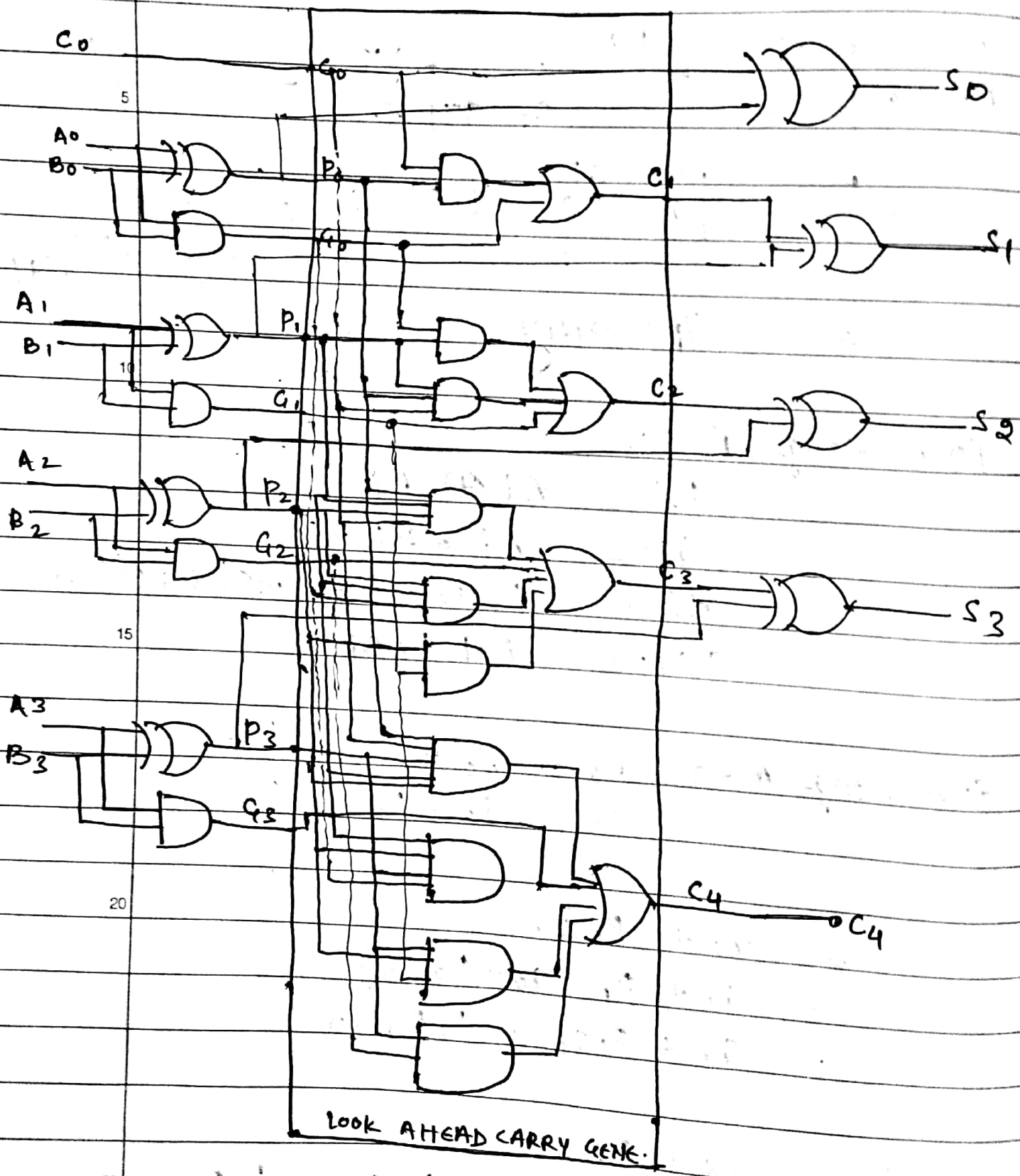
for stage '0' → $S_0 = P_0 \oplus C_0$

$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1$$
$$= G_1 + P_1 (G_0 + P_0 C_0)$$
$$= G_1 + P_1 G_0 + P_1 P_0 C_0$$

$$C_3 = G_2 + P_2 C_2$$
$$= G_2 + P_2 (G_1 + P_1 G_0 + P_1 P_0 C_0)$$
$$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$

$$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$$



→ Look Ahead carry adder is fastest adder.

Gi → carry generating term

Pi → carry propogation term

→ The G variable can be generated directly from A and B i/ps using AND gates.

→ The P variable are obtained directly from A & B i/ps using EX-OR gates.

carry.

→ Each ∧ function can be implemented with one level of AND gates followed by an OR gate.

→ Each o/p sum requires 2 EX-OR gates.

The o/p of 1st EX-OR gate generates Pi variable and the AND gate generates the Gi variable. The carries are propogated through the carry look ahead generator and applied as i/ps to the second EX-OR gate.

→ No. of 2 AND gate used in look ahead carry n/w — $\frac{n(n+1)}{2}$

→ No. of OR gate — n

Camlin Page
Date / /

BCD Adder -

A BCD adder is a ckt that adds two BCD digits in parallel, using ordinary binary addition and produces a sum digits in BCD.

→ A BCD adder ckt must be able to perform the following -

1) Add 2 4-bit BCD code, using ordinary binary addition.
2) Determine if the sum of this addition is greater than 9, if it is, then add 6 (0110) to this sum and generate a carry to the next significant decimal position.

If The circuitry of a BCD adder must include the correction logic in its internal construction, so that the correction can be added in.

From table C can be expressed as - $C = K + Z_8 (Z_4 + Z_2)$

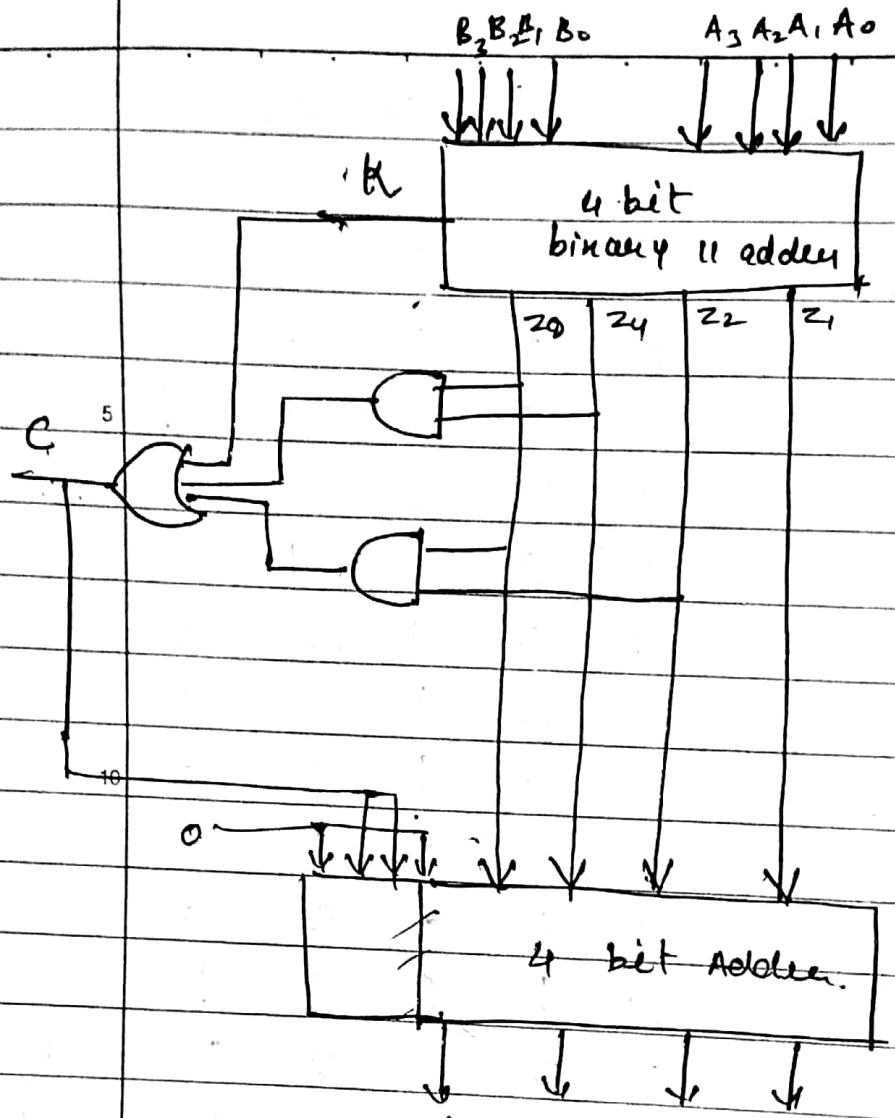
Camlin | Page
Date / /

Binary Sum

K	Z_8	Z_4	Z_2	Z_1
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1
0	1	0	0	0
0	1	0	0	1
0	1	0	1	0
0	1	0	1	1
0	1	1	0	0
0	1	1	0	1
0	1	1	1	0
0	1	1	1	1
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1

BCD Sum

C	S_8	S_4	S_2	S_1	
0	0	0	0	0	
0	0	0	0	1	
0	0	0	1	0	
0	0	0	1	1	
0	0	1	0	0	
0	0	1	0	1	
0	0	1	1	0	
0	0	1	1	1	
0	1	0	0	0	
0	1	0	0	1	
1	0	0	0	0	10
1	0	0	0	1	11
1	0	0	1	0	12
1	0	0	1	1	13
1	0	1	0	0	14
1	0	1	0	1	15
1	0	1	1	0	16
1	0	1	1	1	17
1	1	0	0	0	18
1	1	0	0	1	19



Decoders -

Decoder is a combinational ckt which convert bits that accepts a binary i/ps

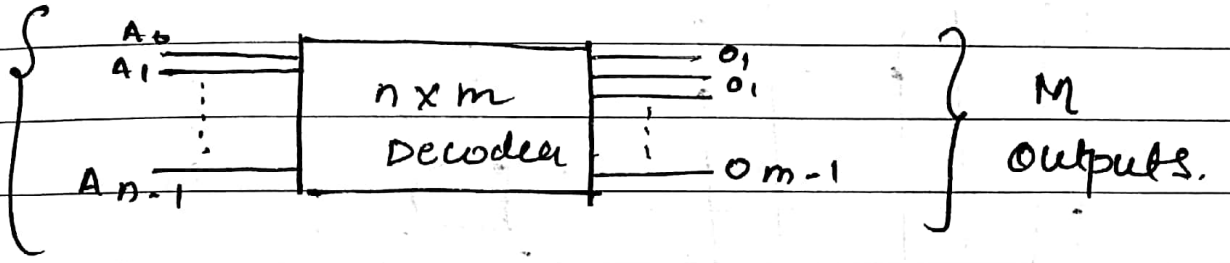
that converts binary information from n i/p lines to a maximum of 2^n unique o/p lines. If the n bit decoded information has unused or don't care combinations, the

decoder o/p will have less than 2^n o/ps.

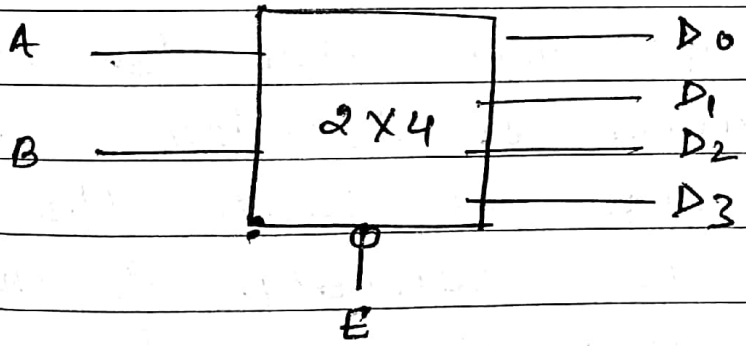
In decoder out of 2^n o/ps, at a time we get one o/p only.

$n \rightarrow$ no. of i/p lines

$m \rightarrow$ no. of o/p lines



2 X 4 Decoder —



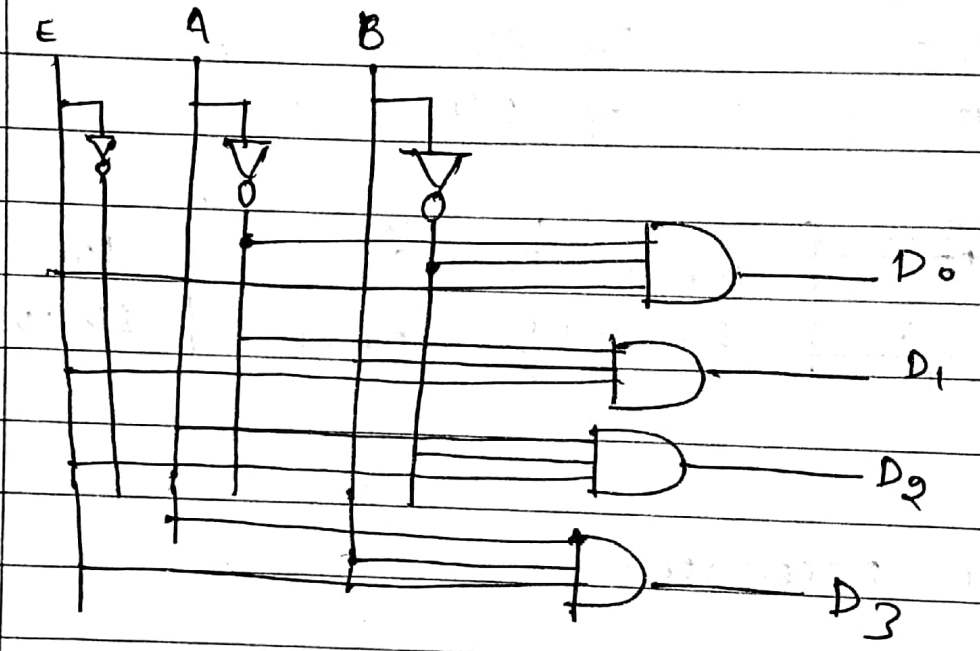
E	A	B	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

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

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

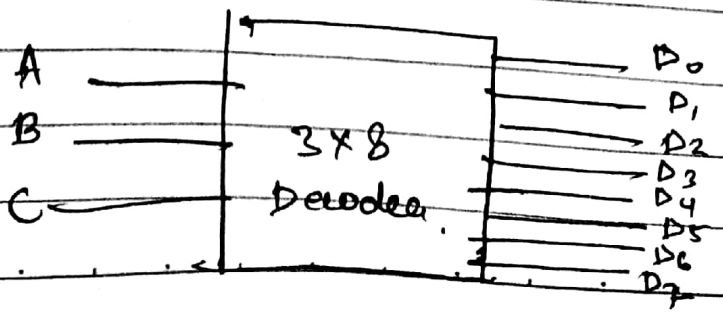
$D_2 = A\,\bar{B}\,\bar{E}$

$D_3 = A\,B\,\bar{E}$



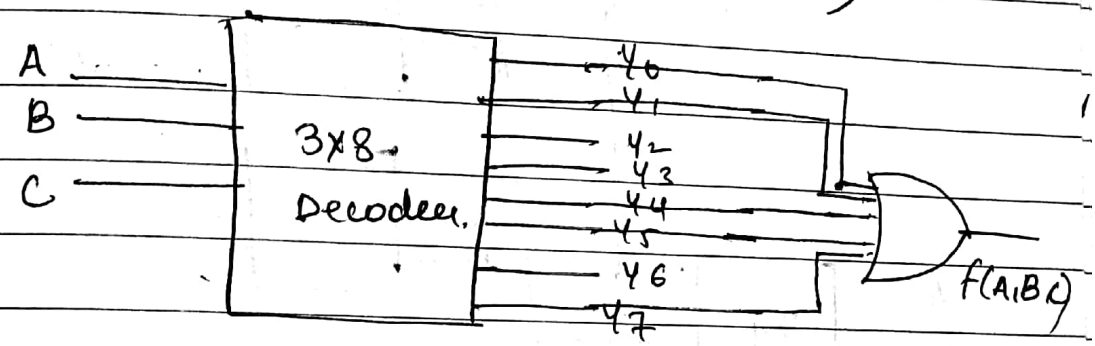
enable i/ps are used to control the operation of decoder.
If enable line is high, the decoder will function normally. If enable line is low, all the o/ps will be forced to the low state regardless of the levels at A inputs.

3×8 Decoder.



Q→ Implement following function using 3X8 decoder.

$f(A,B,C) = \Sigma m(0,1,4,5,7)$



Q→ A combinational ckt defined by $f(w,x,y,z) = xy + z'$ Design the ckt with a decoder and external gates.

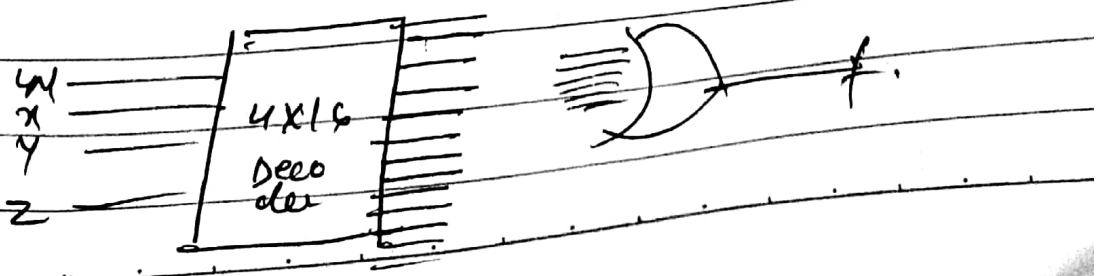
$f(w,x,y,z) = xy + z'$

$= xy(z+z') + z'(x+x')(y+y')$

=

wx \ yz	00	01	11	10
00	I_0	1	3	I_2
01	I_4	5	I_7	I_6
11	I_{12}	15	I_{14}	I_{13}
10	I_8	9	11	I_{10}

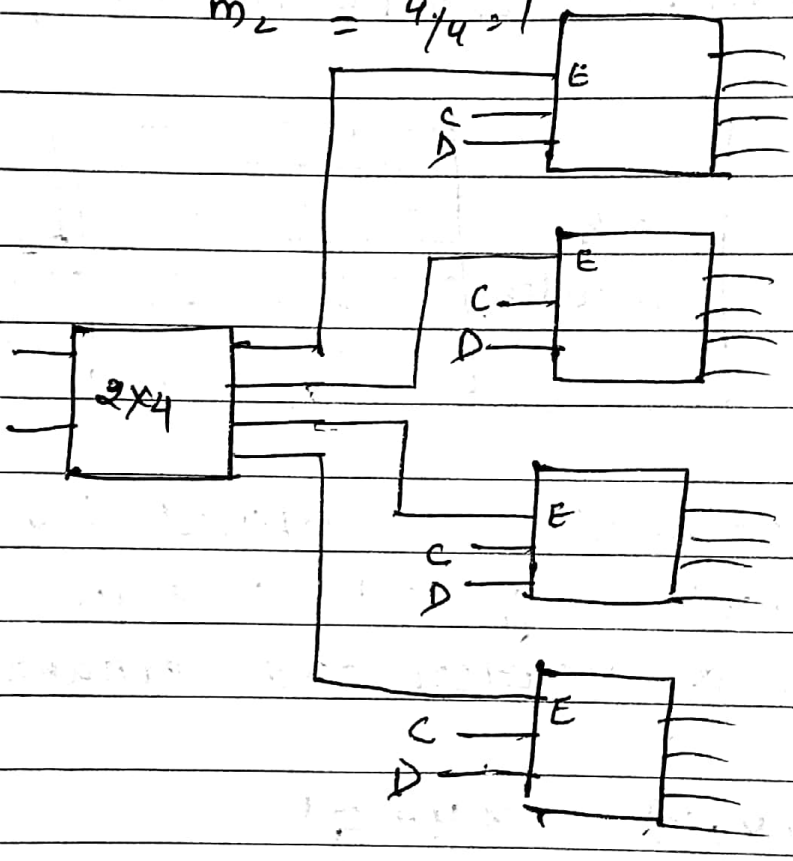
$\Sigma(0,2,4,6,7,8,10,12,14,15)$



4 X 16 using 2 X 4 Decoder.

$m_1 = \frac{16}{4} = 4$

$m_2 = 4/4 = 1$



$2 \times 4 \longrightarrow 5 \times 32$

$\frac{32}{4} = 8$

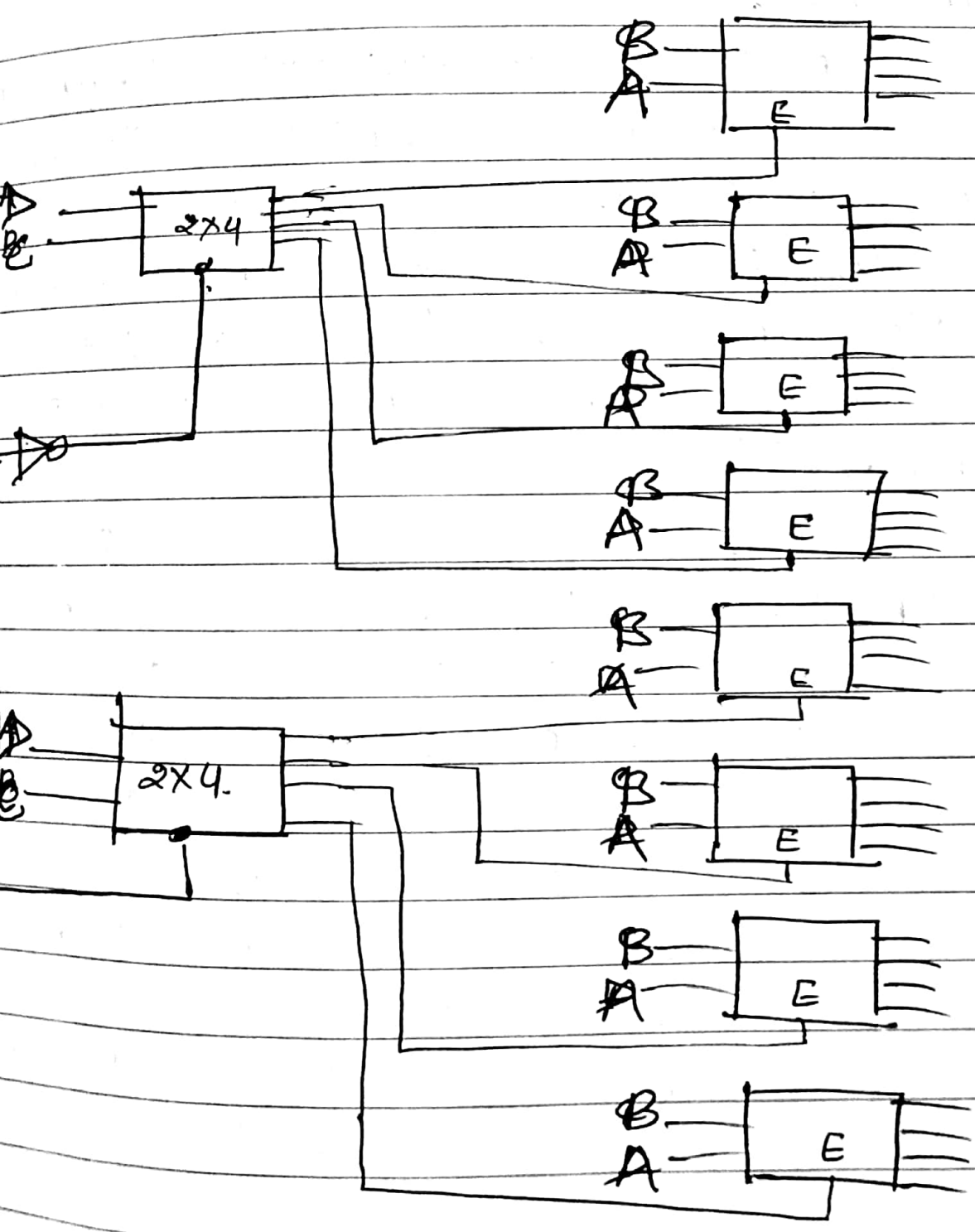
$8/4 = 2$

10

6X64 $\xrightarrow{16+4+1}$ 2X

6X64 $\xrightarrow{8+1}$ 8X

8X256 $\xrightarrow{16+1}$ 4X



Implement half adder using 2×4 decoder.

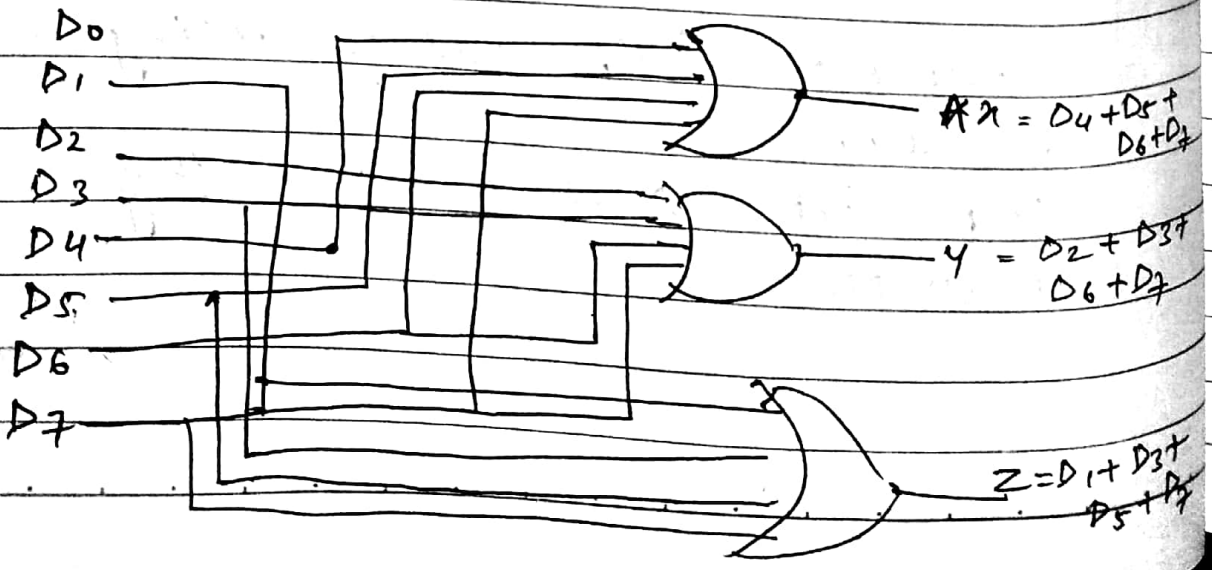
Implement full adder using 3×8

Encoders –

An encoder is a digital function that produces a reverse operation from that of a decoder.

An encoder has 2^n i/p lines and n o/p lines. The o/p lines generate the binary code for the 2^n i/p variables.

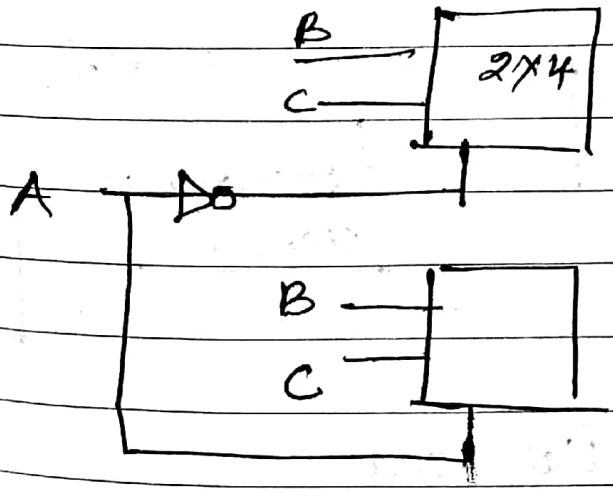
i/p								o/p		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1



Camlin | Page
Date / /

priority Encoder —
An encoder is unable to provide an appropriate o/p code if 2 or more i/p lines are activated simultaneously, thus priority encoders can resolve the problem of simultaneous inputs. When the encoder is enabled and 2 or more i/p signals are activated simultaneously, it ignores the low priority inputs and encodes the highest priority i/p.

3×8 ⟶ 2×4



4×16 ⟶ 3×8