Eskala Ranga Vara (1

UNIT-III

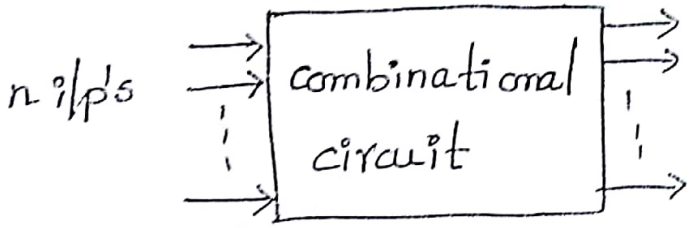
Analysis and synthesis of Combinational circuits

consists of interconnection of logic gates

Combinational circuits:-

→ When logic gates are connected together to produce a specified o/p for certain specified combinations of input variables, with no storage involved, the resulting circuit is called 'combinational circuit'.

⇒ The outputs at any time are determined from the present combination of inputs.



Block diagram of combinational ckt.

→ The 'n' i/p binary variables come from an external source, the 'm' output variables go to an external destination.

→ Each i/p and o/p variable exists physically as a binary signal that represents logic 1 and logic 0.

→ In many applications, the source and destination are storage registers. If the registers are included with the combinational gates, then the total circuit must be considered as a sequential circuit.

→ For n i/p's variables, there are 2^n possible binary input combinations. For each possible input combinations, there is one possible value for each o/p variable.

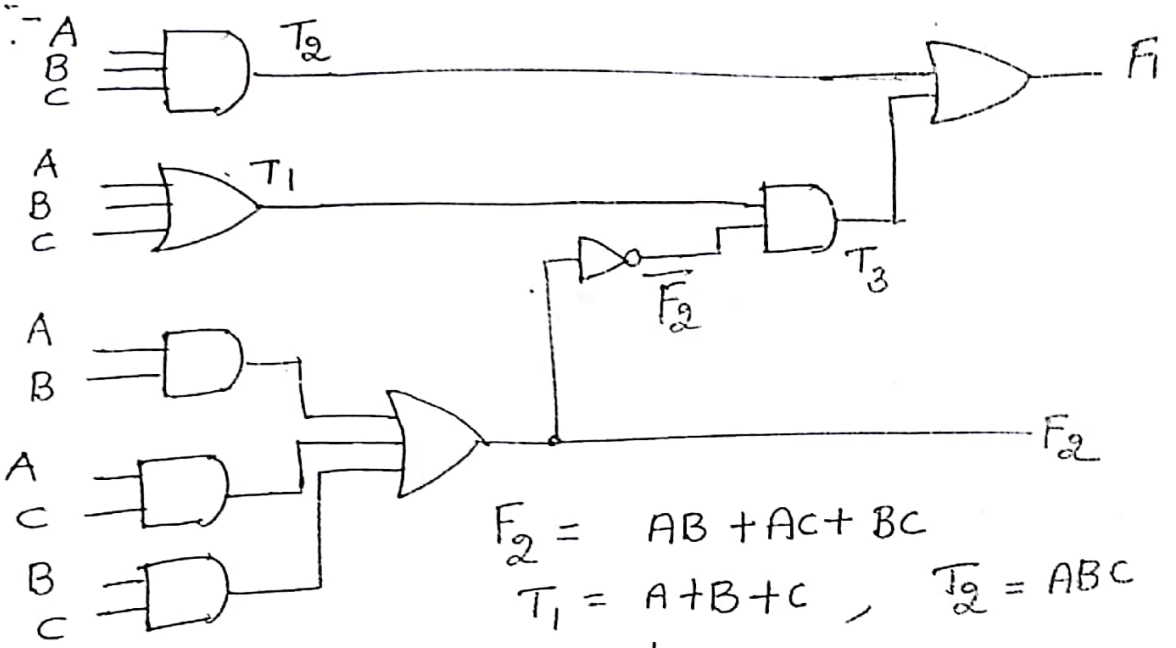
Analysis Procedure:- → The analysis of a combinational circuit can be specified with a truth table that lists the o/p values for each combinations of i/p variables.

circuit requires that we determine the function that the circuit implements.

Steps:-

1. First make sure that the given circuit is combinational circuit and not the sequential circuit. The combinational circuit has logic gates with no feedback path (or) memory elements.
2. Label all gate o/p's that are a function of i/p variables with arbitrary symbols, and determine the boolean functions for each gate o/p.

Eg:-



$F_2 = AB + AC + BC$

$T_1 = A + B + C$, $T_2 = ABC$

$T_3 = F_2' \cdot T_1$ ⇒ $F_1 = T_3 + T_2$

$F_1 = T_3 + T_2 = \overline{F_2} T_1 + T_2 = (AB + AC + BC)' (A + B + C) + ABC$

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

$= 0 + (\overline{A}\,\overline{C} + \overline{A}\,\overline{B} + \overline{B}\,\overline{C})(\overline{B} + \overline{C})(A + B + C) + ABC$

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

$= 0 + \overline{A}B\overline{C} + \overline{A}\,\overline{B}C + ABC + A\overline{B}\,\overline{C}$

→ Now let us construct a truth table for F_1.

A	B	C	F_2	F_2'	T_1	T_2	T_3	F_1
0	0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	1	0	1	1	0	1

Design procedure :-

→ The design of combinational circuits starts from the outline of the problem statement and ends in a logic circuit diagram.

Steps :-

1). The problem defination
2). The determination of number of available input variables and required output variables.
3). Assigning letter symbols to input and output variables.
4) The derivation of truth table indicating the relationships between input and output variables.
5). obtain simplified boolean expression for each output.
6). obtain the logic diagram.

Eg :- code conversion :-

1) BCD to Ex-cess 3 code convertor :-

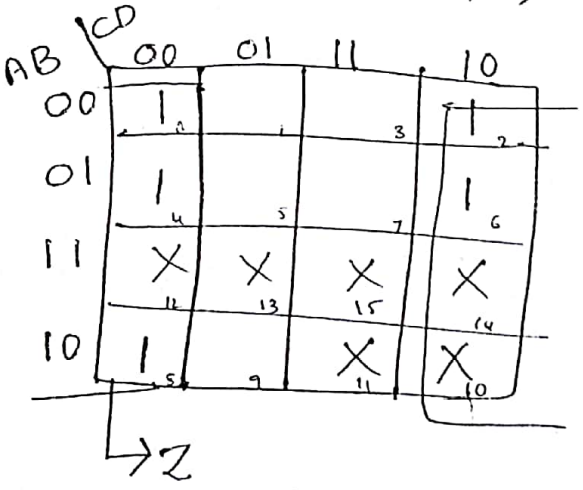
2) i/p Var = 4, o/p Var = 4.

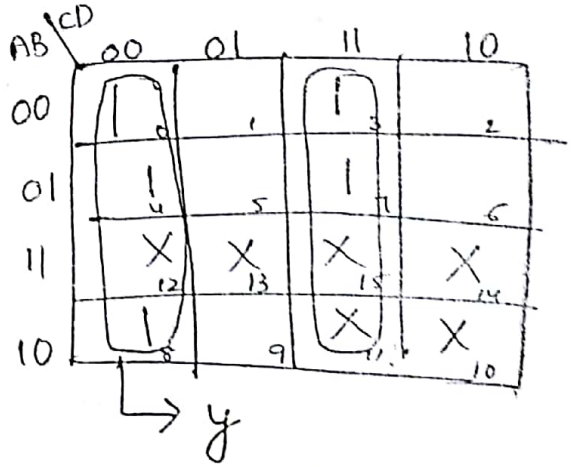
3) A, B, C, D w, x, y, z

4) TT

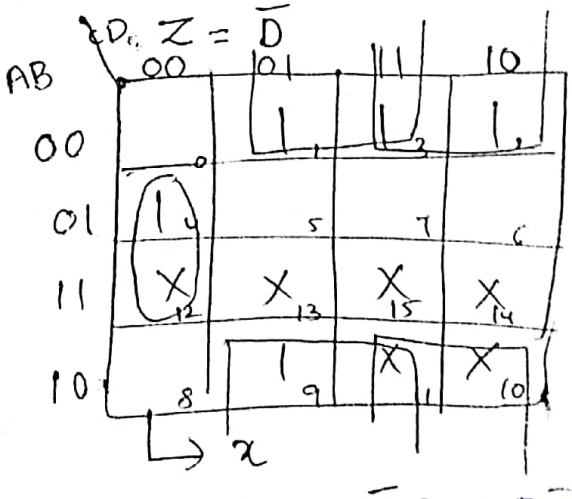
I/P BCD				O/P Excess-3 code			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

BCD, 'X' terms; 10, 11, 12, 13, 14, 15.

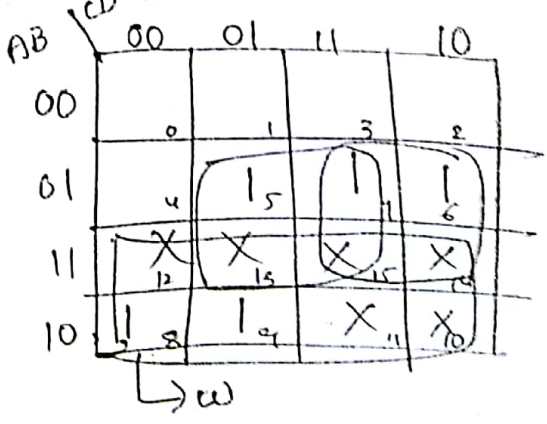




$y = \bar{C}\bar{D} + CD$



$x = \bar{B}D + \bar{B}C + B\bar{C}\bar{D}$



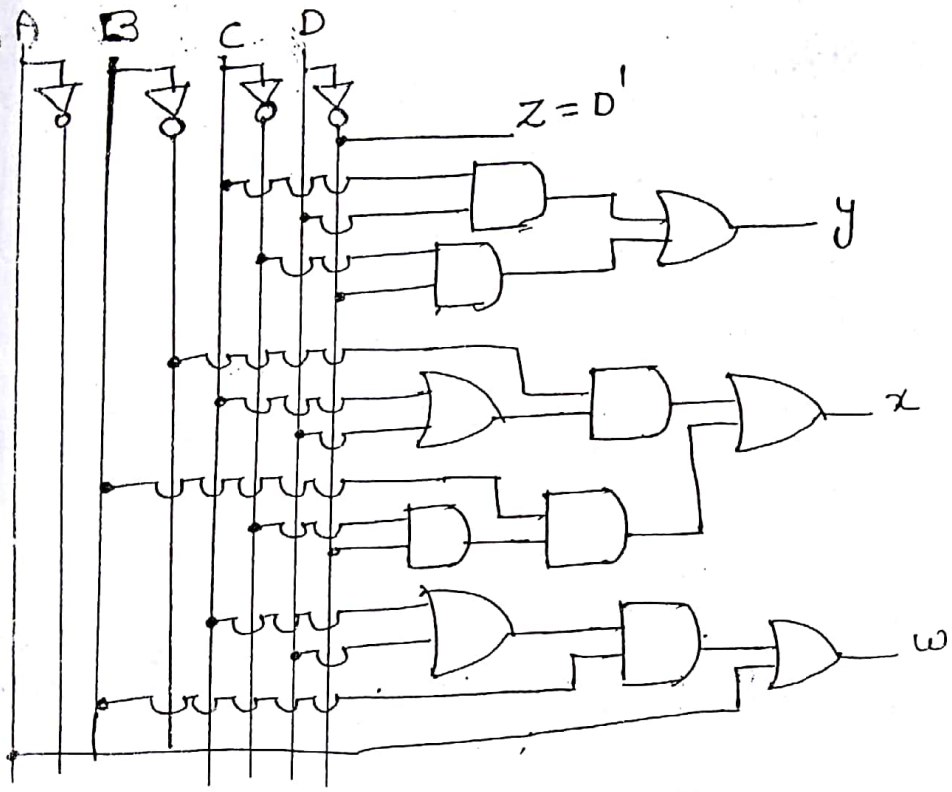
$w = A + BD + BC$

$z = D'$, $y = CD + \bar{C}\bar{D} = CD + \overline{(C+D)}$

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

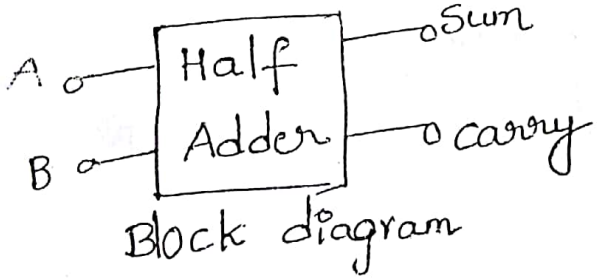
$w = A + B(C+D)$.

Logic diagram:-



Adders:-

1) Half adder:- → The logic circuit which performs addition operation of two digits is called HA.

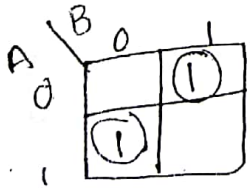


Block diagram

Truth Table

A	B	Sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

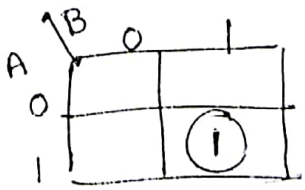
k-map



Sum

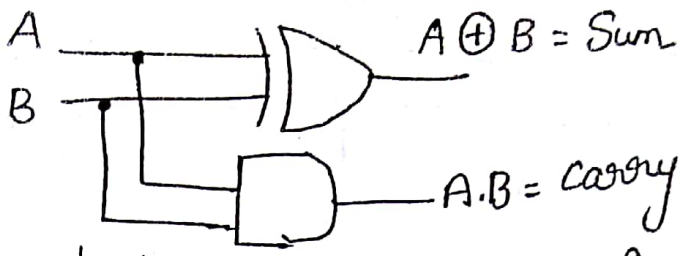
$Sum = A\bar{B} + \bar{A}B$

$= A \oplus B$



$Carry = AB$

Logic Ciraut :-

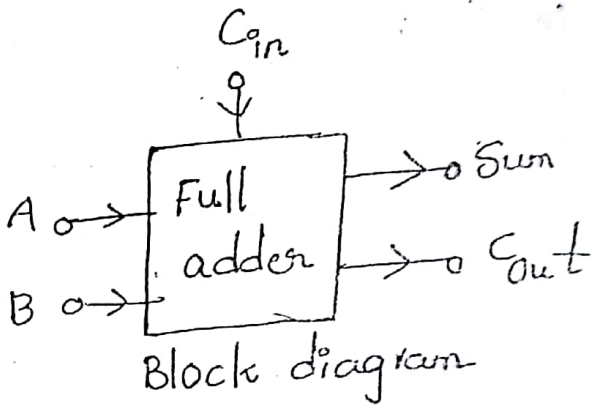


2) Full adder :-

→ The logic ciraut which is used to perform the addition operation with 3 digits / bits is called F.A

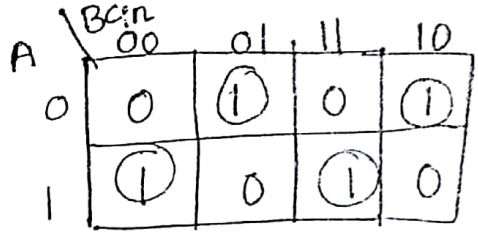
TT :-

A	B	C_{in}	Sum	C_{out}
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



Block diagram

K-map :- 3 variables = $2^3 = 8$ cells

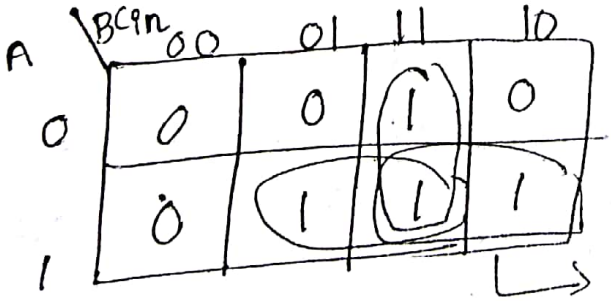


$$Sum = \bar{A}\,\bar{B}\,C_{in} + \bar{A}\,B\,\bar{C_{in}} + A\,\bar{B}\,\bar{C_{in}} + ABC_{in}$$

$$= \bar{A}\left[\bar{B}C_{in} + B\,\bar{C_{in}}\right] + A\left[\bar{B}\bar{C_{in}} + BC_{in}\right]$$

$$= \bar{A}\left[B \oplus C_{in}\right] + A\left[\overline{B \oplus C_{in}}\right] \qquad \left[\because B \oplus C_{in} = \overline{B \oplus C_{in}}\right]$$

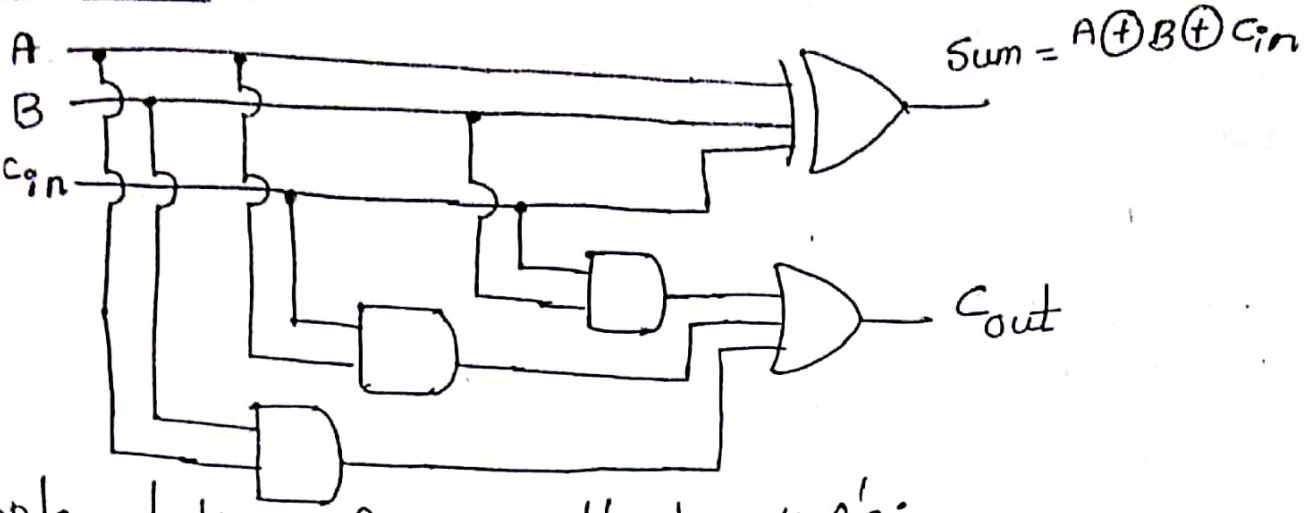
$$Sum = A \oplus B \oplus C_{in}$$



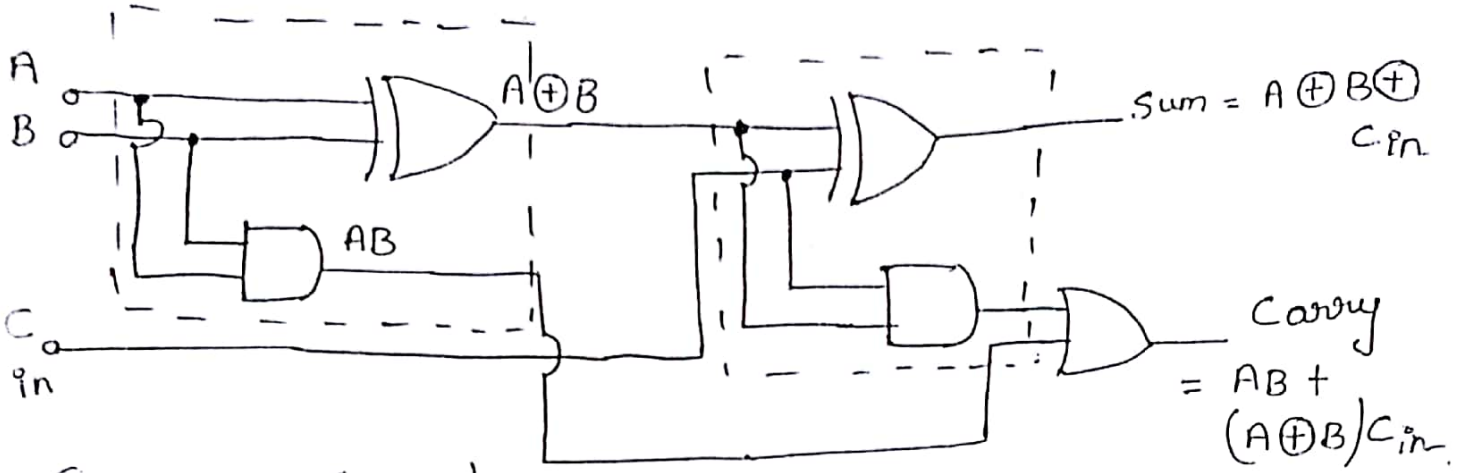
$$C_{out} = AC_{in} + AB + BC_{in}$$

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

Logic Circuit :-



Implementation of FA with two H.A's :-



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

$= AB + (\bar{A}B + A\bar{B})C_{in} = AB + \bar{A}BC_{in} + A\bar{B}C_{in}$

$= AB(1 + C_{in}) + \bar{A}BC_{in} + A\bar{B}C_{in} = AB + A\bar{B}C_{in} + \bar{A}BC_{in} + A\bar{B}C_{in}$

$= BC_{in}(A + \bar{A}) + AB + A\bar{B}C_{in}$

$= BC_{in}(1) + AB(1 + C_{in}) + A\bar{B}C_{in} = BC_{in} + AB + ABC_{in} + A\bar{B}C_{in}$

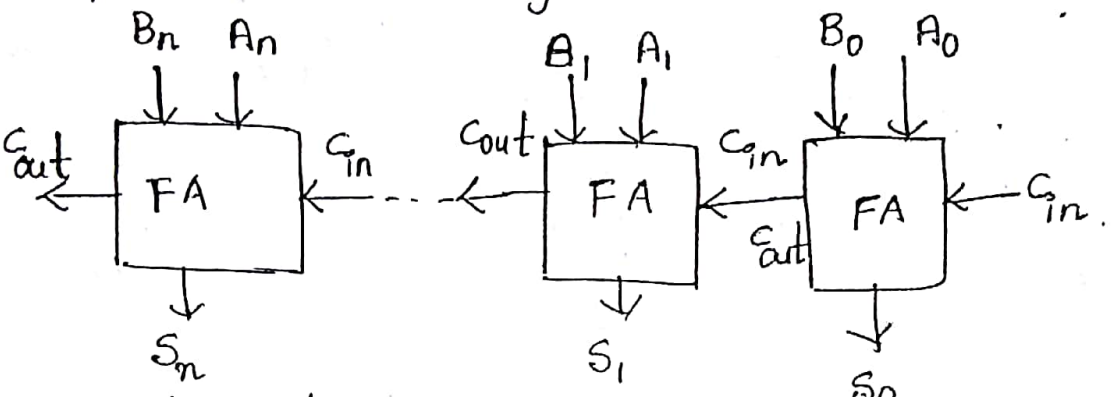
$= BC_{in} + AB + AC_{in}(1) = BC_{in} + AB + AC_{in}$

→ Binary Adder / Parallel Adder :- / Ripple A binary adder is a circuit that produces the arithmetic sum of two binary numbers with more than one bit.

→ In order to add binary numbers additional full-adders must be used. A n-bit parallel adder can be constructed using no. of F.A ckts connec in parallel.

→ The block diagram of n-bit parallel adder using no.

of full-adder circuits connected in cascade, ie, the o/p carry o/p of each adder is connected to the carry input of the next higher-order adder.



4 bit p/l adder
i 74LS83
74LS283

F.A's are in cascade i/p are p/l.

Block dgm of n-bit parallel adder :-

Serial Adder	Parallel Adder
1. SA uses shift registers	1. P.A uses reg with p/l Load capacity
2. S.A requires only one F.A ckt	2. The no. of FA ckts in the Parallel adder = no. of bits in the binary no's.
3. SA is a Sequential ckt	3. Excluding reg, P.A is a purely Combinational ckt.
4. Time required for addition depends on no. of bits	4. Doesnot depends on bits
5. Slower	5. Faster

Carry Look ahead adder :-

→ For an n-bit p/l (or) ripple type the C_{in} of the first stage is made equal to zero. The C_{out} of 1st stage becomes as C_{in} of 2nd stage.

→ The Sum & Carry o/p of many stages cannot be produce until the i/p carry occurs, this leads to a time delay in addition process. The delay is known as carry propogation delay.

Eg :- 5 - 0101
3 - 0011
1000 = 8.

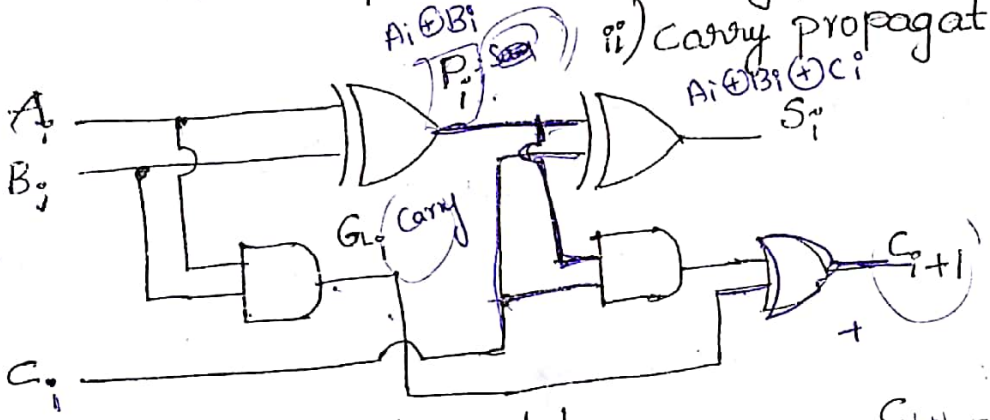
→ The carry in the MSB bit depends upon the previous carry. The adder will not produce correct result until LSB carry has generated to the intermediat F.A. This represents time delay (or) propogation delay produced in each F.A.

→ For eg propogation delay is 30nSec then the total time required for 4-bit adder is 120 nSec.

→ one method of speeding up this process by eliminating inter stage carry delay is called look ahead - carry addition.

→ This method utilizes logic gates to look at the lower order bits of the augend and addend to see if a higher order carry is to be generated.

→ It uses 2 functions : i) carry generate
ii) carry propagate



Full adder ckt.

$P_i = A_i \oplus B_i$

$G_i = A_i B_i$

$S_i = P_i \oplus C_i$

$C_{i+1} = C_i P_i + G_i$

C_{i+1} → C_2

→ G_i → carry generate & it produces carry when both A_i & B_i are 1, regardless of the input carry.

→ P_i → carry propagate because it is term associated with the propagation of the carry from C_i to C_{i+1}.

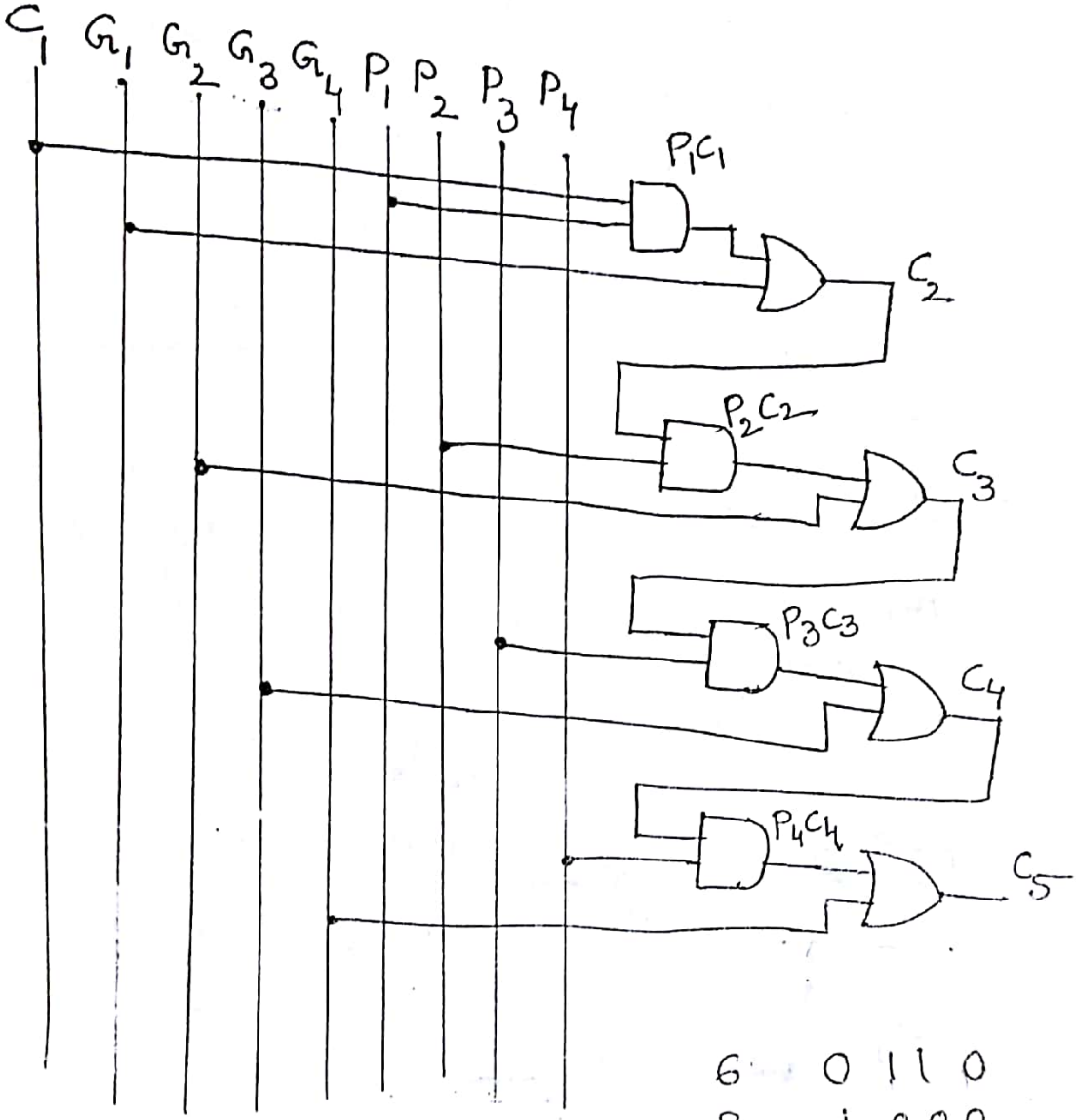
$i = 1,\ C_2 = C_1 P_1 + G_1$

$i = 2,\ C_3 = C_2 P_2 + G_2 = G_2 + P_2 (G_1 + P_1 C_1)$

$i = 3,\ C_4 = G_3 + P_3 C_3 = G_3 + P_3 [G_2 + P_2 (G_1 + P_1 C_1)]$.

$i=4$, $C_5 = G_4 + P_4 C_4 = G_4 + P_4 [G_3 + P_3 (G_2 + P_2 (G_1 + P_1 C_1))]$

Logic circuit :-



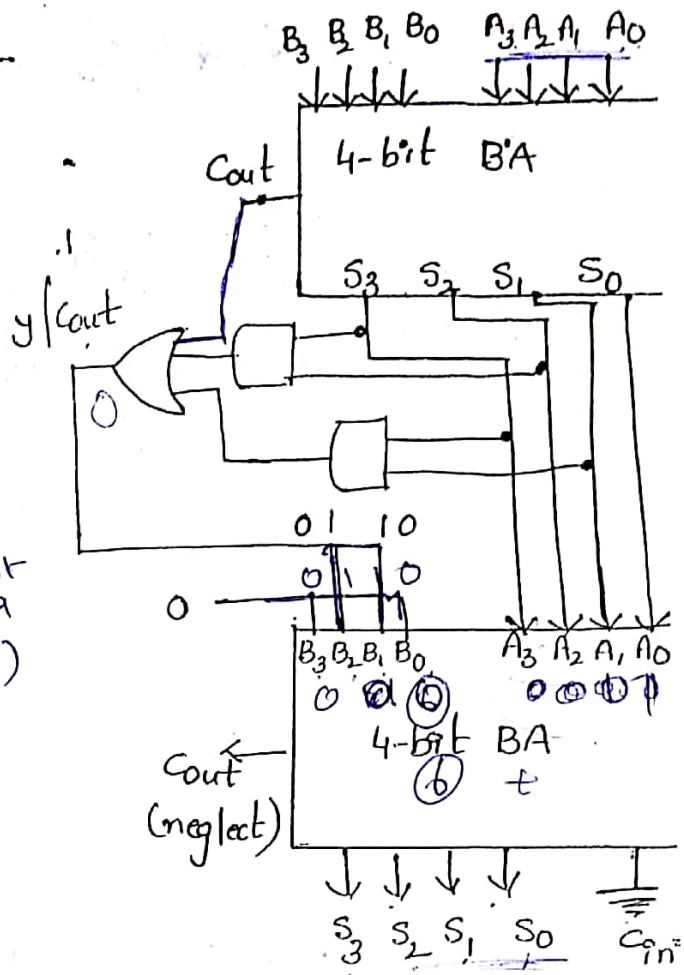
Decimal adder :- Eg :- + 6 0 1 1 0
8 1 0 0 0
1 0 1 1 1 0 Illegal
0 1 1 0
0 1 0 0
0 0 0 1 1 0 1 0 = 14.
1

Design a logic circuit to detect sum is greater than 9.

S_3S_2 \ S_1S_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10			1	1

$Y = S_3 S_2 + S_1 S_3$

Dec	I/P's S_3	S_2	S_1	S_0	o/p y
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0 (ie not adding any 6)
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1



→ If the value is greater than 9 then the o/p (y_{Cout}) is become one (1) and 6 is generated it is added to the result. OW (<9) the result is as it is.

Subtractor :- HS :- HS is a combination circuit that subtract two bits and produces their difference.

TT :-

A	B	Diff	Borr
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

k-map :-

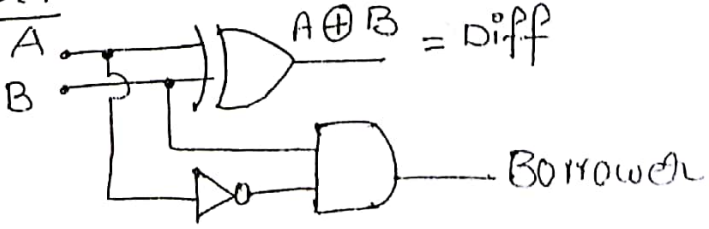
A \ B	0	1
0	0	(1)
1	(1)	0

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

A \ B	0	1
0	0	(1)
1	0	0

$B = \bar{A}B$

Logic ckt :-

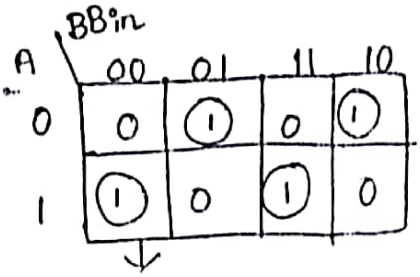


Full Subtractor:- → It is a combinational ckt that Subtract 3 bits & produces their Bout.

TT:-

A	B	B_{in}	Diff	B_{out}
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

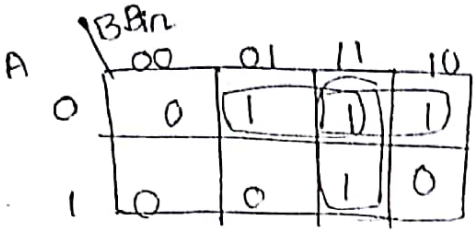
K-map:-



$$Diff = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B_{in}} + A\bar{B}\bar{B_{in}} + ABB_{in}$$

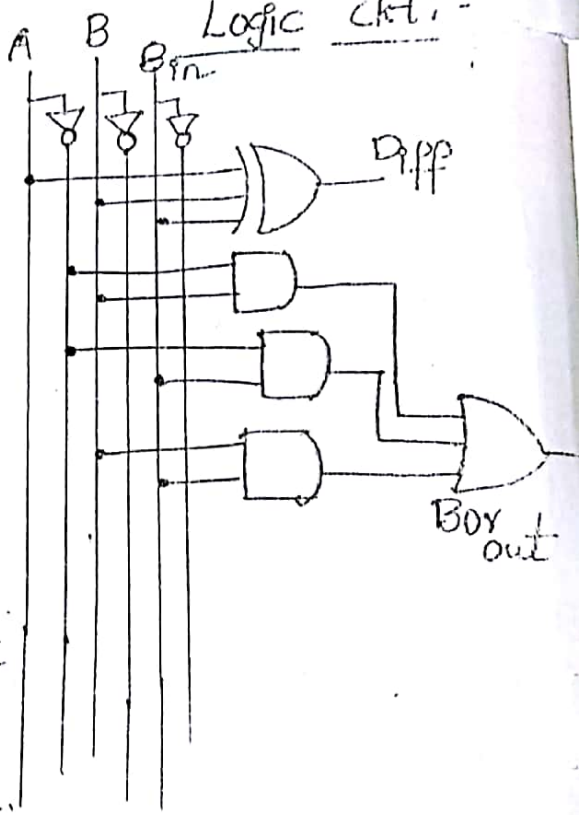
$$= \bar{A}[\bar{B}B_{in} + B\bar{B_{in}}] + A[\bar{B}\bar{B_{in}} + BB_{in}]$$

$$= A \oplus B \oplus B_{in}$$

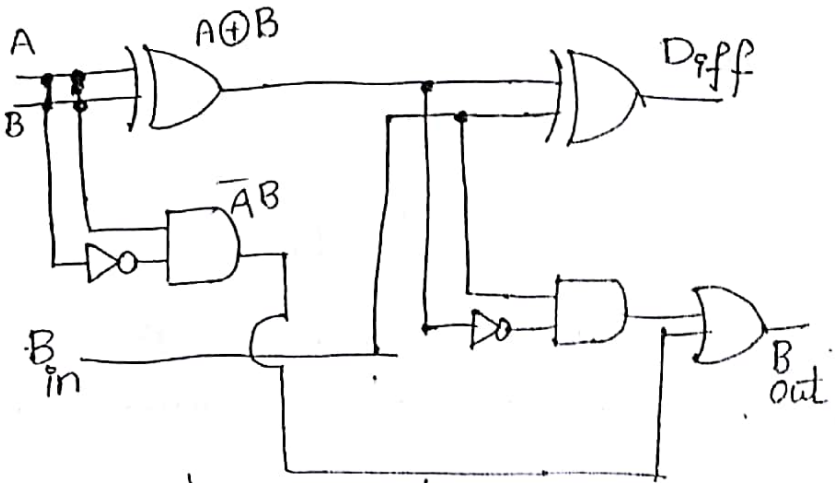


$$B = \bar{A}B_{in} + \bar{A}B + BB_{in}$$

Logic ckt:-



Implementation of FS with two HS:-

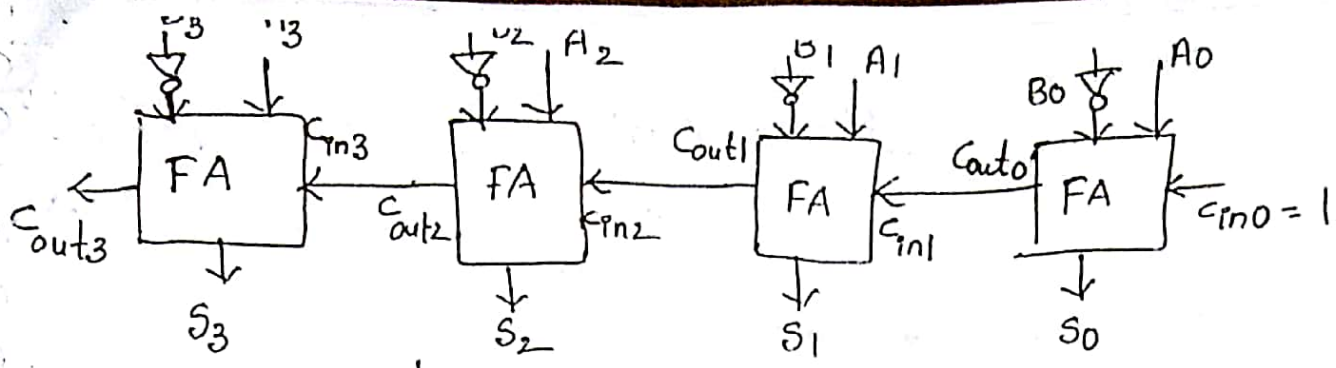


Binary | Parallel | ripple Subtractor:-

A - B :- 2's comple of B + A

1's comple of B + A

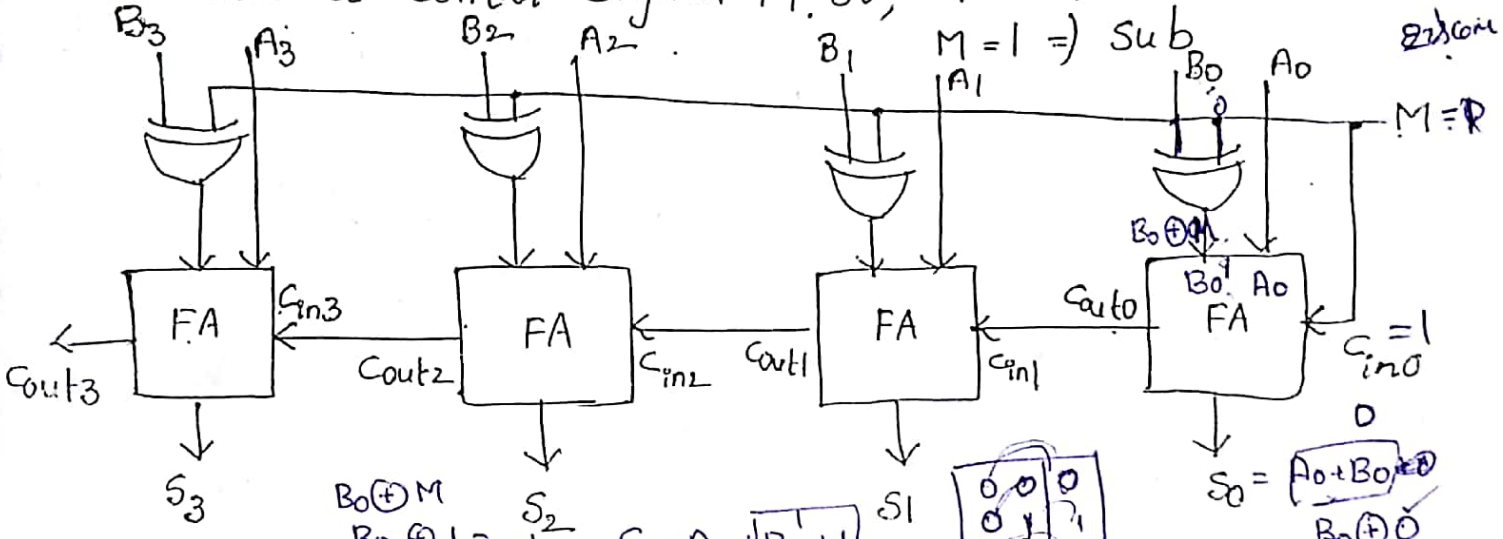
→ +1



Parallel Adder/subtractor :-

→ To make the ckt both as Parallel adder and subtractor we need a control signal M. so, M = 0 ⇒ add

M = 1 ⇒ sub



0	0	0
0	1	1
1	0	1
1	1	0

Binary Multiplier :-

→ The combination circuits implemented to perform multiplication is called combinational multiplier.

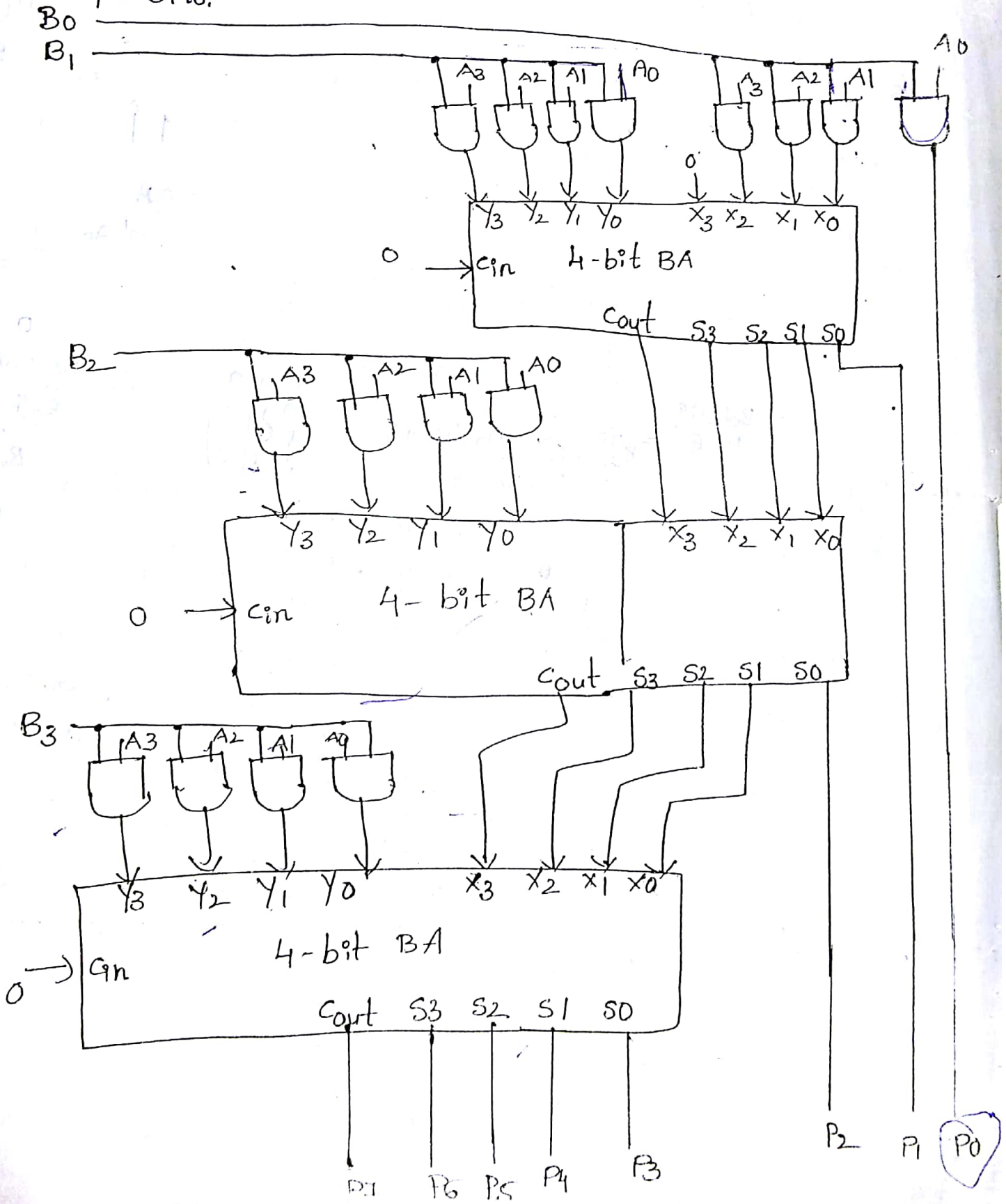
→ The advances in VLSI technology have made it possible to build combinational ckts that perform n×n bit multiplication.

$$\begin{array}{rcccc} & A_3 & A_2 & A_1 & A_0 \\ \times & B_3 & B_2 & B_1 & B_0 \\ \hline & A_3B_0 & A_2B_0 & A_1B_0 & A_0B_0 \\ & A_3B_1 & A_2B_1 & A_1B_1 & A_0B_1 \\ & A_3B_2 & A_2B_2 & A_1B_2 & A_0B_2 \\ + & A_3B_3 & A_2B_3 & A_1B_3 & A_0B_3 \\ \hline P_7 & P_6 & P_5 & P_4 & P_3 \quad P_2 \quad P_1 \quad P_0 \end{array}$$

→ Partial product

→ Let us generalized the multiplication process for a 4x4 multiplier for two unsigned integers:
multiplicand $A = A_3\ A_2\ A_1\ A_0$ and multiplier
$B = B_3\ B_2\ B_1\ B_0$

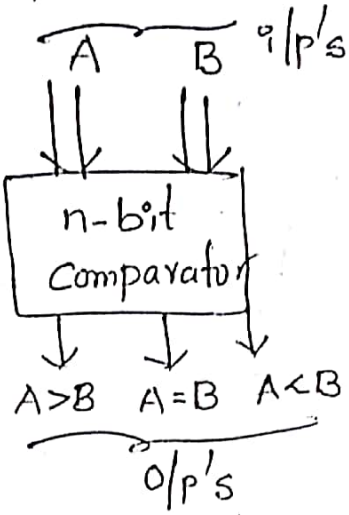
→ 4 = Partial product & each pp consists of 4 product components.



Magnitude comparator:-

→ A comparator is a special combinational circuit designed primarily to compare the relative magnitud of two binary numbers.

→ The below fig represents the block diagram of an n-bit comparator. It receives two n-bit numbers A and B as inputs and outputs are A>B, A=B, A<B

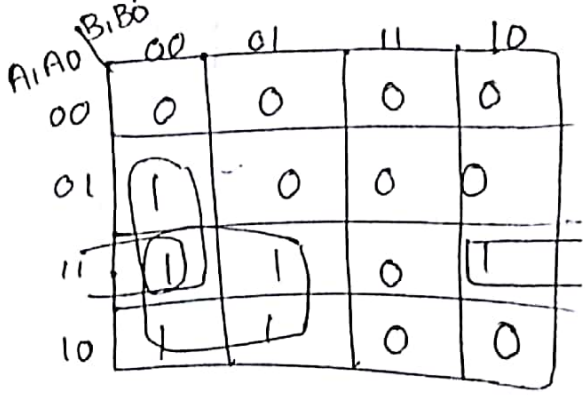


→ Design 2-bit comparator using gates.

Inputs				outputs		
A_1	A_0	B_1	B_0	A>B	A=B	A<B
0	0	0	0	0	1	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	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

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

K-map.



A>B

$A>B = A_1\bar{B_1} + A_0\bar{B_1}\bar{B_0} + A_1A_0\bar{B_0}$

A_1A_0 \ B_1B_0	00	01	11	10
00	(1)	0	0	0
01	0	(1)	0	0
11	0	0	(1)	0
10	0	0	0	(1)

$A = B$

$$A=B = \overline{A_1}\,\overline{A_0}\,\overline{B_1}\,\overline{B_0} + \overline{A_1}A_0\overline{B_1}B_0 + A_1A_0B_1B_0 + A_1\overline{A_0}B_1\overline{B_0}$$

$$= \overline{A_1}\,\overline{B_1}\left[\overline{A_0}\,\overline{B_0} + A_0B_0\right] + A_1B_1\left[A_0B_0 + \overline{A_0}\overline{B_0}\right]$$

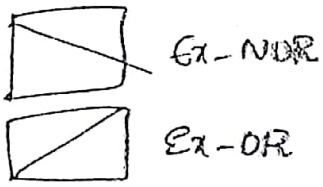
$$= (A_0 \odot B_0)(A_1 \odot B_1)$$

A_1A_0 \ B_1B_0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

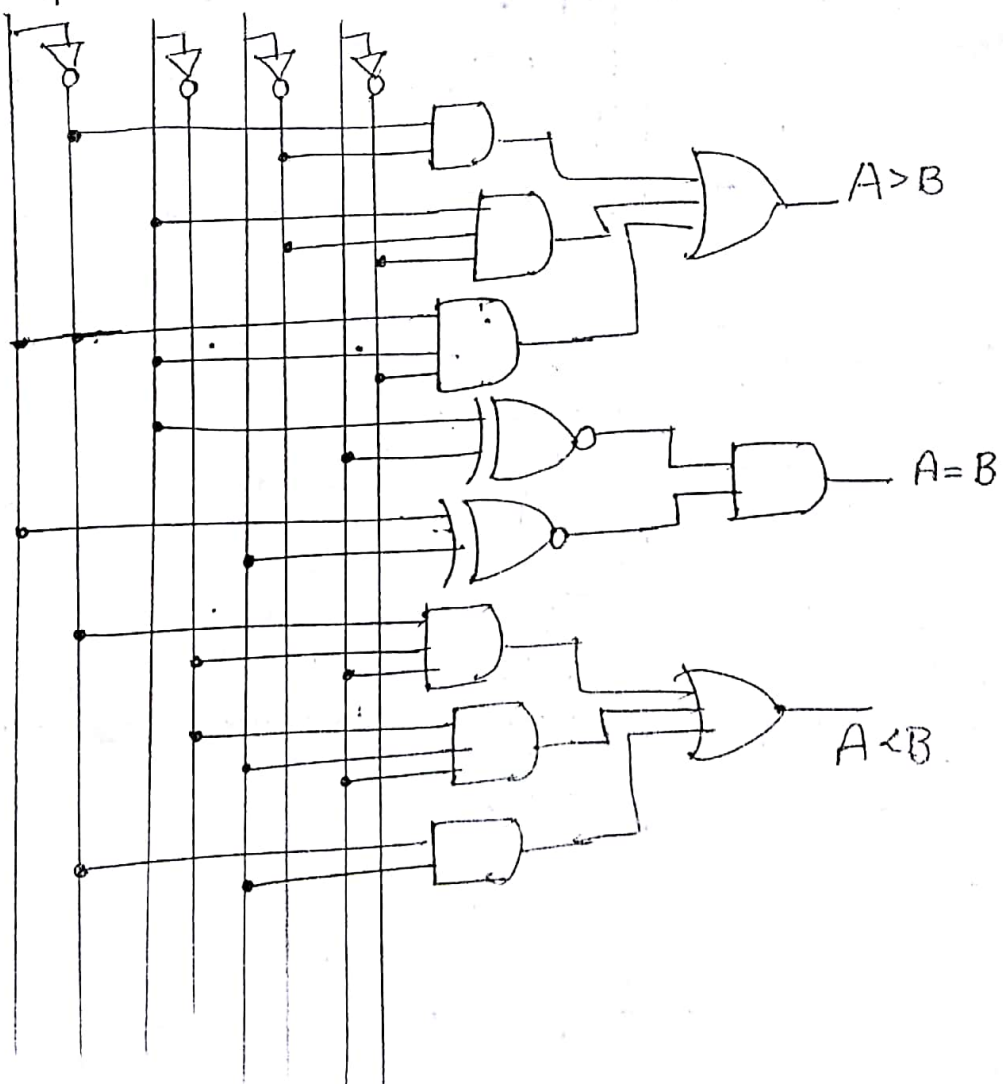
$A < B$

$$A<B = \overline{A_1}\,\overline{A_0}B_0 + \overline{A_0}B_1B_0 + \overline{A_1}B_1$$

Note :- k-map



Logic diagram :-



Now to design a 4 bit comparator we will take the results of 2-bit comparator.

→ Let $A = A_3 A_2 A_1 A_0$; $B = B_3 B_2 B_1 B_0$

a) For $A=B = x_3 x_2 x_1 x_0$ $[x_3 = A_3 = B_3, ----]$

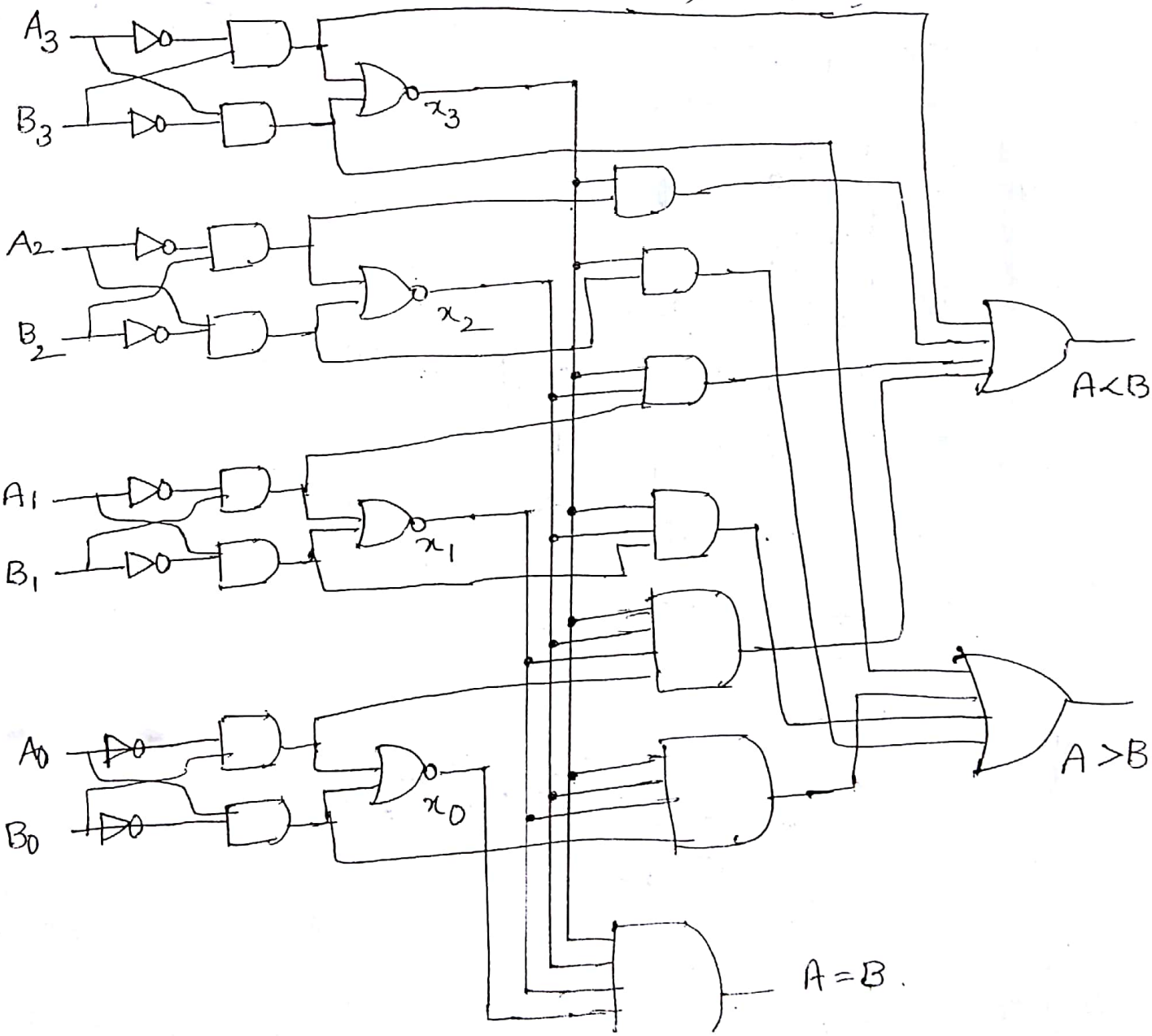
b) For $A>B$, $A<B$, we inspect the relative mag of pairs from MSB to LSB.

$$A>B = \overline{A_3}\,\overline{B_3} + x_3 A_2 \overline{B_2} + x_3 x_2 A_1 \overline{B_1} + x_3 x_2 x_1 A_0 \overline{B_0}$$

$$A<B = \overline{A_3} B_3 + x_3 \overline{A_2} B_2 + x_3 x_2 \overline{A_1} B_1 + x_3 x_2 x_1 \overline{A_0} B_0$$

$$x_3 = A_3 B_3 + \overline{A_3}\,\overline{B_3} = \frac{(A_3 + \overline{B_3})(\overline{A_3} + B_3)}{(A_3 + \overline{B_3})(\overline{A_3} + B_3)} = \overline{(A_3 + \overline{B_3})} + \overline{(\overline{A_3} + B_3)} = \overline{A_3} \cdot B_3 + A_3 \overline{B_3}$$

= Ex NOR.



Decoders:- UNIT-5

State Reduction and assignment:-

adv:- Less Logic gates, Less no. of FF, H/w ↓, delay ↓, speed ↑.

State Assignment:-

i) states which have same next state for given input should be given adjacent assignment

ii) states which have next state of the same state should be given adjacent assignment.

iii) [simplification of o/p fn] state which have the same o/p for the given i/p should be given adj assig-

Eg:-

PS	Next state		o/p	
	x=0	x=1	x=0	x=1
S_0	S_1	S_2	0	0
S_1	S_3	S_2	0	0
S_2	S_1	S_4	0	0
S_3	S_5	S_2	0	0
S_4	S_1	S_6	0	0
S_5	S_5	S_2	1	0
S_6	S_1	S_6	0	1

State Table

i) a) x = 0

PS	S_0	S_2	S_4	S_6
NS	S_1	S_1	S_1	S_1

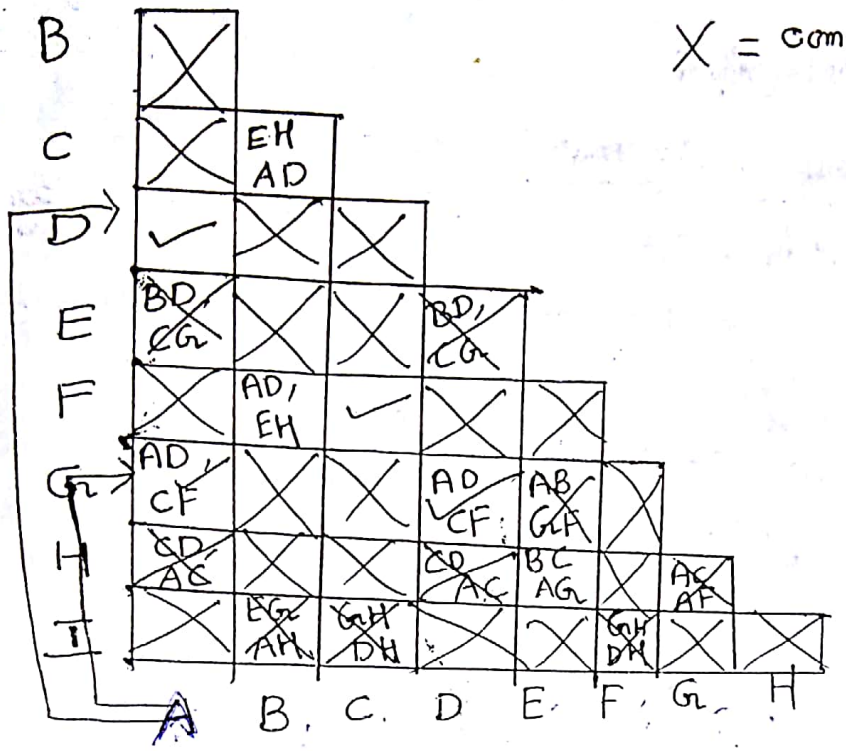
PS	S_3	S_5
NS	S_5	S_5

b) x = 1

PS	S_0	S_1	S_3	S_5
NS	S_2	S_2	S_2	S_2

PS	S_4	S_6
NS	S_6	S_6

X = comp at 2nd level



$\overline{AD}$, $\overline{AG}$, $\overline{BC}^{\alpha}$, $\overline{BF}^{\alpha}$, $\overline{CF}^{\alpha}$, $\overline{DG}$, EH, $\underline{I}$

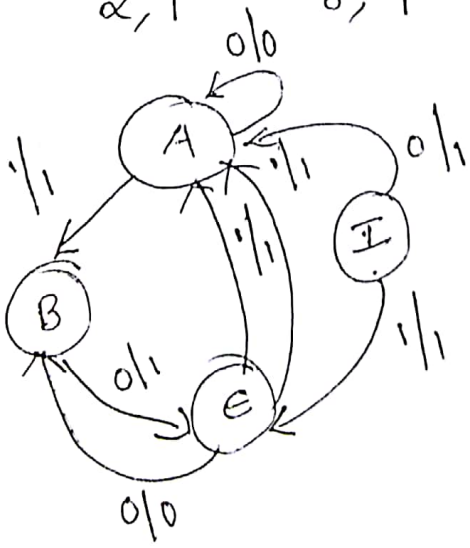
ADG, BCF, EH, $\underline{I}$

α β δ η.

9 states reduced to 4 states

PS	NS, Z	
	I_1	I_2
α	α, 0	β, 1
β	δ, 1	α, 1
δ	β, 0	α, 1
η	α, 1	δ, 1

PS	NS, Z	
	$I_1=0$	$I_2=1$
A	A, 0	B, 1
B	E, 1	A, 1
E	B, 0	A, 1
I	A, 1	E, 1



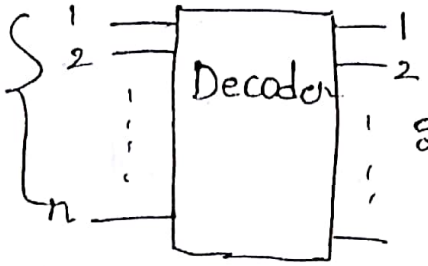
AD, AG, BC, BF, BI, (F, CI, DG, EH, FI

Decoders:- → A decoder is a combinational circuit that converts binary information from n i/p lines to a max of 2^n unique o/p lines.

→ The decoders presented here are called n-to-m line decoders, where $m \leq 2^n$.

→ It converts coded inputs to coded o/p's where the input and output codes are different.

i/p's



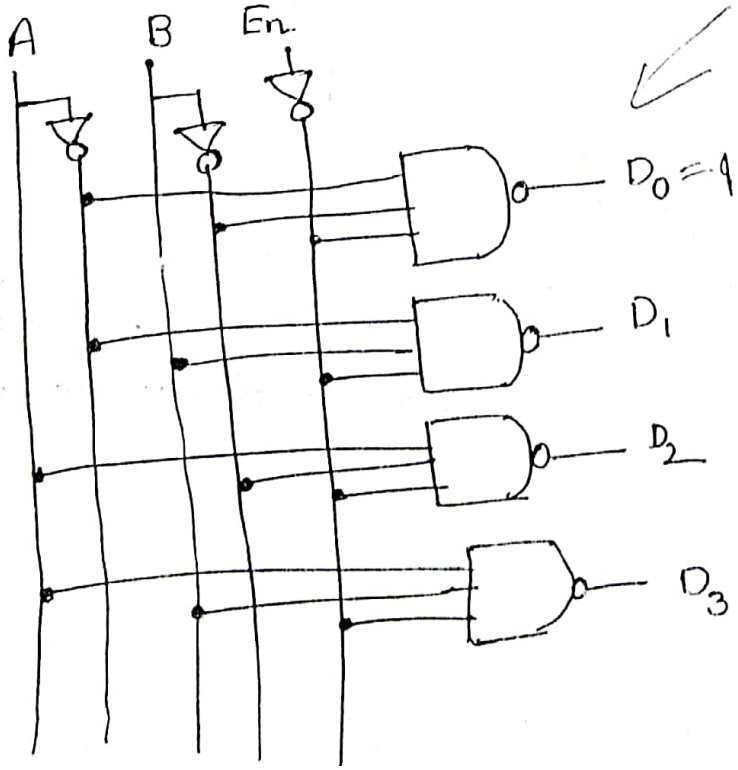
i/p to o/p relation = $n : 2^n$

1) 2:4 Decoder (or) Binary decoder:-

→ Here enable input constructed with NAND gates (So, En = 0, o/p will be produced).

TT :-

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



TT by En = 0

En	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

2) 3:8 Decoder :-

TT:-

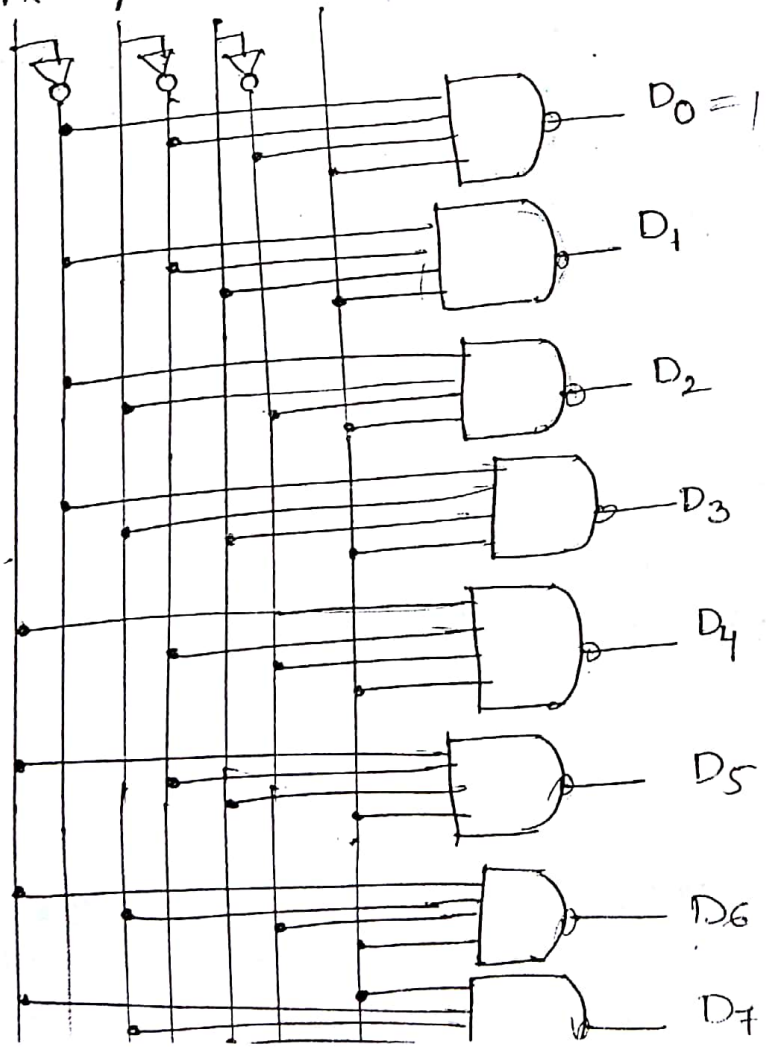
Inputs			EN	outputs							
x	y	z		D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
X	X	X	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0	0
1	0	1	1	0	0	0	0	0	1	0	0
1	1	0	1	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

Logical circuit:-

A=1, B=1, EN=1

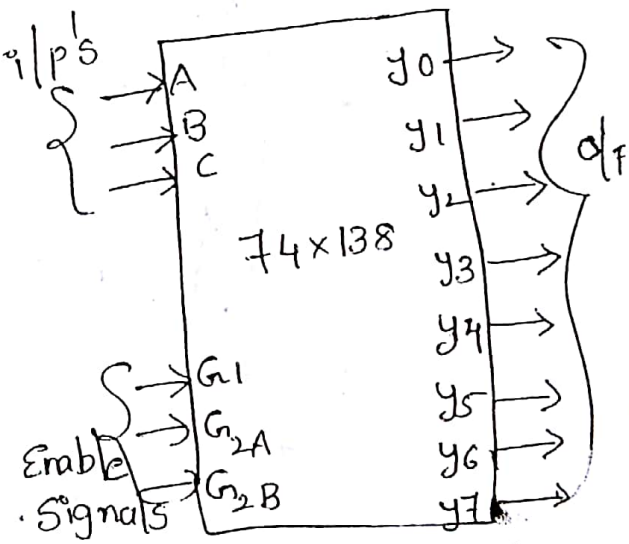
A=1 B=

x y z En



Pin diagram of 3:8 Decoder

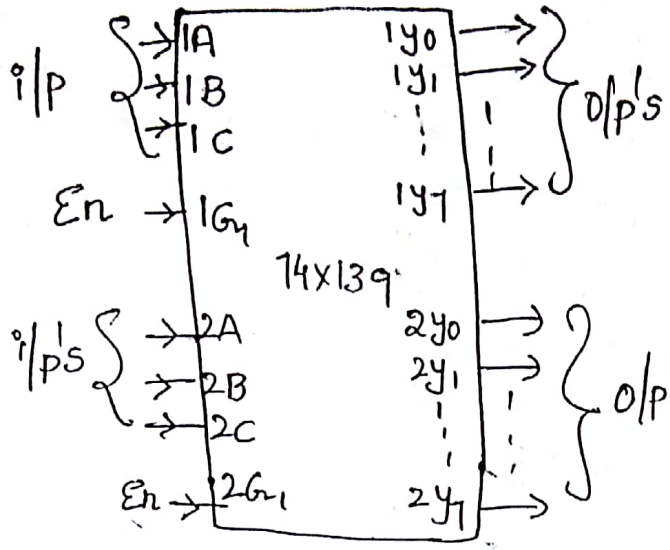
74 series ICs



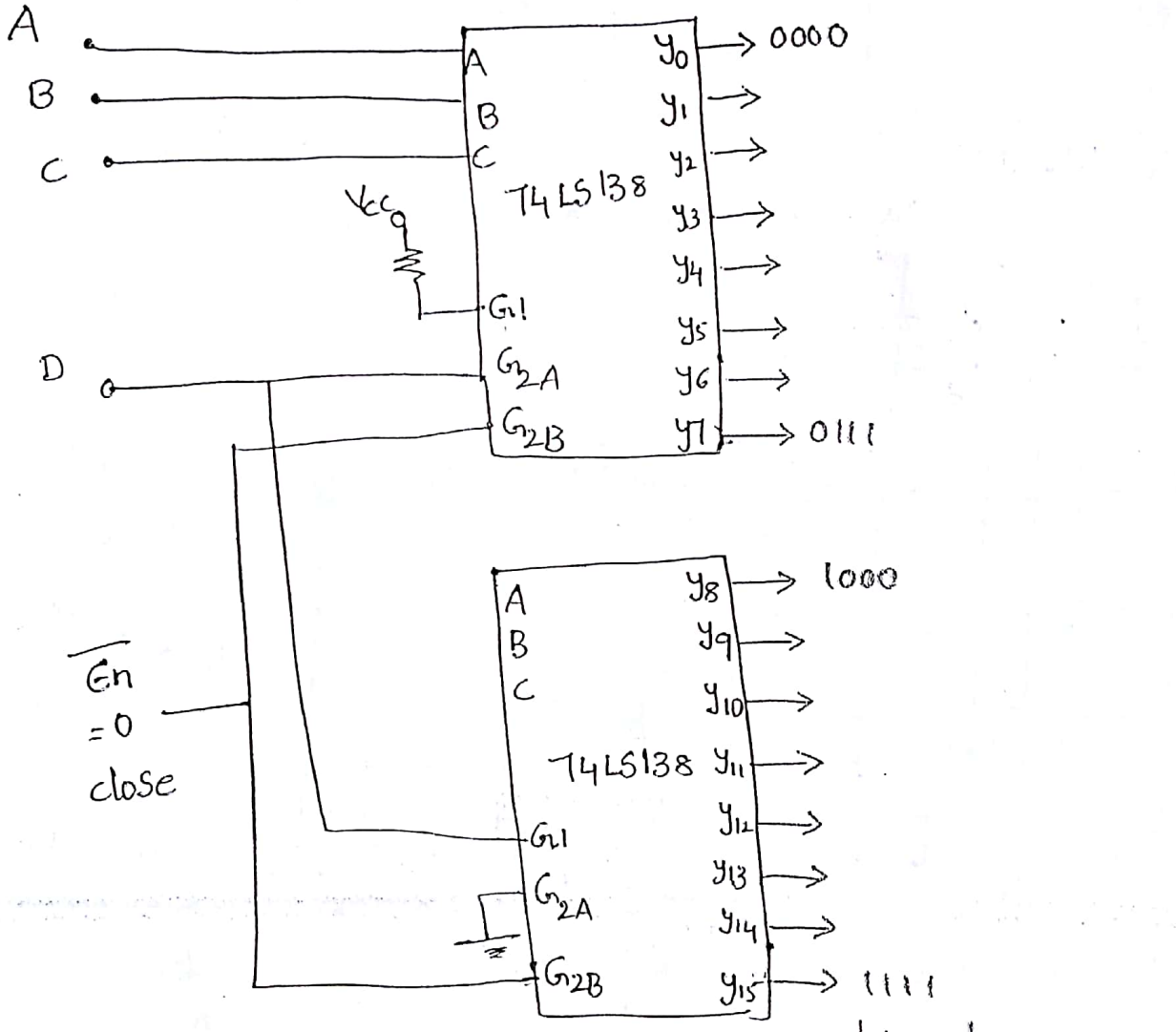
G_1 = active high Enable signal

G_{2A}, G_{2B} = active Low Enable signals

74x139 :- 3:8 dual decoder



→ Design a 4:16 using 2 3:8 decoders.



1st Decoder 2nd decoder

D = 0, active low y_0 ---- y_7 y_8 ---- y_{15}

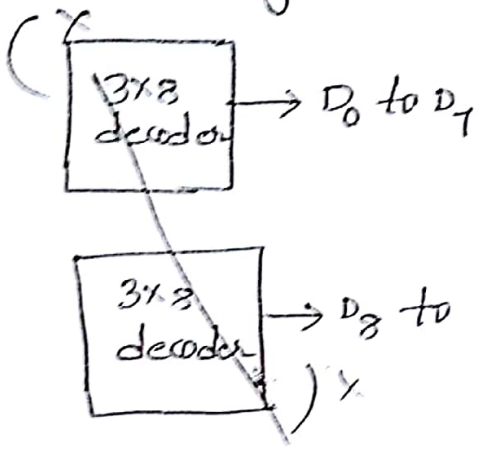
0000 0111 0000 0000

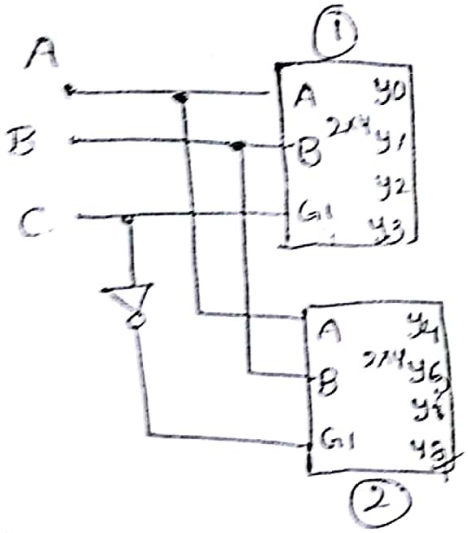
D = 1, active high 0000 ---- 0000 1000 ---- 1111

→ A decoder with enable input can function as a demultiplexer – a circuit that receives information from a single line and directs it to one of 2^n possible output lines.

→ The selection of a specific output is controlled by the bit combination of 'n' selection lines.

→ Design 3:8 decoder using two 2:4 decoders.





when C = 1, ① st dec = ON
C = 0, 2nd dec = ON

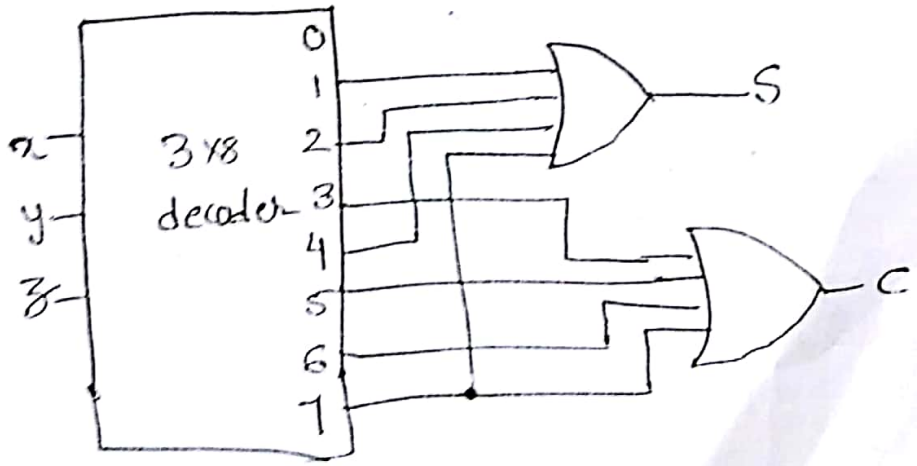
Combinational Logic implementation

From TT of FA we obtain

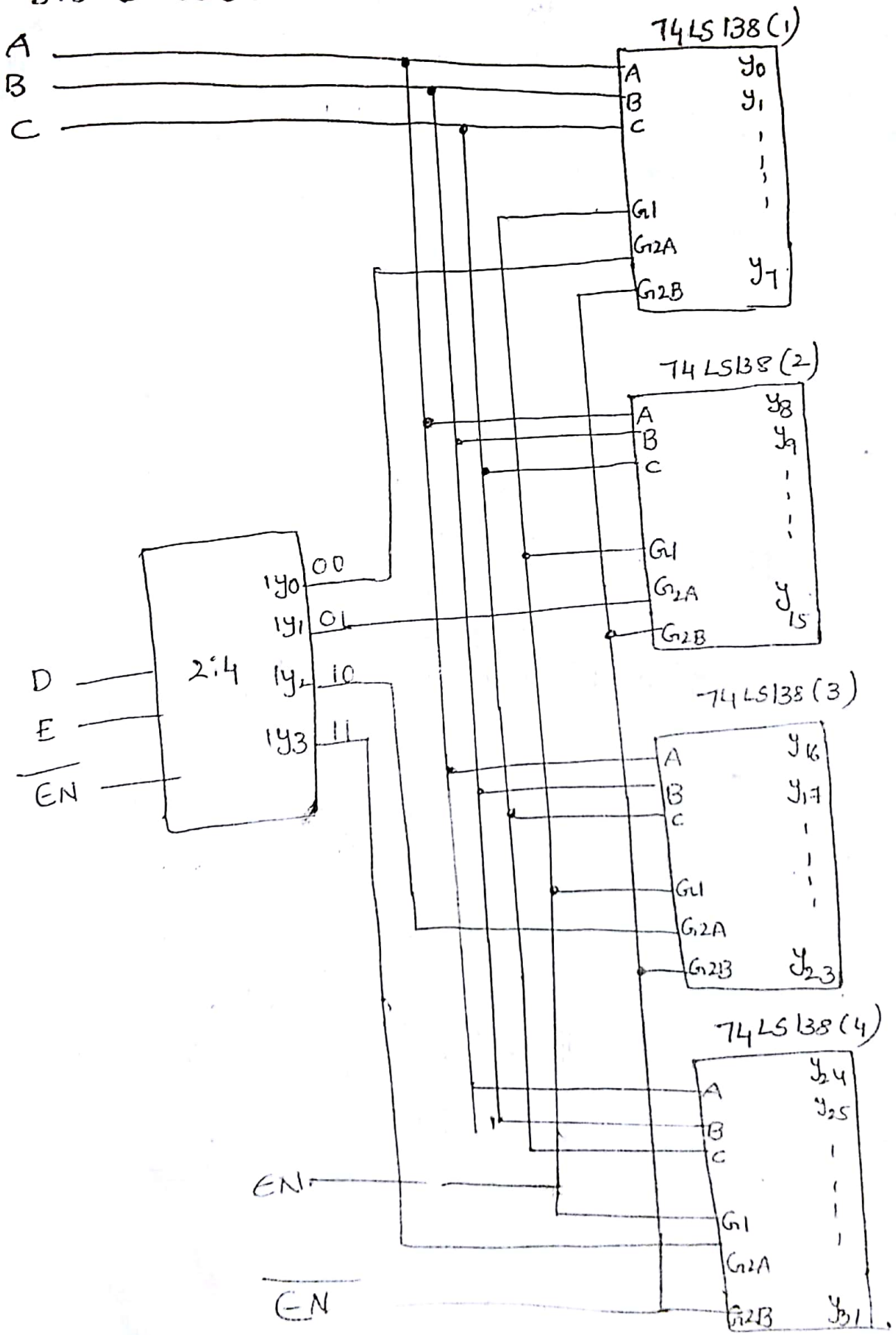
$S(x,y,z) = \Sigma(1,2,4,7)$

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

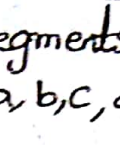
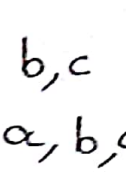
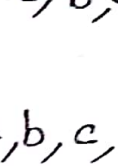
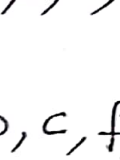
3:8 decoder is needed



⇒ Design 5:32 decoder using one 2:4 & four 3:8 decoders.

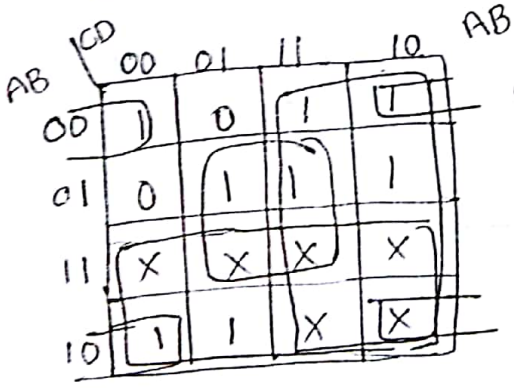


BCD to Seven Segment decoder:-

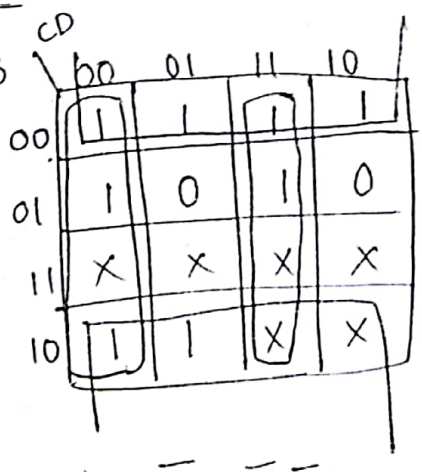
Digit	Segments activated	Display
0	a, b, c, d, e, f	
1	b, c	
2	a, b, d, e, g	
3	a, b, c, d, g	
4	b, c, f, g	
5	a, c, d, f, g	
6	a, c, d, e, f, g	
7	a, b, c	
8	a, b, c, d, e, f, g	
9	a, b, c, d, f, g	

Digit	A	B	C	D	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

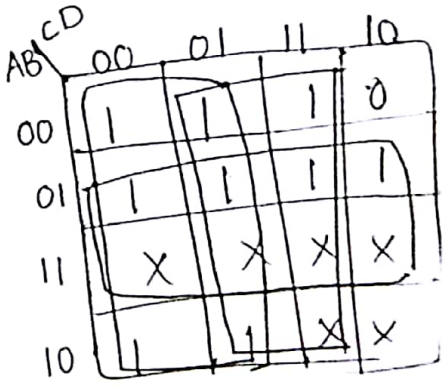
K-map simplification:-



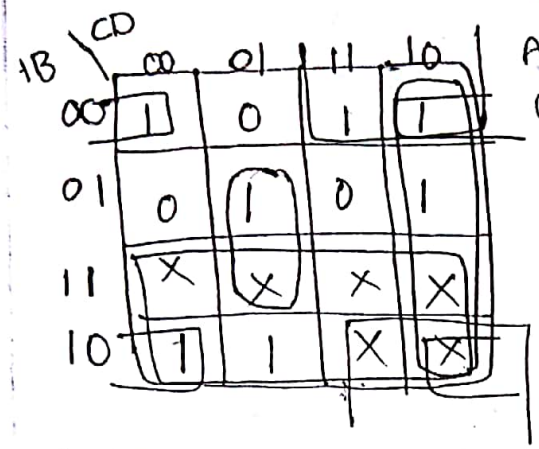
$a = A + C + \overline{B}\,\overline{D} + BD$



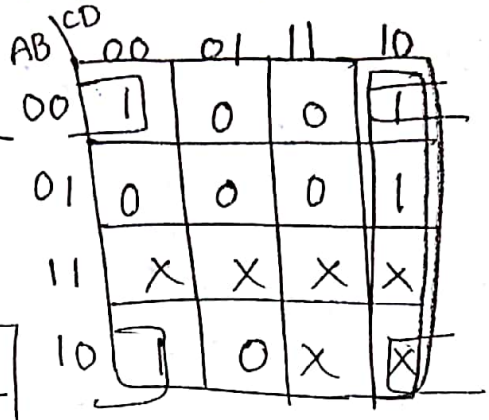
$b = \overline{B} + \overline{C}\,\overline{D} + CD$



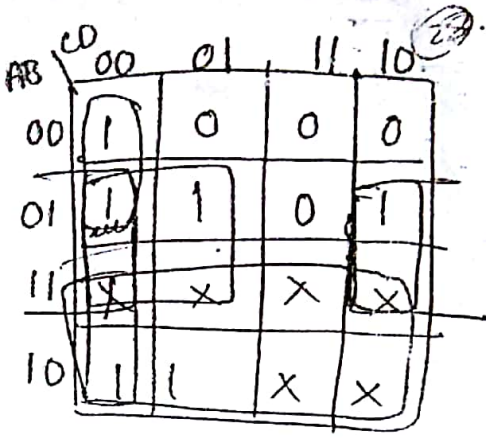
$C = B + \overline{C} + D$



$d = \bar{B}\bar{D} + A + B\bar{C}D + C\bar{D} + \bar{B}C$

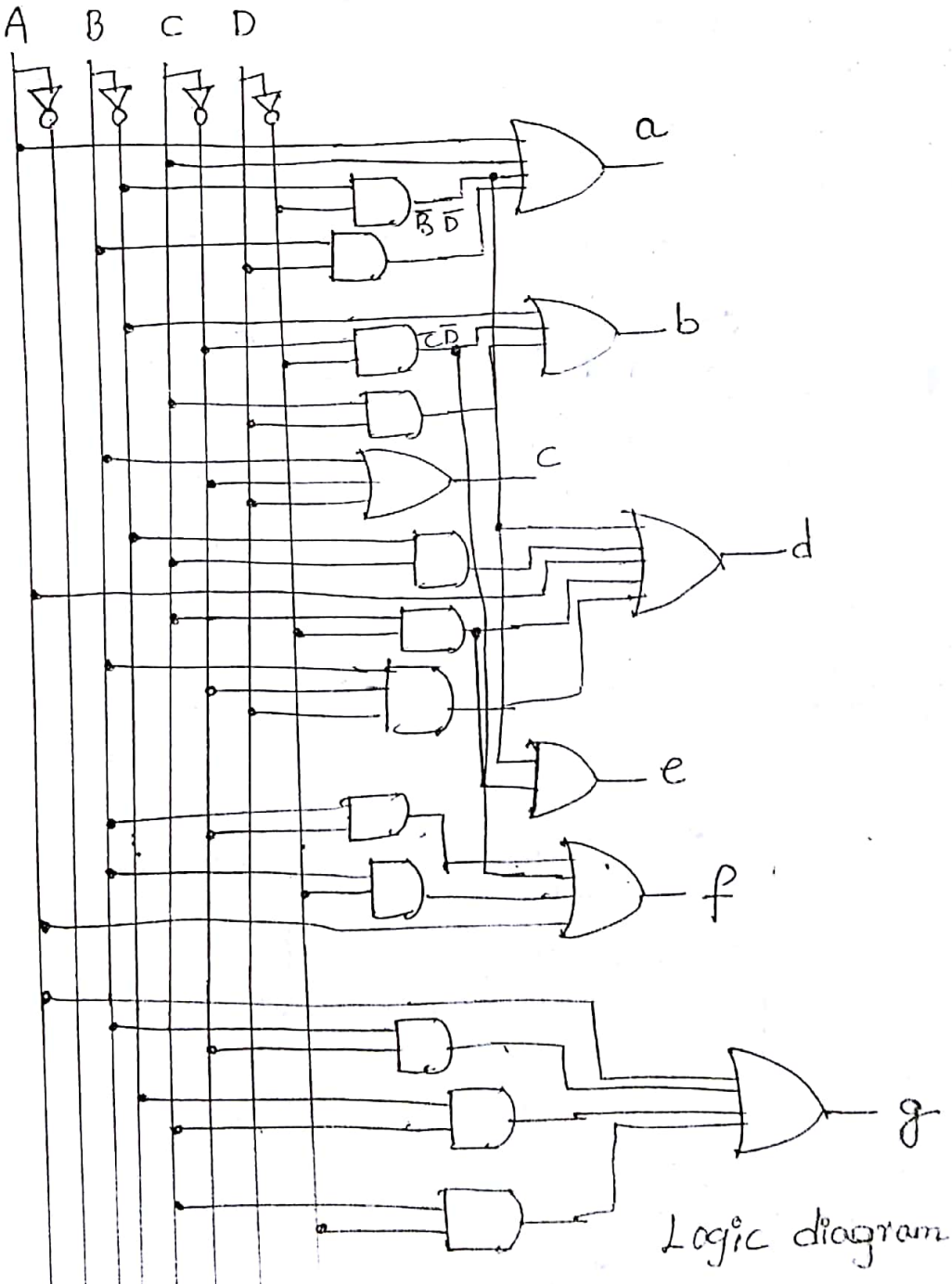


$e = \bar{B}\bar{D} + C\bar{D}$



$f = A + B\bar{C} + \bar{C}\bar{D} + B\bar{D}$

$g = A + B\bar{C} + \bar{B}C + C\bar{D}$

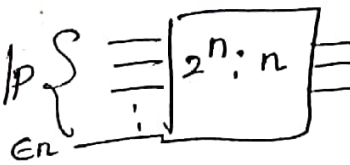


Logic diagram

Encoders :- → An encoder is a digital circuit that performs the inverse operation of a decoder

→ An encoder has 2^n (more) input lines and n ouptput Lines.

→ The output lines generate the binary code corresponding to the input value, so it is called as octal to binary encoder.



8 : 3 encoder :-

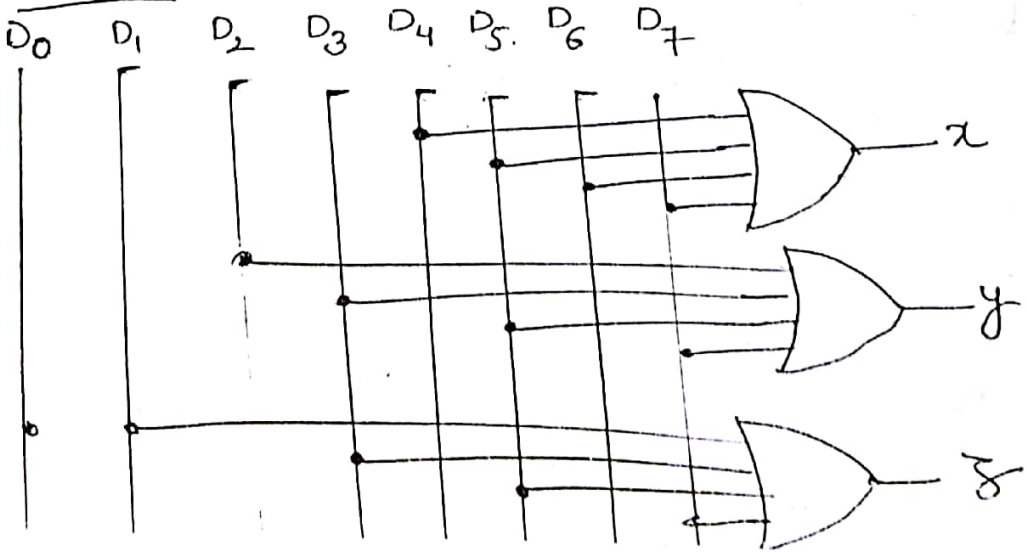
IT :-

Inputs								outputs		
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

$x = D_4 + D_5 + D_6 + D_7$; $y = D_2 + D_3 + D_6 + D_7$

$z = D_1 + D_3 + D_5 + D_7$

Logic circuit :-



Priority Encoder:- → A priority encoder is an encoder circuit that includes the priority function.

→ The operation of the priority encoder is such that if two (or) more inputs are equal to 1 at the same time, the input having the highest priority will take precedence.

↳ D_0 D_1 D_2 D_3

X	X	X	1	1st priority
X	X	1	0	2nd priority

MSB = 1 = Largest no
So highest prior

→ The TT of a 4 i/p priority encoder is shown below: In addition to the two o/p's x and y, the circuit had a 3rd o/p designated by 'V' → valid bit indicator → 1 → when one (or) more i/p's are equal to 1

V → 0 → If all i/p's are '0'.

TT :-

Inputs				outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

→ Instead of listing all 16 minterms of 4 variables, the TT uses an 'X' to represent either 1 (or) 0.

→ For eg: X100 → 0100 & 1100 minterms

→ From TT, the higher the subscript number (MSB), the higher the priority of the input.

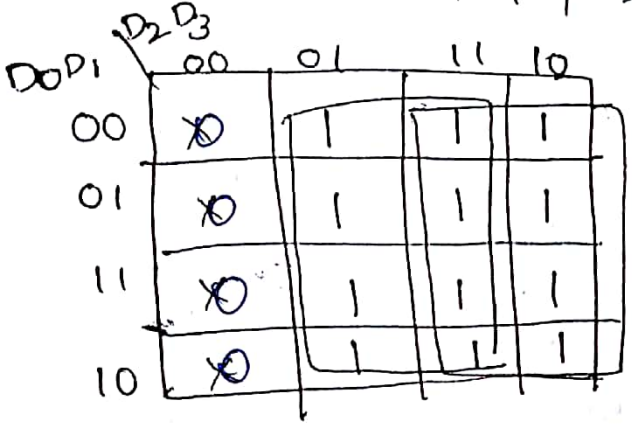
→ I/P D_3 has the highest priority, so, regardless of the values of the other i/p's, when this i/p is 1, the o/p for xy is 11 and so on

k-map :-

$x = \Sigma m(1,2,3,5,6,7,9,10,11,13,14,15) + \Sigma d(0,4,8,12)$

X	X	1	0	
0	0	1	0	= 2
0	1	1	0	= 6
1	0	1	0	= 10
1	1	1	0	= 14

X	X	X	1	
0	0	0	1	= 1
0	0	1	1	= 3
0	1	0	1	= 5
0	1	1	1	= 7
1	0	0	1	= 9
1	0	1	1	= 11
1	1	0	1	= 13
1	1	1	1	= 15



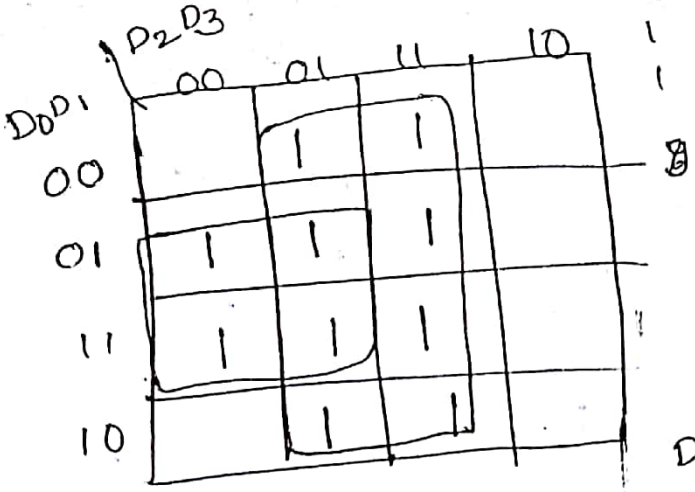
$x = D_2 + D_3$

$y = \Sigma m(1,3,4,5,7,9,11,12,13,15) + \Sigma d(0,2,6,8,10,14)$

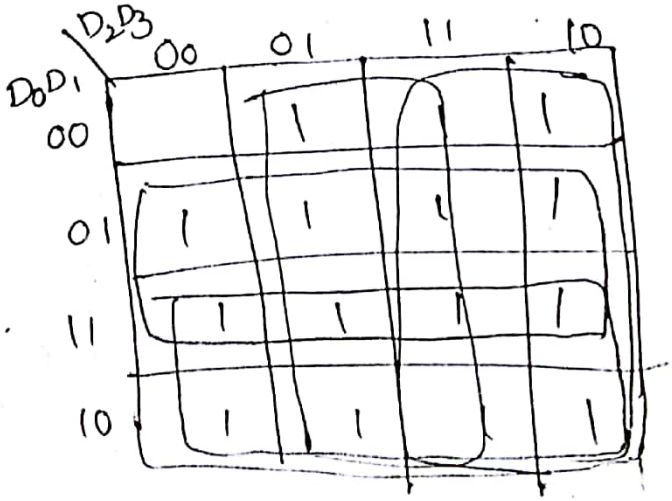
X	1	0	0	
0	1	0	0	= 4
1	1	0	0	= 12

X	X	X	1	
0	0	0	1	= 1
0	0	1	1	= 3
0	1	0	1	= 5
0	1	1	1	= 7
1	0	0	1	= 9
1	0	1	1	= 11
1	1	0	1	= 13
1	1	1	1	= 15

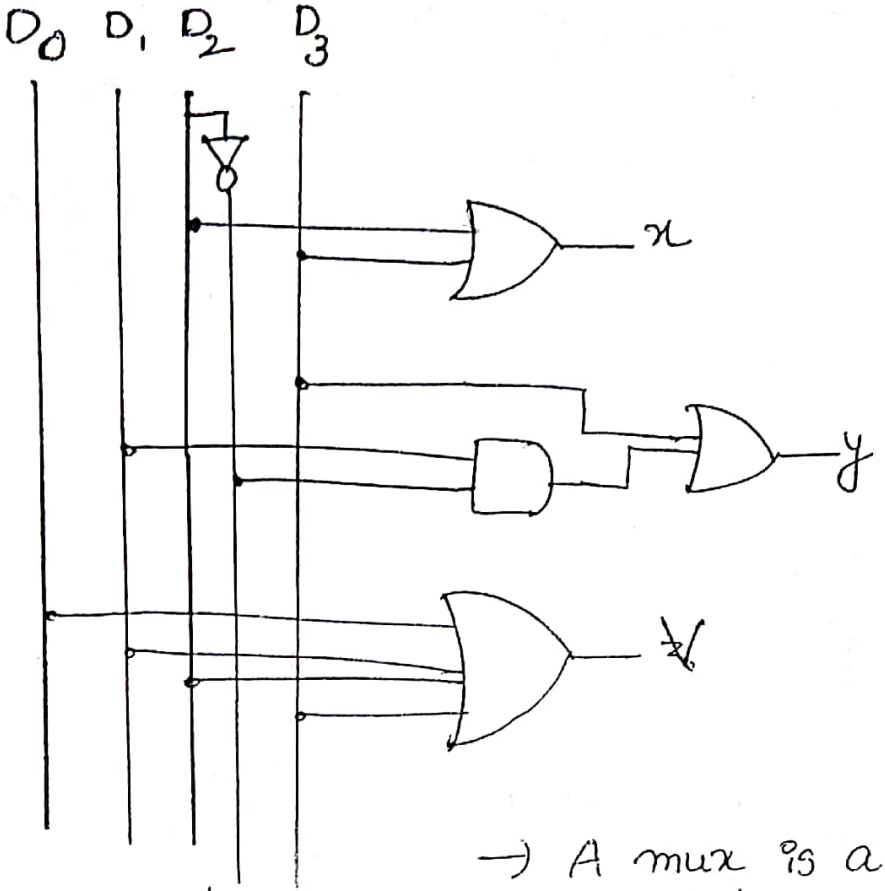
$y = D_3 + D_1\overline{D_2}$



$V = \Sigma m(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15) + \Sigma d(0)$



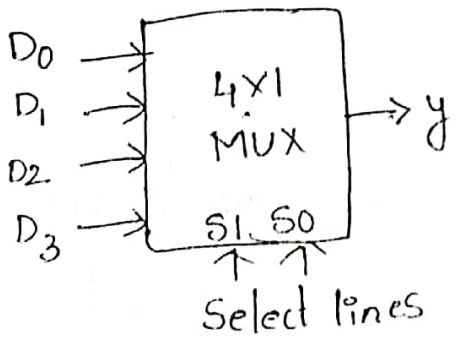
$V = D_2 + D_3 + D_0 + D_1$



Multiplexers :- → A mux is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.

→ The selection of a particular input line is controlled by a set of selection Lines. Normally, there are 2^n input lines and n selection lines

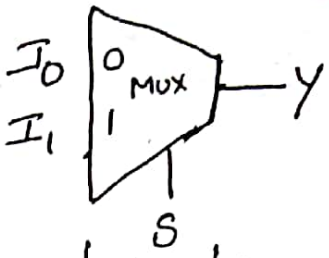
i) 4×1 MUX :- $2^2 \times 1$ = 2 select lines



TT :-

EN	S1	S0	o/p(y)
1	0	0	D_0
1	0	1	D_1
1	1	0	D_2
1	1	1	D_3

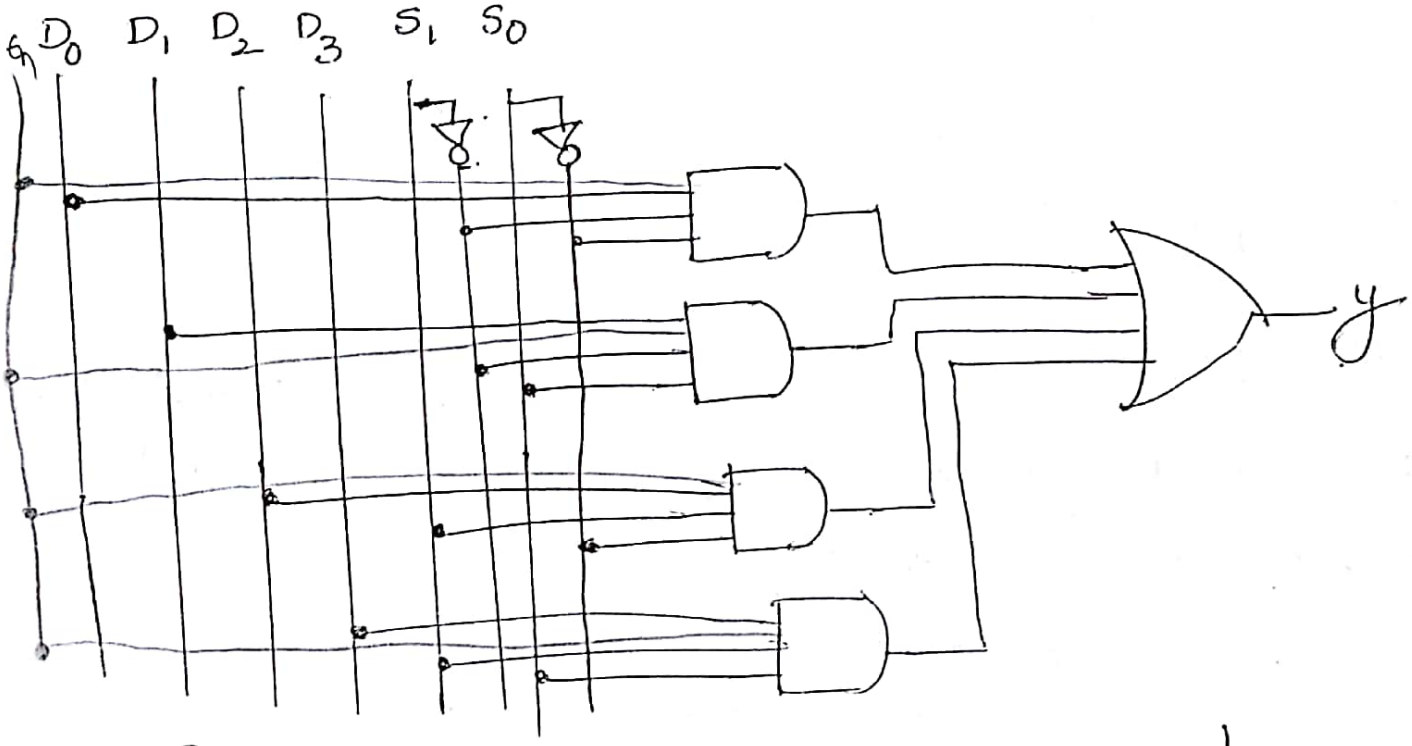
→ when we transferred data from registers to bus bar system



Block diagram

2) 8x1 MUX :-

Logic dgm :-



→ Design 8x1 MUX

→ 32x1 MUX By : 2 (16x1) + 1 (2x1)

3:8 decoder + 8 (4x1) + 1 (8x1)

2:4 decoder + 4(8x1) + 1(4x1)

Boolean function Implementation :-

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

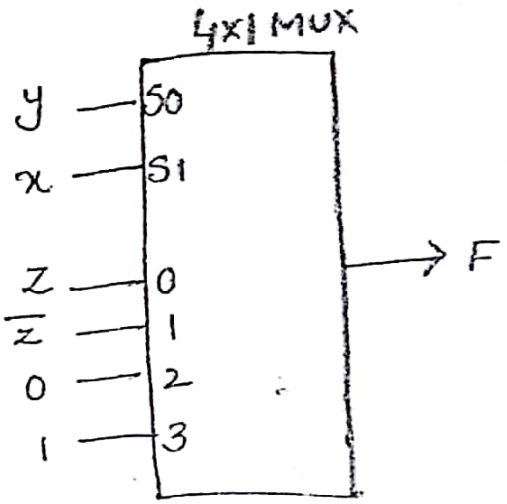
→ To simplify this we have 2 methods

i) Truth table

ii) Implementation table

1) TT :-

S_1 x	S_0 y	i/p z	F(o/p)	
0	0	0	0	$F = z$
0	0	1	1	
0	1	0	1	$F = \bar{z}$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	



Now to get o/p F we apply implementation table.

z \ xy	00	01	10	11
0	0	(2)	4	(6)
1	(1)	3	5	(7)

$\Sigma(1,2,6,7)$

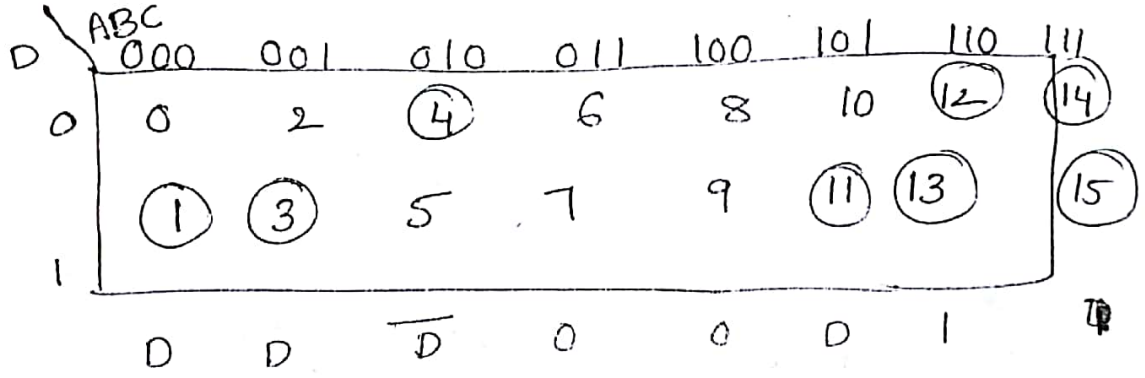
$F = z \quad \bar{z} \quad 0 \quad 1$

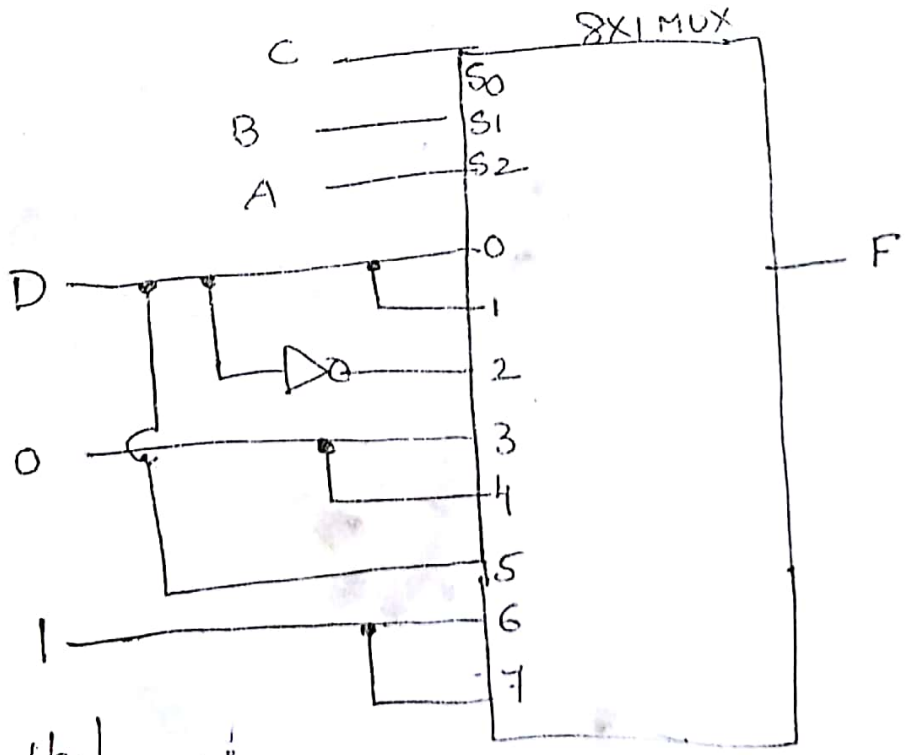
2) $f_{S_1 S_0}(A, B, C, D) = \Sigma(1,3,4,11,12,13,14,15)$

	A	B	C	D	F(o/p)	
	0	0	0	0	0	
0	0	0	0	1	1	$F = D$
	0	0	1	0	0	
1	0	0	1	1	1	$F = D$
	0	1	0	0	1	
2	0	1	0	1	0	$F = \bar{D}$
	0	1	1	0	0	
3	0	1	1	1	0	$F = 0$
	1	0	0	0	0	
4	1	0	0	1	0	$F = 0$

S2 A	S1 B	S0 C	I/P D	F(O/P)	
5	1	0	1	0	0	F=D
	1	0	1	1	1	
	1	1	0	0	1	F=1
6	1	1	0	1	1	
	1	1	1	0	1	F=1
7	1	1	1	1	1	

→ Here 3 selection lines are present now we use an implementation table



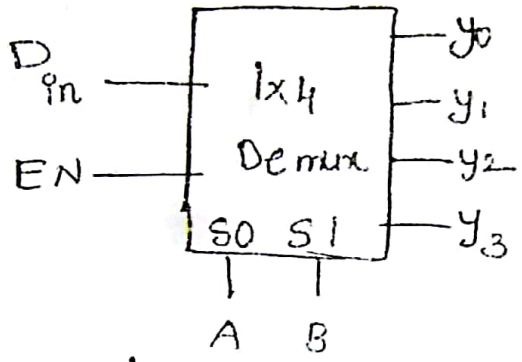


De-multiplexer:

→ Demux is a combinational ckt which performs a reverse operation of MUX. which collects the

information from a single line and transfer into multiple lines.

1x4 Demux :-



TT :-

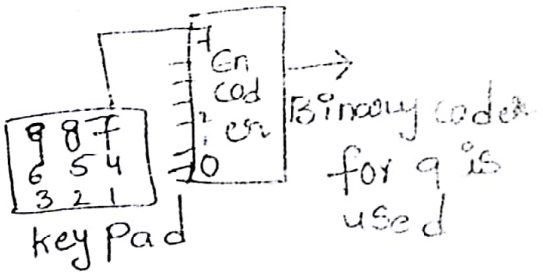
A	B	Din	o/p
0	0	1	Y0
0	1	1	Y1
1	0	1	Y2
1	1	1	Y3

Applica :- → In Security Monitoring system
→ In Synchronous data transmission system

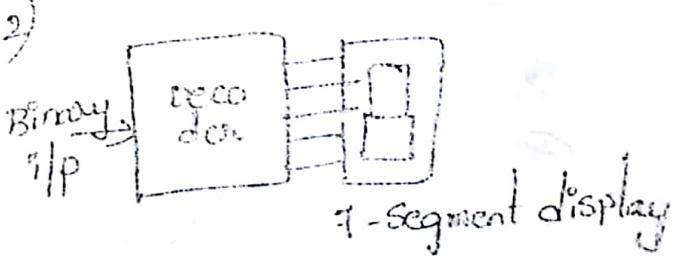
Applic of Mux :-

1) It is used in 7 segment display
2) It can act as a function generator
3) Data selection and date rotating
4) parallel to serial conversion

1)



2)



3)