UNIT - IV

COMBINATIONAL CIRCUITS

Combinational ckt :—

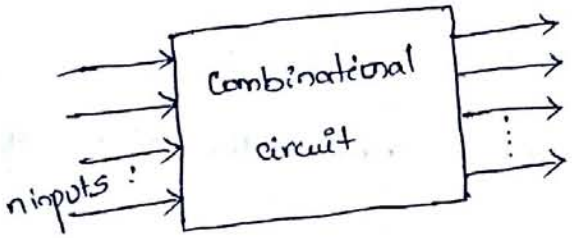
logic ckts for digital systems may be combinational or Sequential.

— A combinational ckt consists of logic gates whose o/p's at any time are determined from the present combination of inputs.

It performs an operation that can be specified logically by a set of Boolean functions.

— It consists of input variables, logic gates and o/p variables.

The logic gate accept signals from the inputs and generate signals to the outputs.



— The n I/p binary variables come from an External source; The m output variables go to an External destination.

— for n input variables, the are 2^n possible o/p values for each combinations. for each possible input combination, there is one possible output value. Thus a Combinational ckt can be specified with a truth table that lists the o/p values for each combination of input variables.

Analysis procedure :—

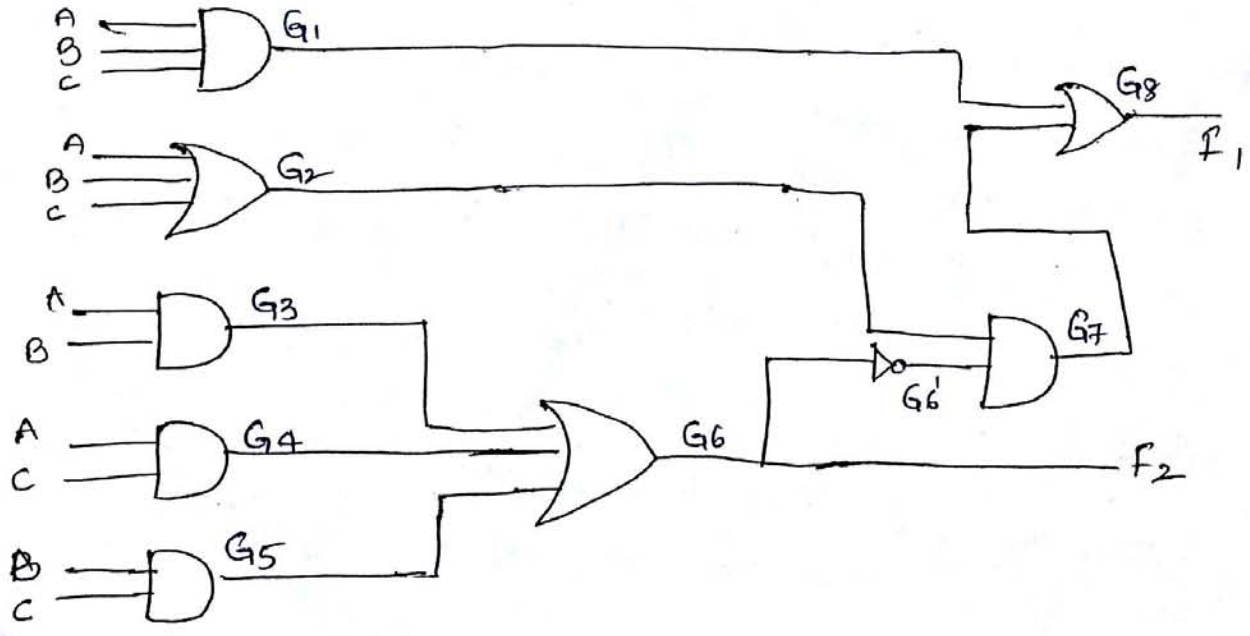
— The first step in the analysis is to make sure that the is combinational and not a Sequential.

— The diagram of a combinational ckt has logic gates with paths or memory Elements. A feedback path is a connec from the o/p of one gate to the input of a second ga forms part of the input to the first stage.

— Once the logic diagram is verified as a combinational ckt one can proceed to obtain o/p Boolean function or truth table.

— To obtain the o/p boolean function from a logic diagram, proceed as follows.

1. Label all gate o/ps that are a function of i/p variables with arbitary symbols. Determine the boolean functions for Each gate o/p.
2. label the gates that are a fn of input variables and previously labeled gates with other arbitary symbols.
3. Repeat the process of step2 until the o/ps of the ckts are obtained.
4. By repeated substitution of previously defined functions, obtain the o/p boolean functions in terms of input variables.



$G_1 \rightarrow ABC \quad G_2 \rightarrow A+B+C \quad G_3 = AB \quad G_4 \rightarrow AC \quad G_5 \rightarrow BC$

$G_6 \rightarrow AB+AC+BC = F_2$

$G(6)' \rightarrow (AB+AC+BC)' = \overline{AB} \cdot \overline{AC} \cdot \overline{BC}$

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

$G_7 \rightarrow (A+B+C)(\bar{A}+\bar{B})(\bar{A}+\bar{C})(\bar{B}+\bar{C})$

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

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

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

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

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

$G_8 \rightarrow \bar{A}B\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + ABC$

— To obtain the truth table directly from logic diagram through the derivation of boolean functions.

1. Determine the no. of i/p variables in ckt. for n-inputs, there possible i/p combination and list the binary numbers from 0 to $2^n - 1$.
2. Label the o/ps of selected gates with the arbitary symbols.
3. obtain the truth table for o/ps of those gates that are a function of i/p variables only.
4. proceed to obtain the T.T for o/ps of those gates that are fn of previously defined values until columns for all o/ps are determined.

A	B	C	G_1	G_2	G_3	G_4	G_5	G_6	G_6	G_7	G_8
0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	0	1	0	0	0	0	1	1	1
0	1	0	0	1	0	0	0	0	1	1	1
0	1	1	0	1	0	0	1	1	0	0	0
1	0	0	0	1	0	0	0	0	1	1	1
1	0	1	0	1	0	0	0	0	1	0	0
1	1	0	0	1	1	0	0	1	0	0	0
1	1	1	1	1	1	1	1	1	0	0	1

Design procedure:

The design procedure starts from the specification of the problem and ends up with a logic diagram or a set of boolean functions from which the logic diagram can be obtained.

1. From the specifications of the ckt, determine the required no. of inputs and o/p's and assign a symbol to each.
2. Derive the truth table that defines the required relationship b/w inputs and outputs.
3. obtain the simplified boolean functions for each o/p as a fn of i/p variables.
4. Draw the logic diagram and verify the correctness of the design.

- A truth table for a combinational ckt consists of input columns and o/p columns.
- Input columns are obtained from the 2^n binary numbers for the n input variables. The binary values for the o/p's are

termined from the Stated Specifications.

- o/p functions Specified in the truth table give the Exact definition of Combinational ckt.

- The o/p binary functions listed in the truth table are simplified by any available method Such as algebric manipulation, the map method.

- The o/p binary functions listed in the truth table are Simplified by any available method Such as algebric manipulation, the map method.

Code Conversion Example

- code converter is a ckt that makes the two systems Even though Each uses a different binary code.

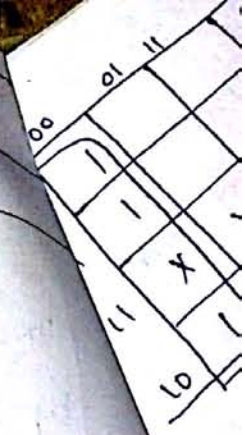
Ex: BCD to Excess-3 code

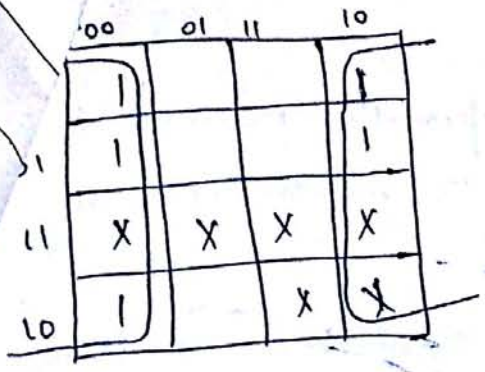
Input is BCD and the o/p is Excess-3 code. The i/p BCD has numbers Starting from 0 to 9 which uses 4-bits to represent. The Excess-3 code for 0 is 3, 1 is 4, ... 9 is 12 which is also represented by 4 bits. So, therefore, we need 4 input as well as 4 output Variables.

- Designate four i/p binary Variables by A, B, C, D and four o/p by w, x, y & z.

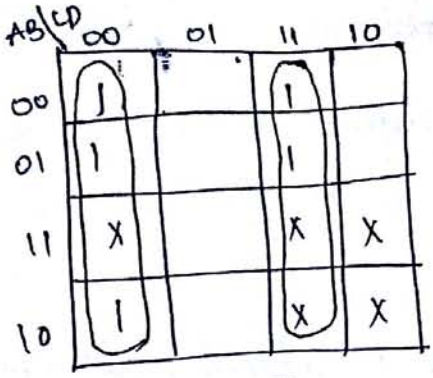
Input BCD				Output Excess-3 code			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

These are don't Care Conditions for BCD.

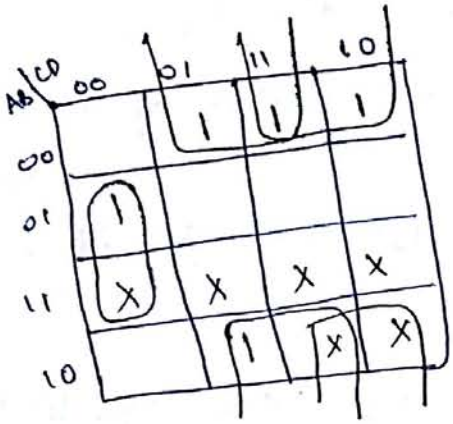




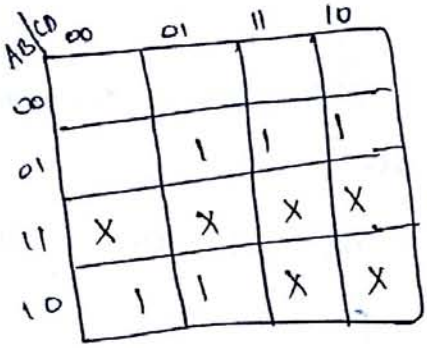
$Z = \bar{D}$



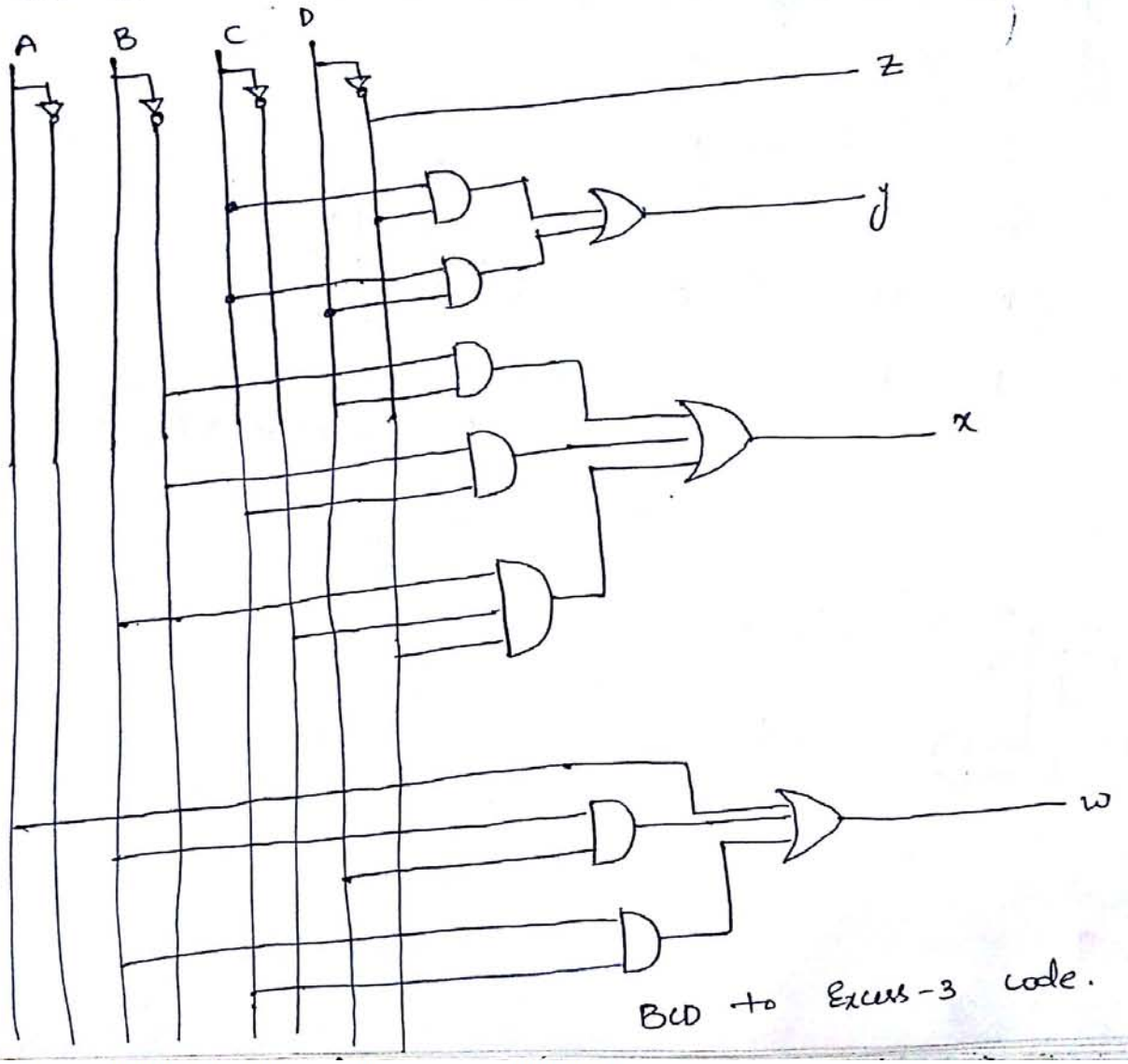
$y = \bar{C}\bar{D} + CD$



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



$W = A + BD + BC$



BCD to Excess-3 code.

BINARY ADDER - SUBTRACTOR

— Digital Computers perform a variety of information processing. The most basic arithmetic operation is the addition of two binary digits.

— When both augend & addend bits are Equal to 1, the binary Sum Consists of two digits. The higher Significant bit of this result is Called a Carry.

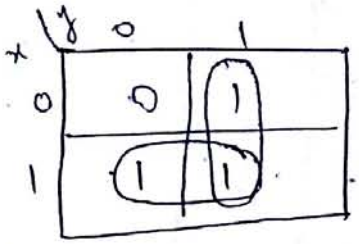
— A Combinational ckt that performs the addition of two bits is called a half adder.

One performs the addition of three bits is a full adder.

Half adder

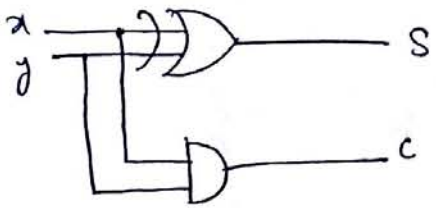
The ckts needs two binary i/p's & two binary o/p. Let the i/p's are x & y and o/p are S & C.

x	y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1



$S = xy' + x'y = x \oplus y$.

$C = xy$.

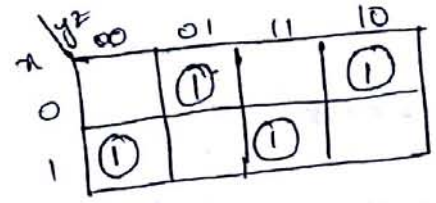


Full adder:

Full adder is a combinational ckt performs the sum of three bits. It consists of three inputs and two o/p's. Two of i/p's are x and y, represents two significant bits to be added. The third i/p, z represents the carry from previous LSB. The two o/p's are S & C.

— The S o/p is Equal to 1 when only one i/p is Equal to 1 or When all three i/p's are Equal to 1. The C o/p has carry to 1 if two or three i/p's are Equal to 1.

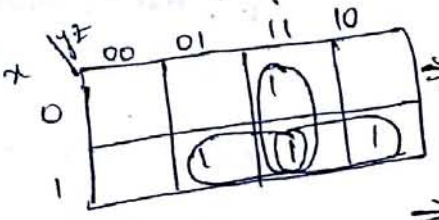
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



$S = x'y'z + x'yz' + xy'z' + xyz$

$= x \oplus y \oplus z$.

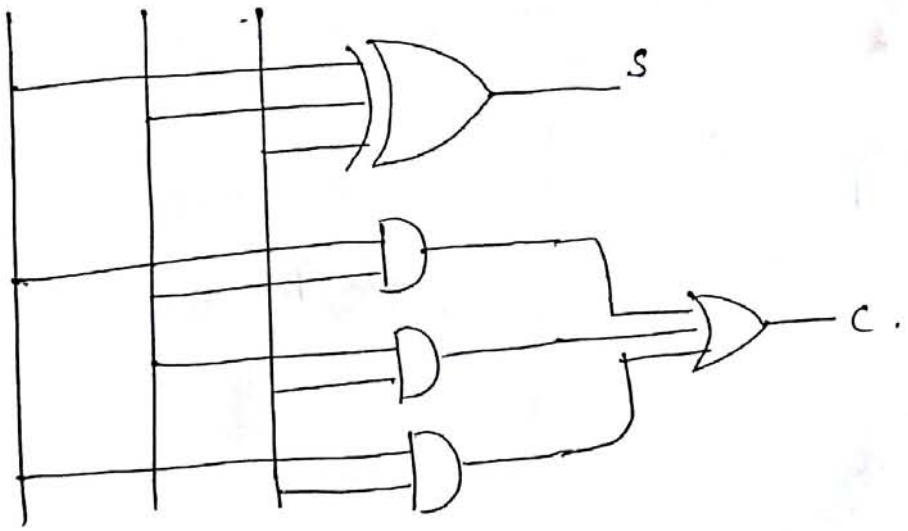
$x'(yz + yz')$
$+x(y'z' + yz)$



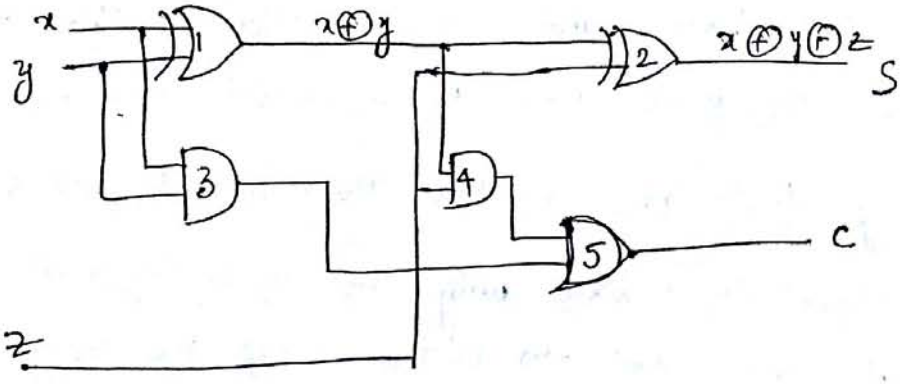
$\Rightarrow x'(y \oplus z)$
$+x(y \odot z)$

$C = yz + xy + xz$ $\Rightarrow x'(y \oplus z) + x(\overline{y \oplus z})$

$= x \oplus y \oplus z$



Implementation of F.A with two H.A and an OR Gate:



1) $x \oplus y$ 2) $x \oplus y \oplus z$ 3) xy 4) $(x \oplus y)(z) = (x'y + xy')z$

$= x'yz + xy'z$

5) $(x'yz + xy'z) + xy$

$x'yz + xy'z + xy$

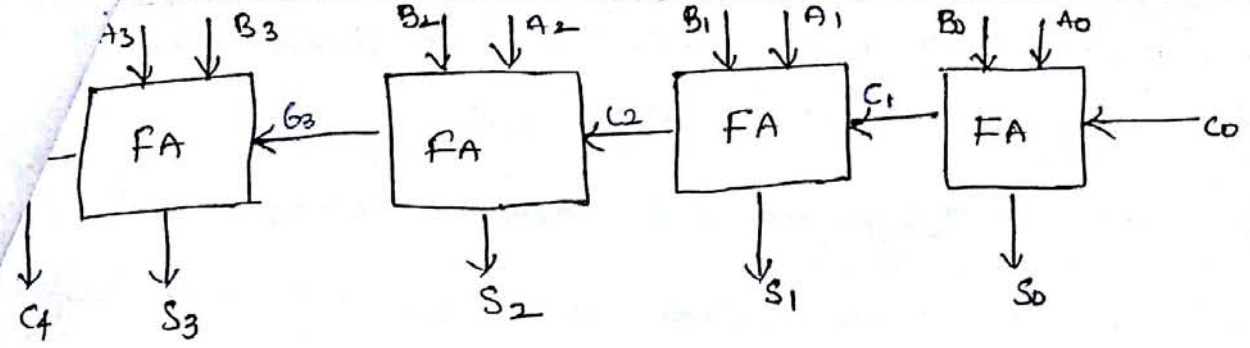
$\sum m(3, 5, 7, 6)$

ii) Binary Adder:- (parallel Binary Adder)

A binary adder is a digital ckt that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascade, with the o/p carry from each F.A connected to the i/p carry to the next FA.

Let two binary numbers A = 1011₂ B = 0011

Subscript i:	3	2	1	0	
I/p carry	0	1	1	0	C_i
Augend	1	0	1	1	A_i
Addend	0	0	1	1	B_i
Sum	1	1	1	0	S_i
O/p Carry	0	0	1	1	C_{i+1}



The bits are added with F.A, Starting from least Significant position, to form the sum & carry. The i/p carry Co in the LSB must be 0. The value of C_{i+1} in a given Significant position is the o/p carry of f.A. This is transferred into the i/p carry of f.A that adds the bits one higher Significant position to left.

Extra

Carry propagation

- The add of two binary numbers in parallel implies that all the bits of the augend and addend are available for computation at the same time. In any Comb. ckt, the s/g must propagate through the gates before the correct o/p sum is available in o/p terminals.
- The total propagation time is Equal to the propagation delay of typical gate times and the no. of gate levels in the ckt.
- Inputs A_3 & B_3 are available as soon as i/p s/gs are applied. However, C_3 doesnot settle to its final value until C_2 is available from the previous stage.

The no. of gate levels for the carry propagation can be found from the ckt of the F.A. The i/p & o/p variables use the Subscript i to denote a typical Stage in the adder.

The Signals at p_i & G_i Settle to their steady state after they propagate through their respective gates.

— A comb. ckt will always have some value at its o/p terminals o/p's will not be correct unless the signals are given enough time to propagate through the gates connected from i/p's to o/p's.

As, the arithmetic operations are implemented by successive additions time consumed during the addition process is very critical.

for reducing the carry propagation delay time, employ faster gates with reduced delays.

Consider the F.A designed with two H.A Define two new binary Variables.

$P_i = A_i \oplus B_i$ & $G_i = A_i B_i$ G_i - Carry generate

$S_i = P_i \oplus C_i$, $C_{i+1} = G_i + P_i C_i$ P_i - Carry propagate.

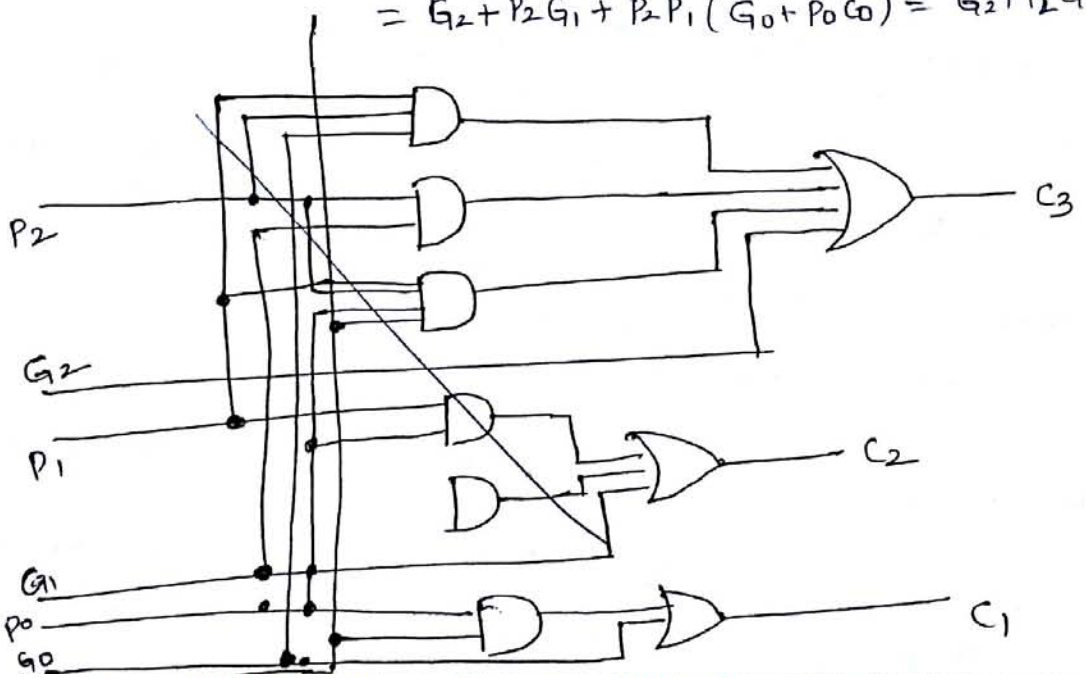
C_0 = i/p Carry.

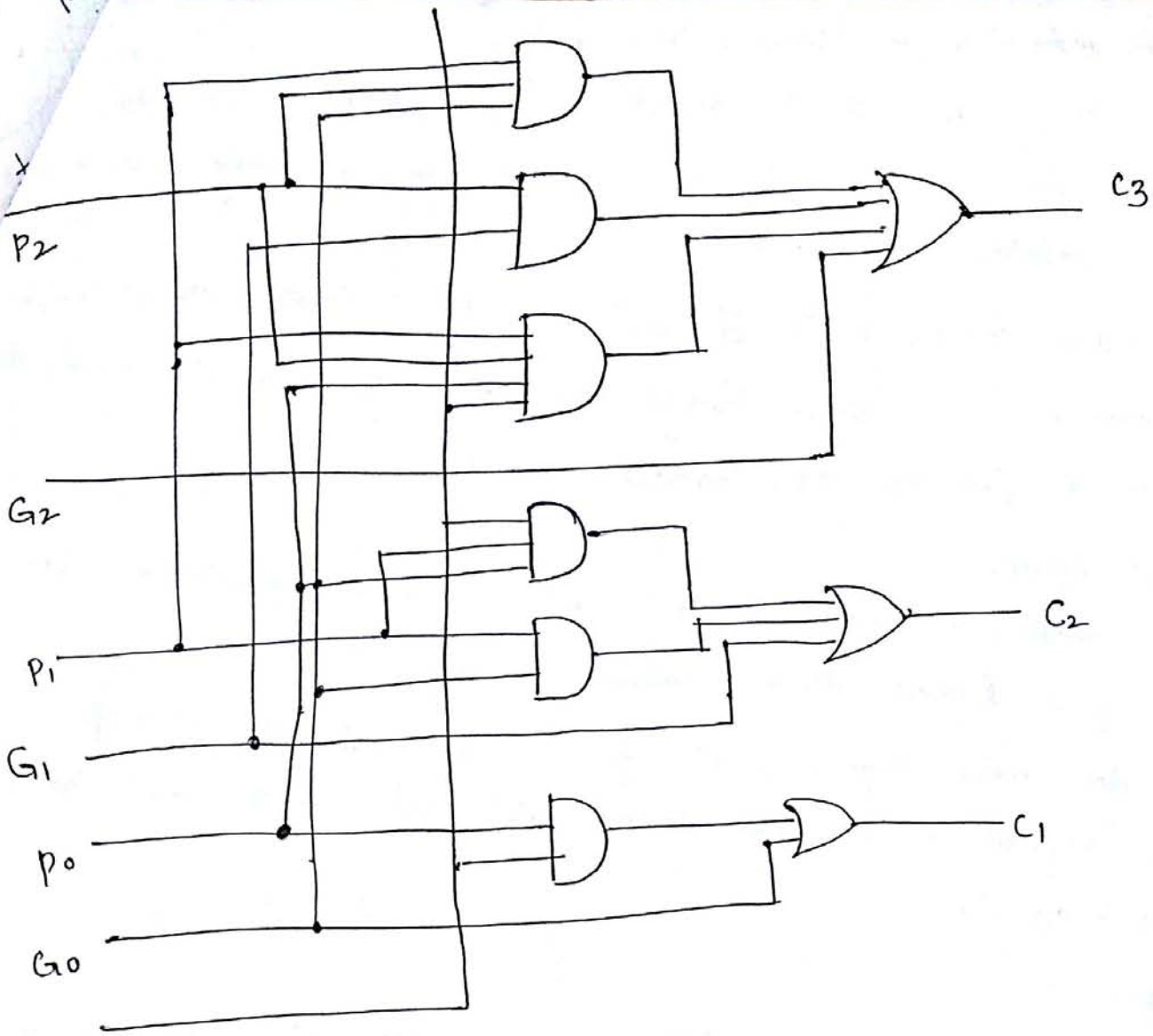
$C_1 = G_0 + P_0 C_0$

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

$C_3 = G_2 + P_2 C_2 = G_2 + P_2 (G_1 + P_1 C_1) = G_2 + P_2 G_1 + P_2 P_1 C_1$

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

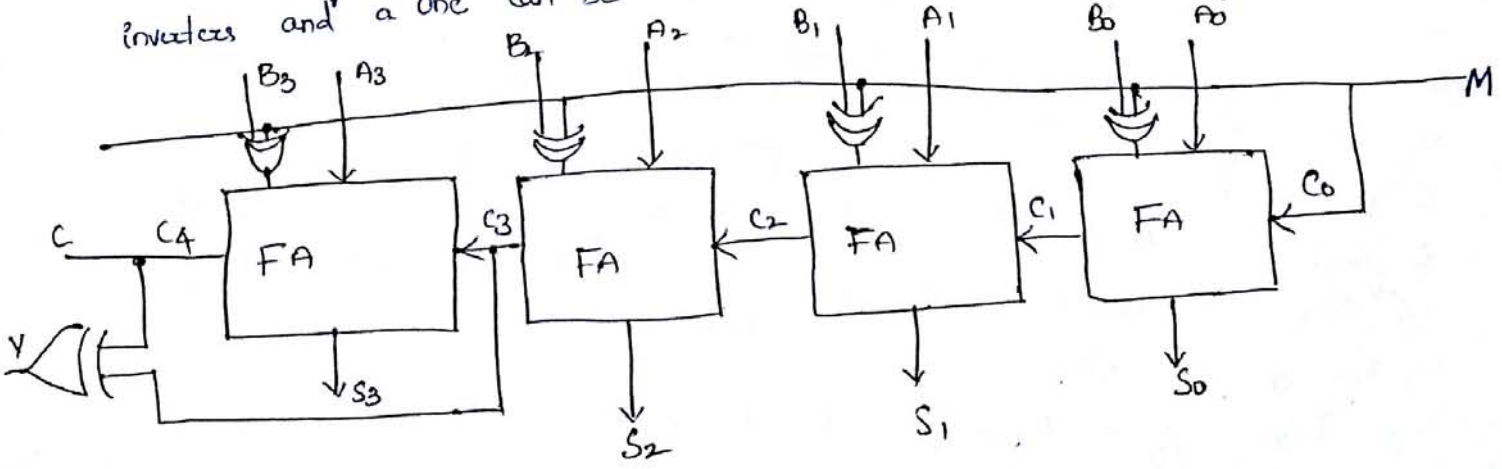




Extra Adder/

Binary Subtractor: The Subtraction of unsigned binary numbers can be done by means of complements.

A-B can be done by taking 2's complement of B and adding it to A. The 2's complement can be obtained by taking the 1's comp and adding one to LSB. The 1's complement can be obtained with inverters and a one can be added to the Sum through the i/p Carry.



— The mode i/p m controls the operation.

When m=0, the ckt is an adder, because when m=0, B⊕

The full adder receive the value of B, the i/p Carry is 0,

the ckt performs A plus B.

When m=1, B⊕1 = B' and Co=1. The B inputs are all Compl

nted and a 1 is added through the i/p Carry. The ckt performs the

operation A plus the 2's Complement of B.

Decimal Adder:

Computers or Calculators that perform arithmatic operation in decimal number System represent decimal numbers in binary coded form

— A dec. adder requires a min of nine i/p's and five Since four bits are required to code Each decimal digit and the ckt must have an i/p & o/p Carry.

BCD Adder:

Consider the arithmatic addition of two dec digits in BCD, togeth with an i/p Carry from a previous stage.

If we apply two BCD digits to a 4-bit binary adder. It will form Sum in binary and produce a result that ranges from 0 to 19.

Binary Sum					BCD Sum					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	1	6
0	0	1	1	1	0	1	0	0	0	7

Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	Decimal
1	0	0	0	0	1	0	0	0	8
1	0	0	1	0	1	0	0	1	9
1	0	1	0	0	1	0	1	0	10
1	0	1	1	0	1	0	1	1	11
1	1	0	0	0	1	1	0	0	12
1	1	0	1	0	1	1	0	1	13
1	1	1	0	0	1	1	1	0	14
1	1	1	1	0	1	1	1	1	15
1	0	0	0	1	0	1	1	0	16
1	0	0	1	1	0	1	1	1	17
1	0	1	0	1	1	0	0	0	18
1	0	1	1	1	1	0	0	1	19

- The columns under the binary Sum list the binary value that appears in the o/p's of the 4-bit binary adder. The o/p of two decimal digits must be represents in BCD and should appear in the form listed in the columns under BCD Sum.

From the table, it is apparent that when the binary Sum is Equal to or less than 1001, the corresponding BCD number is identical, and therefore no Conversion is needed.

When the binary Sum is >1001, we obtain a non-valid BCD. The addition of 6 (0110) to the binary Sum converts it to the Correct BCD representation and produces an o/p Carry.

The Correction is needed when binary Sum has an o/p Carrey

K = 1

The other Six combinations from 1010 through 1111 that need have a 1 in position Z_8. To distinguish them from binary 10 which also have a 1 in Z_8, we Specify further either Z_4 or must have a 1. The condition for a correction & an o/p Carry can Expressed by the Boolean fn.

$$C = K + Z_8Z_4 + Z_8Z_2.$$

When C=1, it is neccessary to add 0110 to the binary Sum & provide an o/p Carry for the next stage.

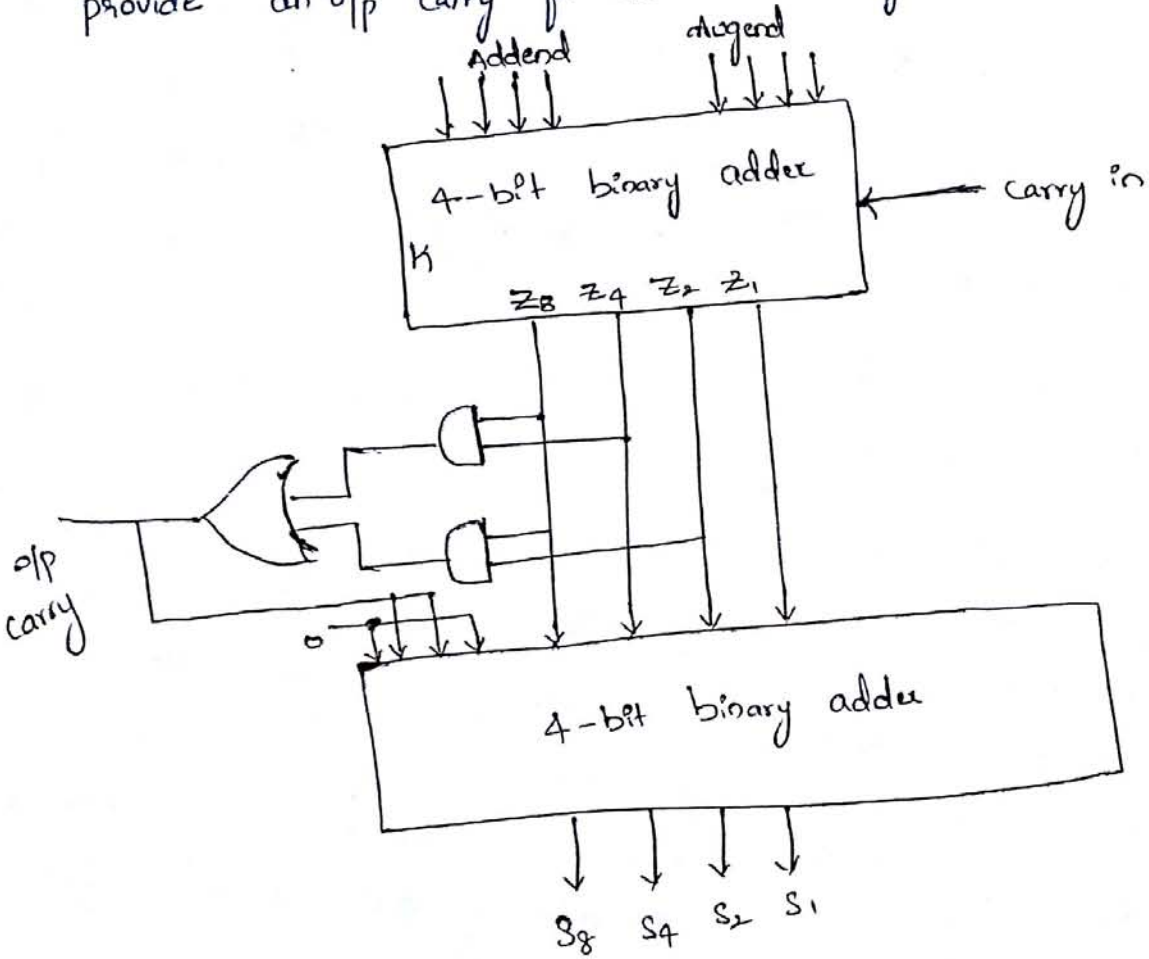


Fig: BCD adder.

Extra

Multiplier

The multiplicand is multiplied by each bit of the multiplier starting from LSB. Each such multiplication forms a partial product. Successive partial products are shifted one position to the left. The final product is obtained from the sum of the partial products

Consider the multiplication of two 2-bit numbers. The multiplicand bits are B_1 and B_0, multiplier bits are A_1 and A_0 and the product is

$C_3\ C_2\ C_1\ C_0$

$B_1\ B_0$

$A_1\ A_0$

$A_0B_1\ \ A_0B_0$

$A_1B_1\ \ A_1B_0$

$C_3\ \ C_2\ \ C_1\ \ C_0$

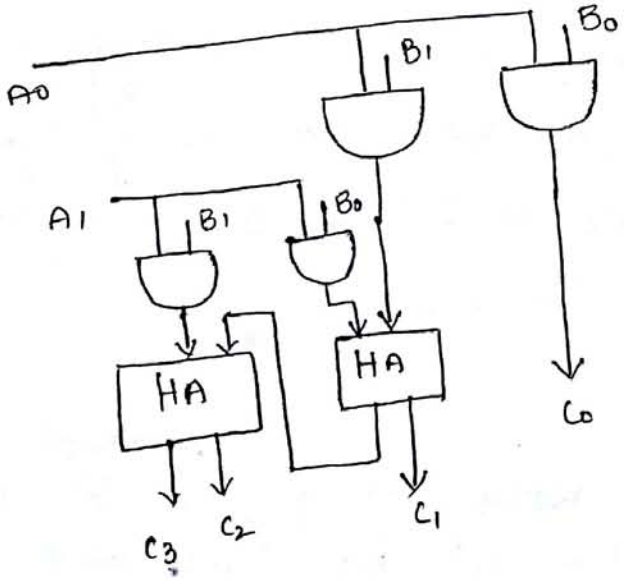


Fig: 2-bit Binary multiplier.

better understanding of boolean laws, rules and theorems.

Magnitude Comparator :

A magnitude Comparator is a comb ckt that compares two A and determines their relative magnitudes. The outcome of the comparision specified by three binary variables that indicate wheather A>B, A=0 or

Let the two numbers, A and B, with four digits Each.

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

The two numbers are Equal if all pairs of significant digits are Equal. $A_3 = B_3$ and $A_2 = B_2$ and $A_1 = B_1$ and $A_0 = B_0$. When the numbers are binary, the digits are either 1 or 0, and the Equality relation of Each pair of bits Can be Expressed logically with an Ex-NOR fn as

$$x_i = A_i B_i + A_i' B_i' \quad \text{for } i = 0, 1, 2, 3.$$

$x_i = 1$ only if the pair of bits in position i are Equal.

The Equality of two numbers, A and B is displayed in a comb ckt by an o/p binary variable that we designate by Symbol (A=B). for the Equality condition to Exist, all x_i variables must be=1 This dictates an AND operation of all variables.

$$(A=B) = x_3 x_2 x_1 x_0$$

- To determine if A> or <B, the relative mag of pairs of Significant digits starting from msB. If two digits are Equal compare the next lower Significant pair of digits. This comparision Continuous until a pair of unEqual digits is Reached.

$$(A>B) = A_3 B_3' + x_3 A_2 B_2' + x_3 x_2 A_1 B_1' + x_3 x_2 x_1 A_0 B_0'$$

$$(A<B) = A_3' B_3' + x_3 A_2' B_2' + x_3 x_2 A_1' B_1 + x_3 x_2 A_0' B_0.$$

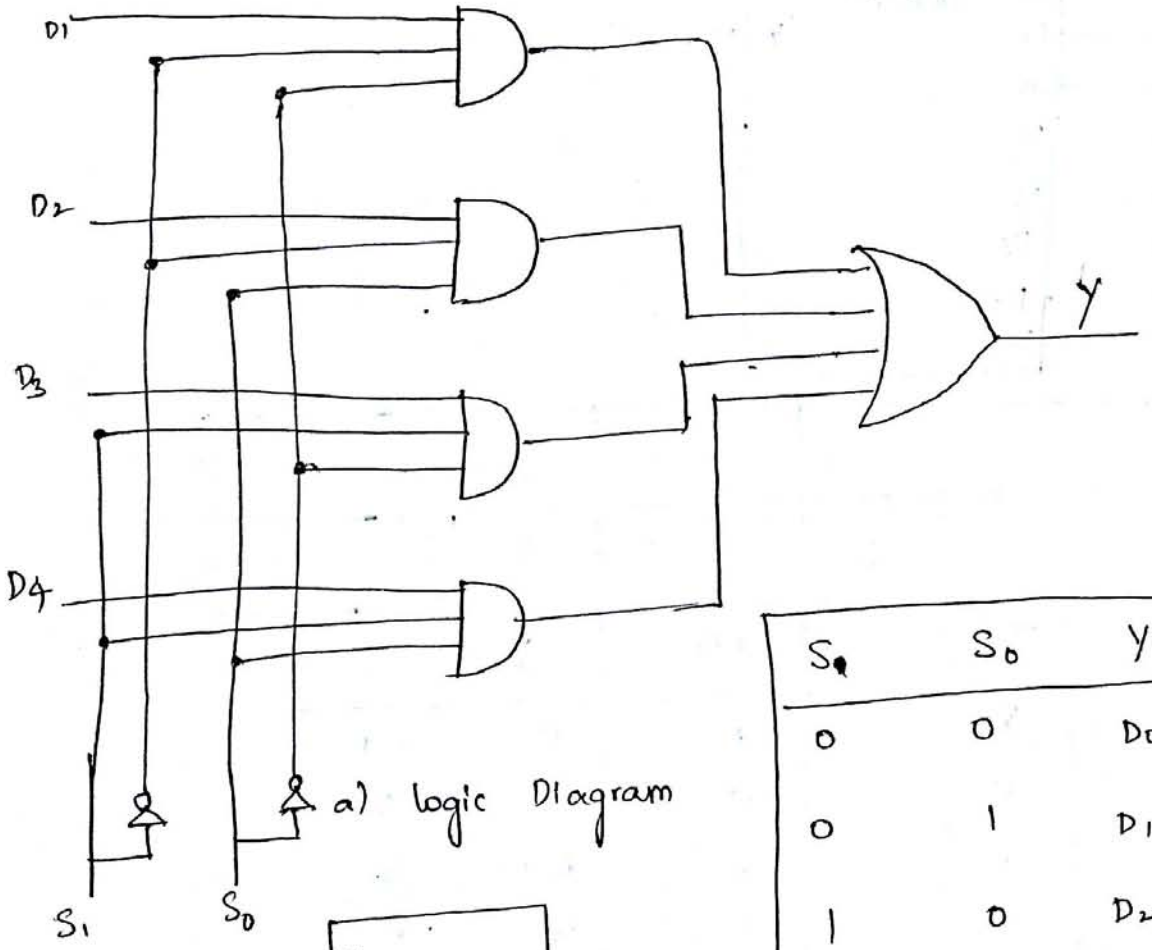
Multiplexers:

It is a digital switch, which allows digital information from several sources to be routed onto a single output line.

- It consists of several data-input lines and a single o/p 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 whose bits combinations determine which input is selected.

4 to 1 line multiplexer:

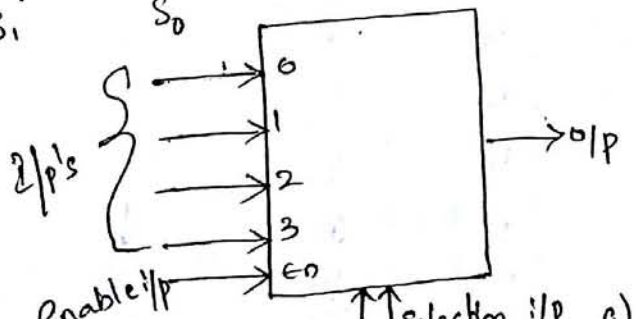
- It consists of four lines, D_1 to D_3, is applied to one input of an AND gate. Selection lines are decoded to select a particular AND gate.



a) logic Diagram

S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

b) functional table

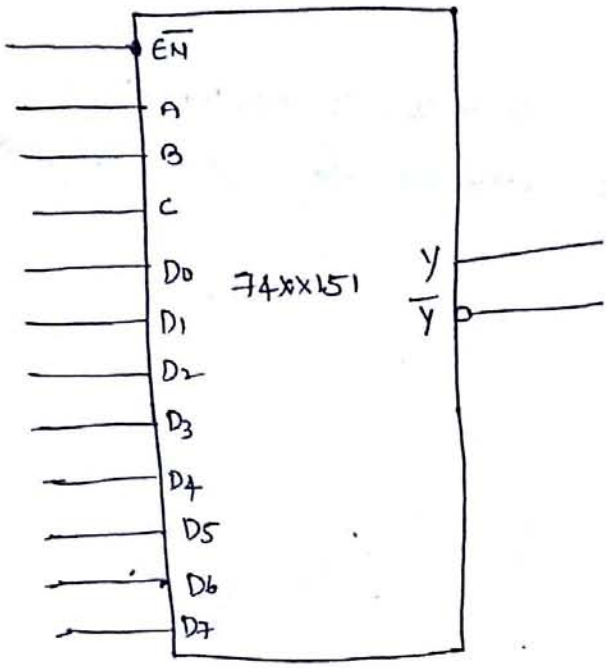


c) logic Symbol

74X151 8 to 1 multiplexers

— It is a 8 to 1 multiplexer. It has eight inputs. It provides two o/p's, one is active high, the other is active low. There are Select lines C, B, A which Selects one of the eight inputs.

In this o/p is not specified in 1s and 0s. Because, multiplexer is a data switch, it does not generate only data of its own, but it simply passes External data, input to the o/p.

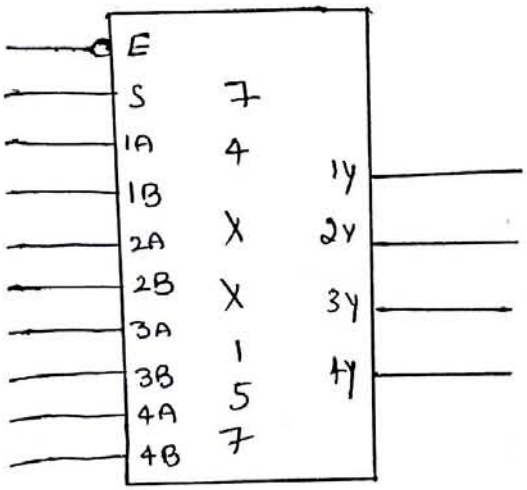


Inputs				outputs	
Select			Enable	y	$\bar{y}$
C	B	A	$\overline{EN}$	y	$\bar{y}$
X	X	X	1	0	1
0	0	0	0	D_0	$\overline{D_0}$
0	0	1	0	D_1	$\overline{D_1}$
0	1	0	0	D_2	$\overline{D_2}$
0	1	1	0	D_3	$\overline{D_3}$
1	0	0	0	D_4	$\overline{D_4}$
1	0	1	0	D_5	$\overline{D_5}$
1	1	0	0	D_6	$\overline{D_6}$
1	1	1	0	D_7	$\overline{D_7}$

74XX157 Quad 2-input multiplexer:-

It Selects four bits of data from two Sources under the control a Common Select input (S)

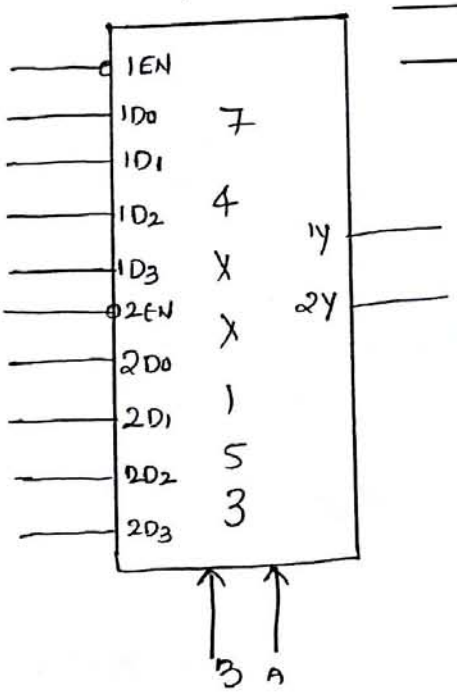
— The Enable input ($\bar{E}$) is active low. when $\bar{E}$ is high, all of the o/p's (Y) are forced low Regardless of all other input conditions.



Inputs		Outputs			
$\bar{E}$	S	1y	2y	3y	4y
1	X	0	0	0	0
0	0	1A	2A	3A	4A
0	1	1B	2B	3B	4B

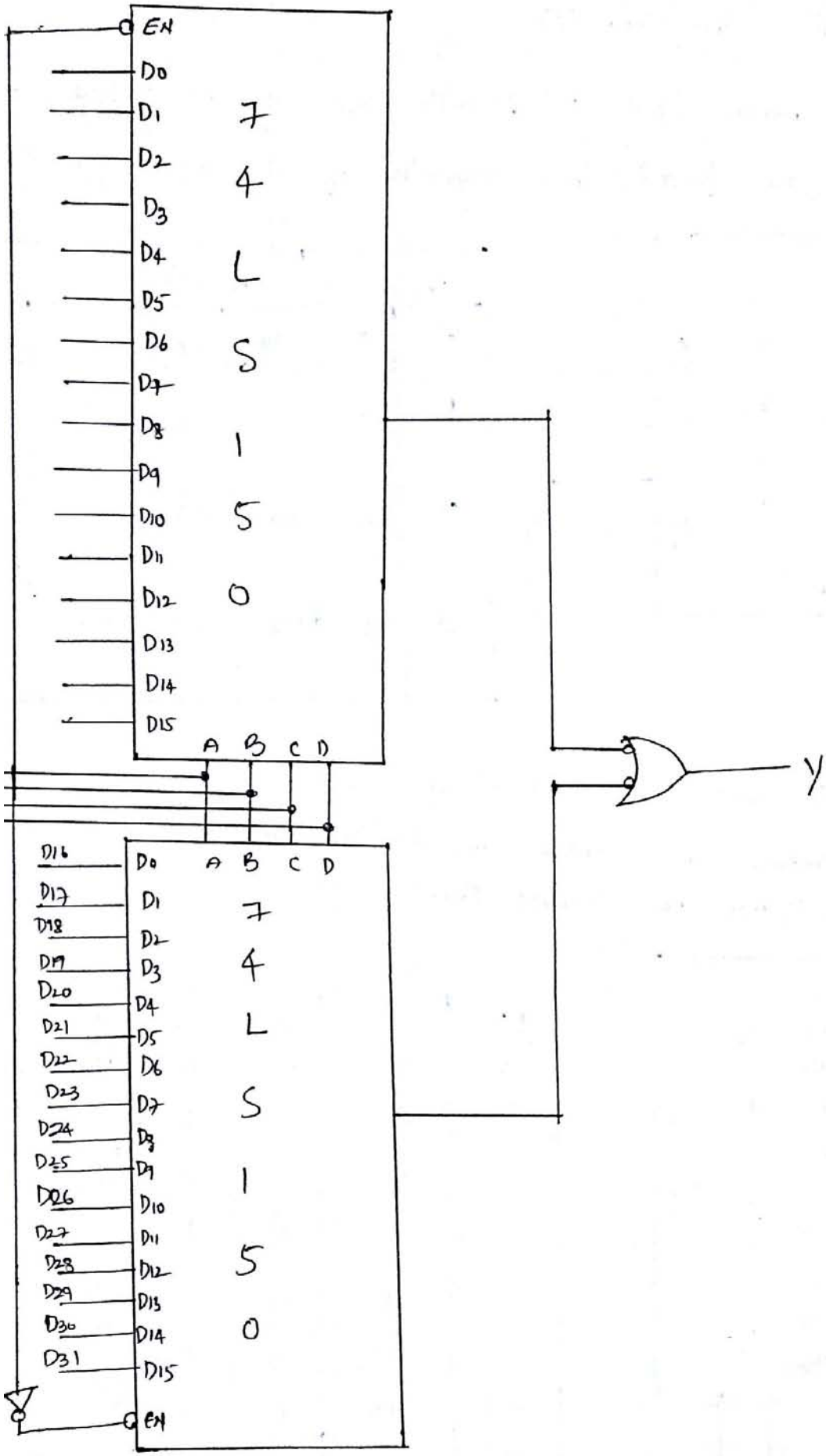
74XX153 Dual 4 to 1 multiplexer:—

— It contains two identical and identical independent 4 to 1 multiplexers. Each multiplexer has separate Enable input.



Inputs				Outputs	
1EN	2EN	B	A	1y	2y
0	0	0	0	1D0	2D0
0	0	0	1	1D1	2D1
0	0	1	0	1D2	2D2
0	0	1	1	1D3	2D3
0	1	0	0	1D0	0
0	1	0	1	1D1	0
0	1	1	0	1D2	0
0	1	1	1	1D3	0
1	0	0	0	0	2D0
1	0	0	1	0	2D1
1	0	1	0	0	2D2
1	0	1	1	0	2D3

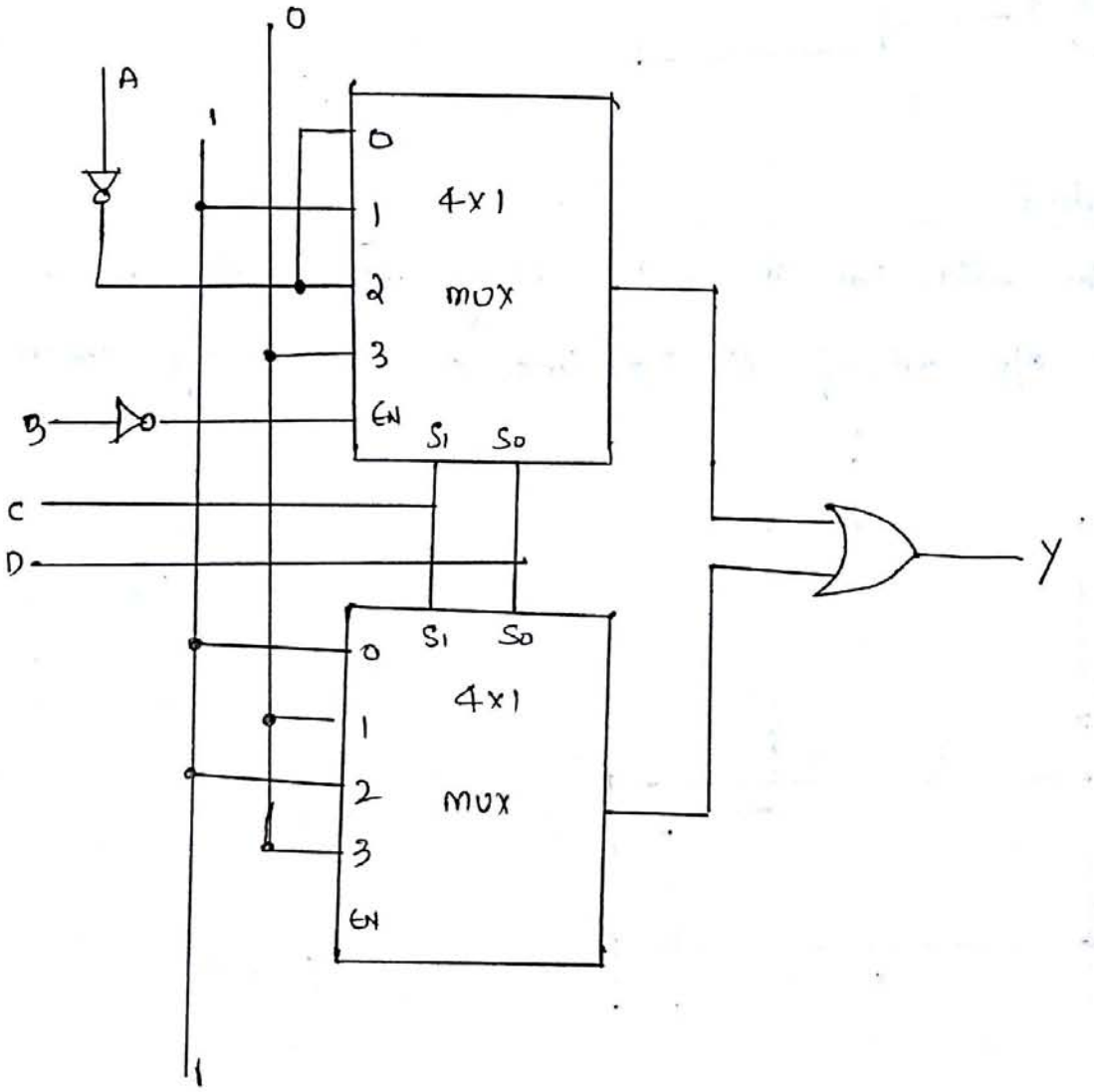
2 to 1 multiplexer using two 74LS150



$F(A, B, C, D) = \Sigma m\ (0, 1, 2, 4, 6, 9, 12, 14)$

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	(0)	(1)	(2)	3	(4)	5	(6)	7
A	8	(9)	10	11	(12)	13	(14)	15

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

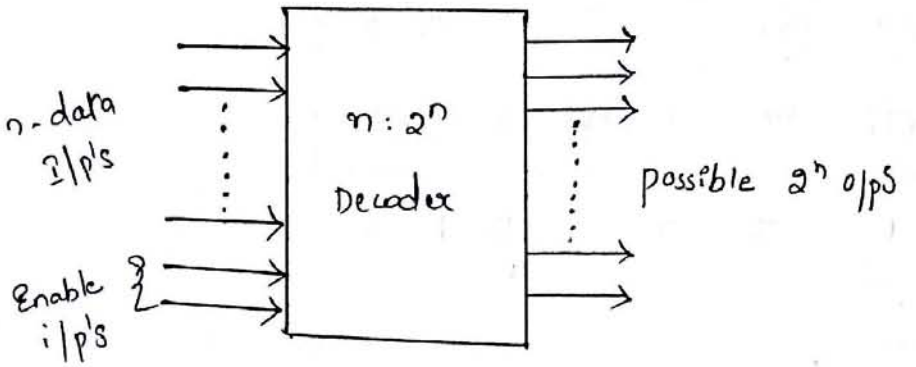


Decoder:—

— Decoder is a multiple-input, multiple o/p logic ckts which converts coded inputs into coded o/p's, where the input and o/p codes are different

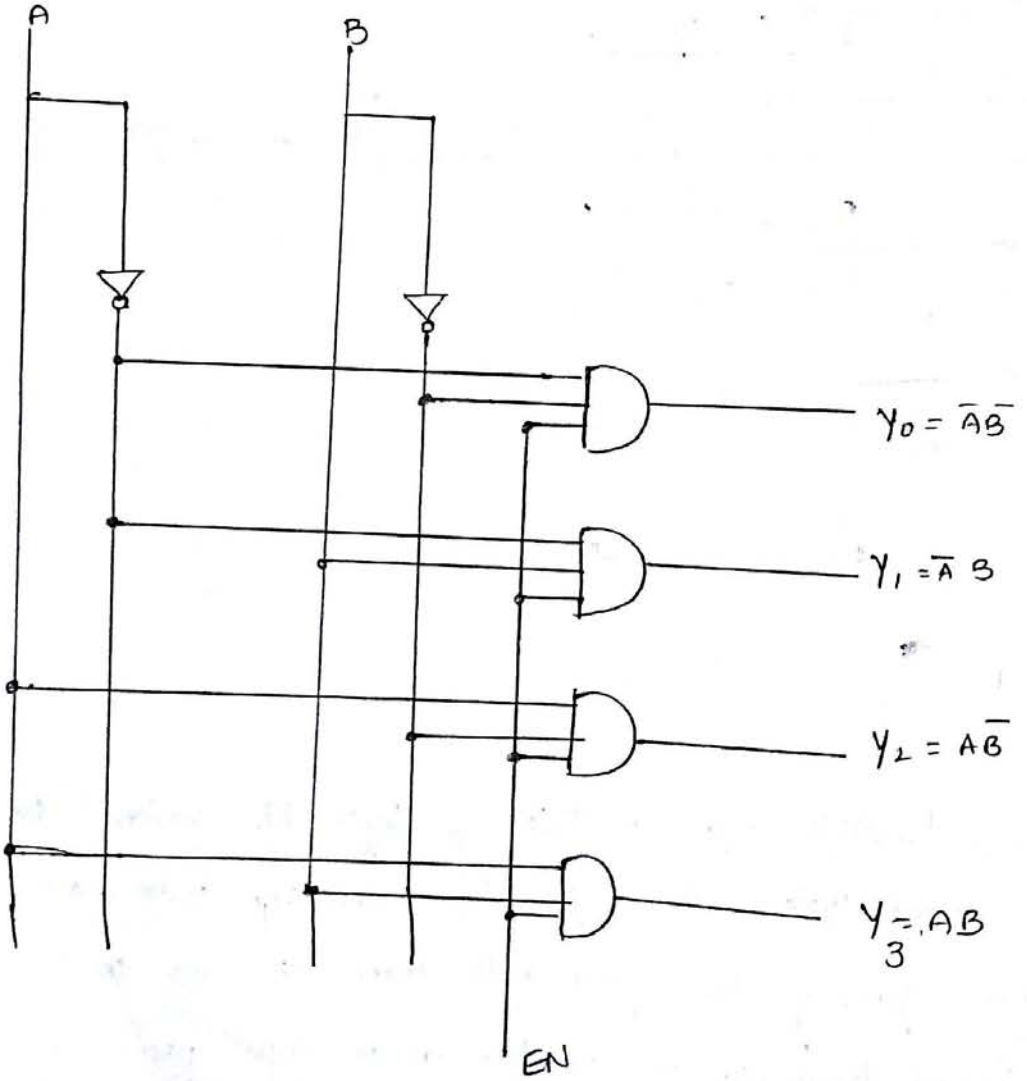
— Input code generally has fewer bits than the o/p code.

— The Encoded information is presented as n input producing 2^n possible outputs. The 2^n o/p values are from 0 through $2^n - 1$



Binary Decoder:

— A decoder which has an n-bit binary input code and a one activated o/p out of 2^n o/p code is called binary Decoder.



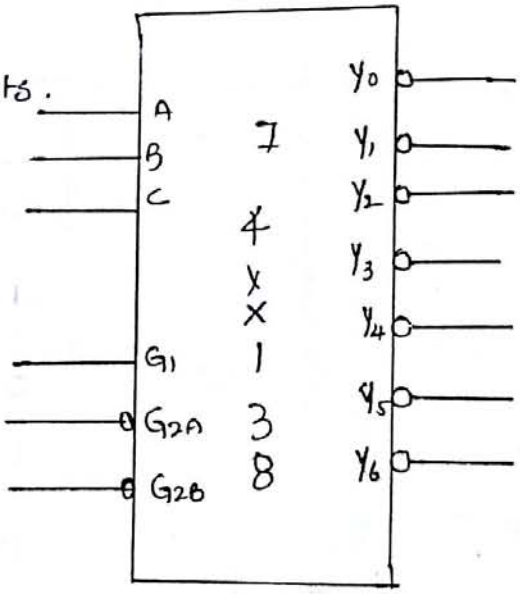
2-4, Decoder.

the two inputs are decoded into four o/p's, each o/p representing one of the minterms of the 2 input variables.

Inputs			outputs			
EN	A	B	Y_3	Y_2	Y_1	Y_0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

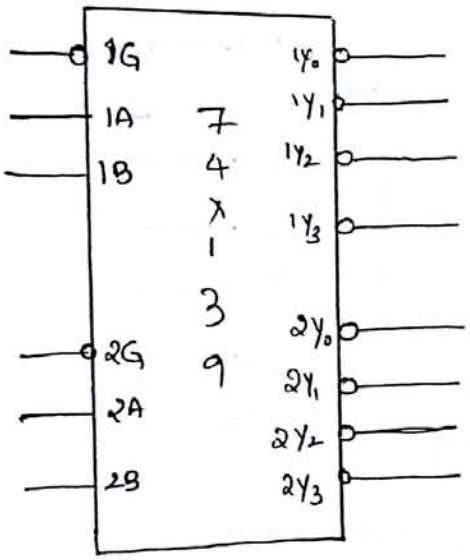
74X138 3 to 8 Decoder:—

- 74X138 is a commercially available 3 to 8 decoder.
- It accepts 3 binary inputs (C, B, A) and when Enabled provides eight individual active low o/p's (Y_0 - Y_7)
- The device has three Enable inputs. two active low ($\overline{G_{2A}}$, $\overline{G_{2B}}$) and one active high (G_1)



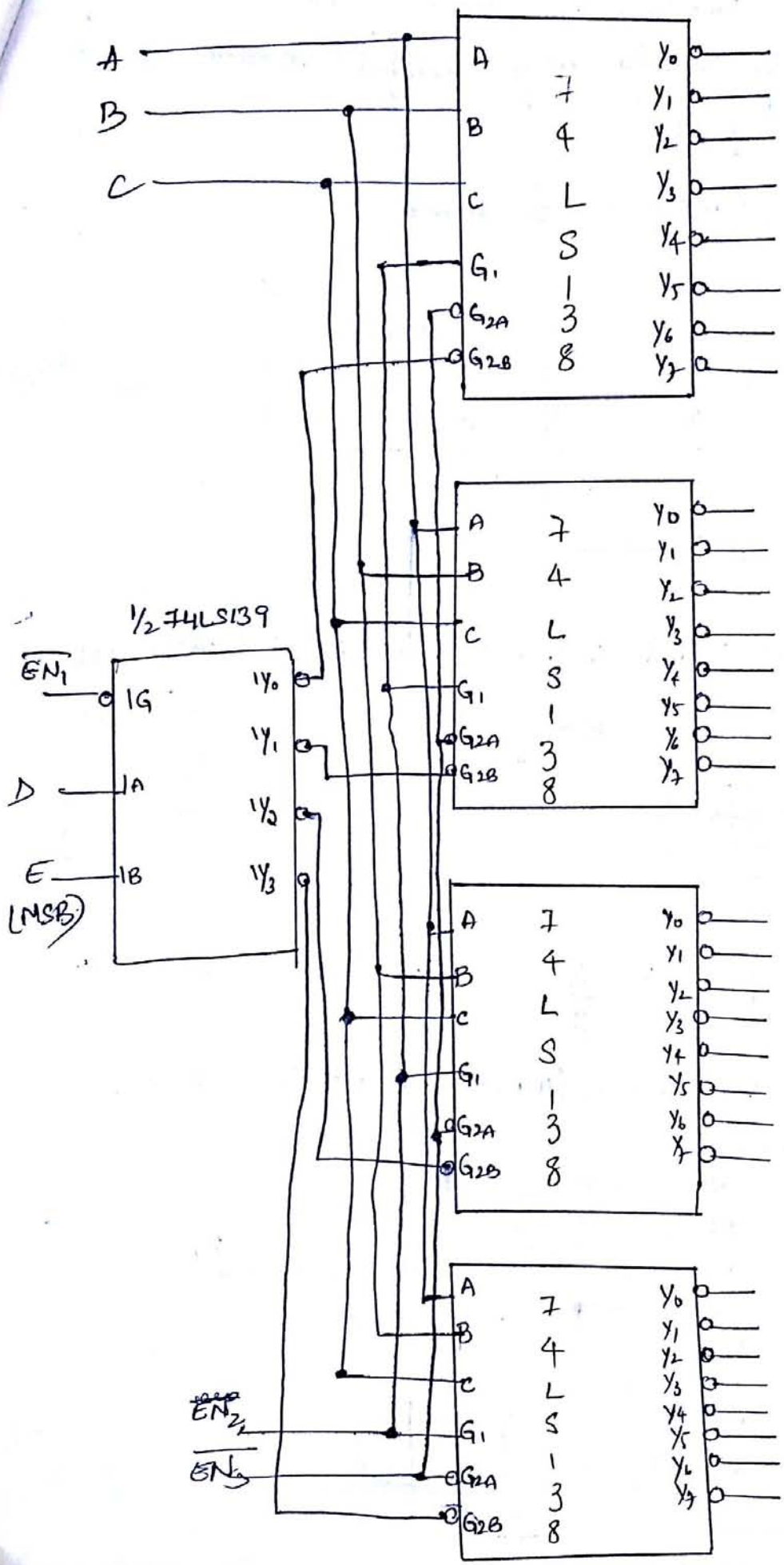
Inputs						Outputs							
G_{2B}	G_{2A}	G_1	C	B	A	$\overline{Y_7}$	$\overline{Y_6}$	$\overline{Y_5}$	$\overline{Y_4}$	$\overline{Y_3}$	$\overline{Y_2}$	$\overline{Y_1}$	$\overline{Y_0}$
1	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	0
0	0	1	0	0	1	1	1	1	1	1	1	0	1
0	0	1	0	1	0	1	1	1	1	1	0	1	1
0	0	1	0	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	0	1	1	1	0	1	1	1	1
0	0	1	1	0	1	1	1	0	1	1	1	1	1
0	0	1	1	1	0	1	0	1	1	1	1	1	1
0	0	1	1	1	1	0	1	1	1	1	1	1	1

74x139 Dual 2 to 4 Decoders –



Inputs			outputs			
$1\overline{G}$	1B	1A	$1\overline{Y_3}$	$1\overline{Y_2}$	$1\overline{Y_1}$	$1\overline{Y_0}$
1	X	X	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

Q) 5 to 32 decoder using one 2 to 4 and four 3-8 decoder IC's.



→ If order to fn

$\overline{EN_1} = 0$

$EN_2 = 1$

$\overline{EN_3} = 0$

EDCBA.

If ED = 00 → $1Y_0 \Rightarrow Y_0 - Y_7$

01 → $1Y_1 \Rightarrow Y_8 - Y_{15}$

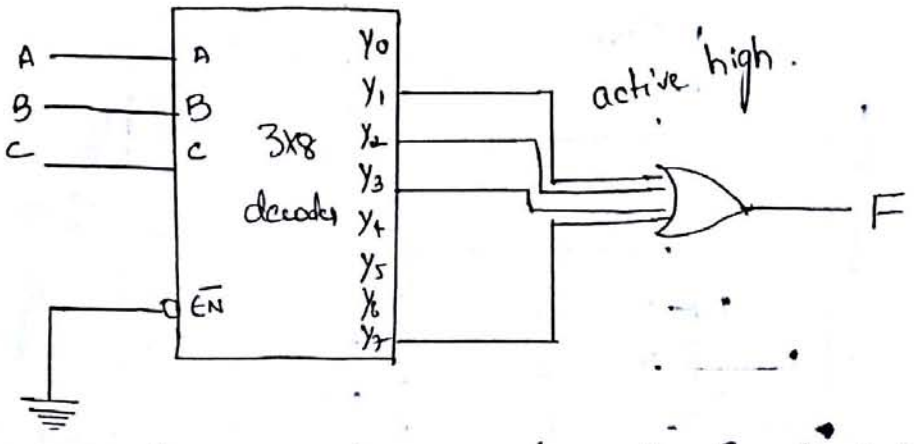
10 → $1Y_2 \Rightarrow Y_{16} - Y_{23}$

11 → $1Y_3 \Rightarrow Y_{24} - Y_{31}$

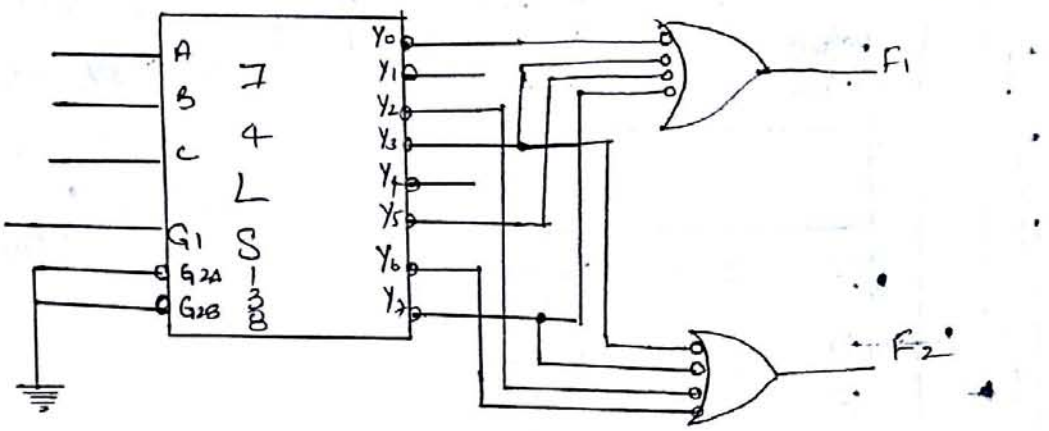
Realization of multiple o/p functions using Binary Decoder.

— The combination of decoder and External logic gates can used to implement Single or multiple o/p functions. The decoder generates minterms for input Variables. Thus by logically ORing Specified minterms we can implement the given function.

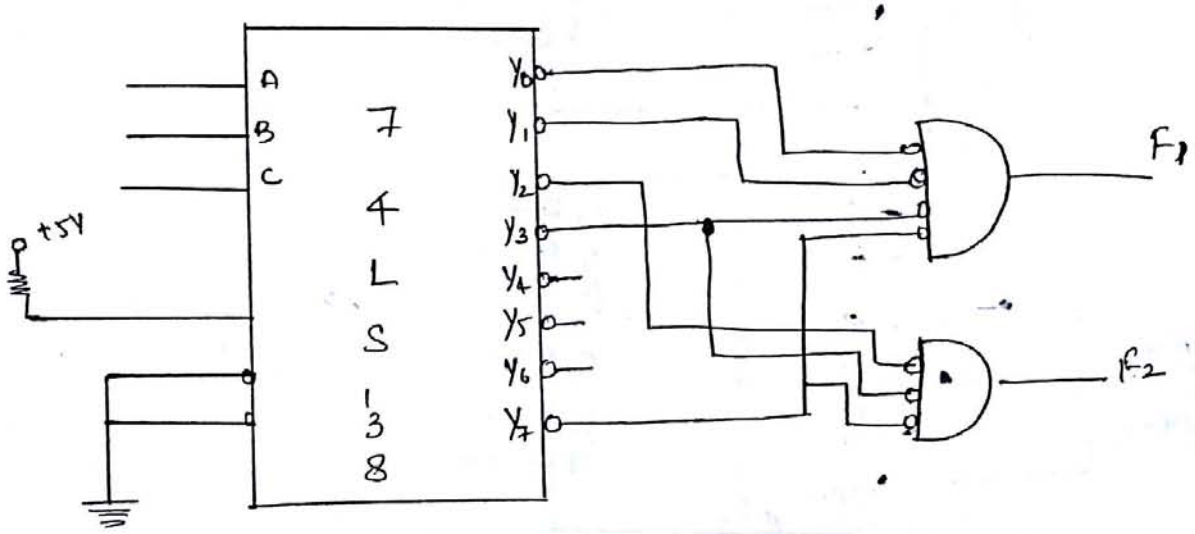
1) $f = \Sigma(1, 2, 3, 7)$ using 3 to 8 decoder.



2) $F(A,B,C) = \Sigma m(0, 3, 5, 7)$ and $f_2(A,B,C) = \Sigma m(2, 3, 6, 7)$ using 74LS138

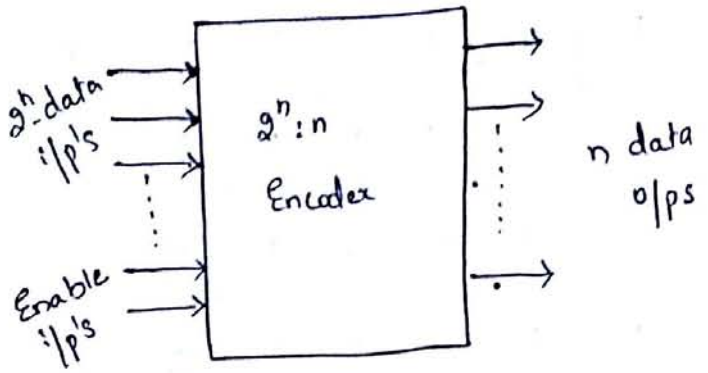


3) $f(A,B,C) = \pi(0, 1, 3, 7)$ and $f_2(A,B,C) = \pi(2, 3, 7)$



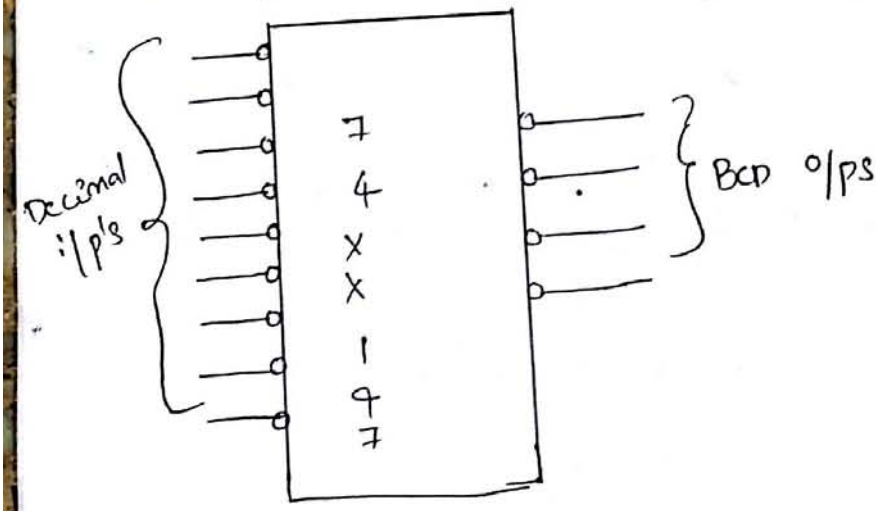
ODER :

An Encoder is a digital ckt that performs the inverse operation of decoder. An Encoder has 2^n input lines and n o/p lines. In Encoder the output lines generate the binary code corresponding to the input value.



Decimal to BCD Encoder :- (priority Encoder)

— Decimal to BCD Encoder, usually has ten input lines and four o/p lines. The decoded decimal data acts as an input for decoder and Encoded BCD o/p is available on the four o/p lines.

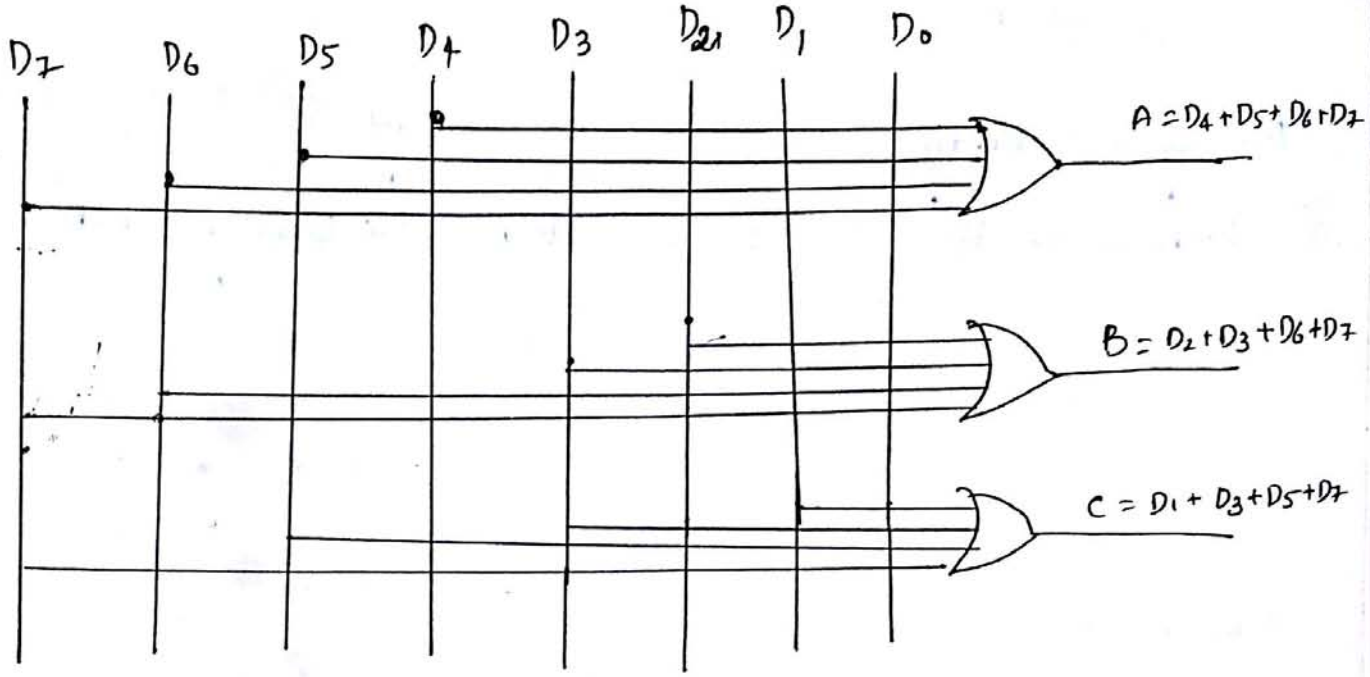


— It has nine input lines and four o/p lines. Both i/p and o/p lines are asserted low. It is important to note that there is no i/p line for decimal Zero. When this condition occurs all o/p lines are 1.

	Decimal Value	Inputs									Outputs			
		1	2	3	4	5	6	7	8	9	D	C	B	A
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1	0
2	2	X	0	1	1	1	1	1	1	1	1	1	0	1
3	3	X	X	0	1	1	1	1	1	1	1	1	0	0
4	4	X	X	X	0	1	1	1	1	1	1	0	1	1
5	5	X	X	X	X	0	1	1	1	1	1	0	1	0
6	6	X	X	X	X	X	0	1	1	1	1	0	0	1
7	7	X	X	X	X	X	X	0	1	1	1	0	0	0
8	8	X	X	X	X	X	X	X	0	1	1	0	1	1
9	9	X	X	X	X	X	X	X	X	0	0	1	1	0

octal to Binary Encoder

Inputs								outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1



List of Code Converters.

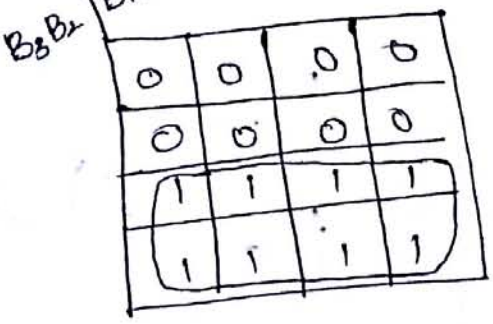
① Binary → Gray ② Gray → Binary ③ BCD → Gray

④ BCD → Ex-3 ⑤ Ex-3 → BCD ⑥ Binary – BCD :

# of a 4-bit Binary-to-Gray code converter.

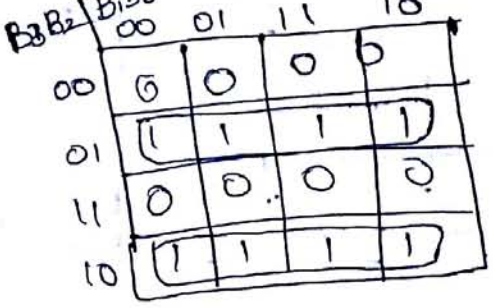
Binary				Gray			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

G_3



$G_3 = B_3$

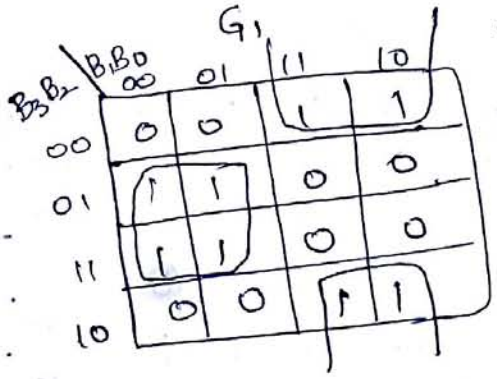
G_2



$G_2 = \bar{B}_3 B_2 + B_3 \bar{B}_2$

$B_2 \oplus B_3$

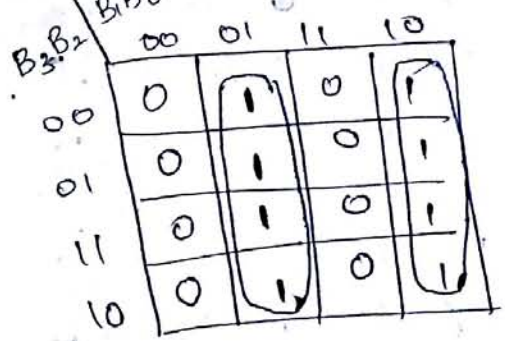
G_1

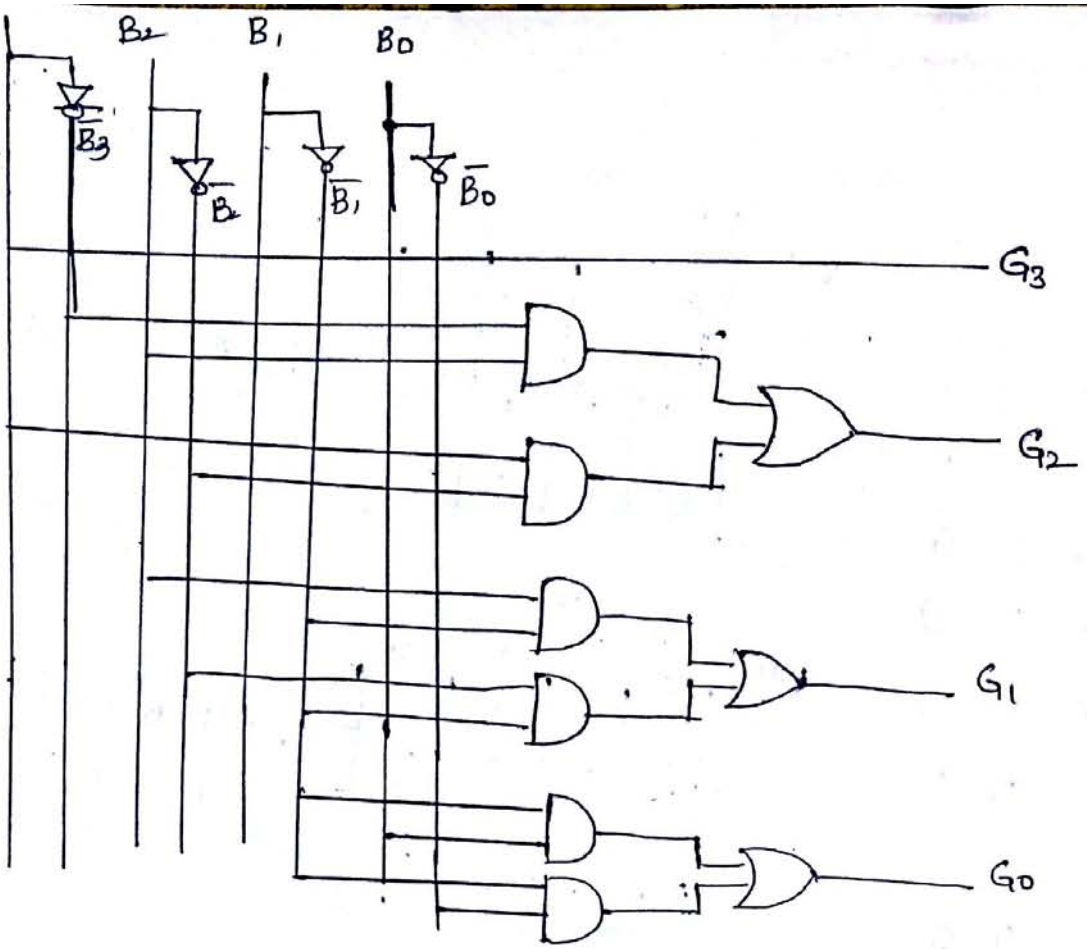


$G_1 = B_2 \bar{B}_1 + \bar{B}_2 B_1$

$B_1 \oplus B_2$

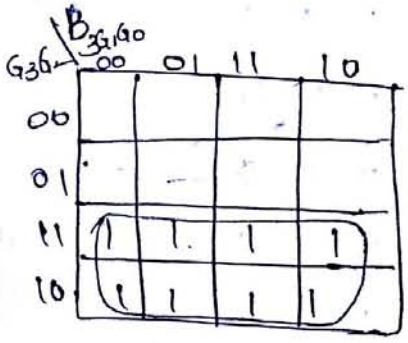
G_0



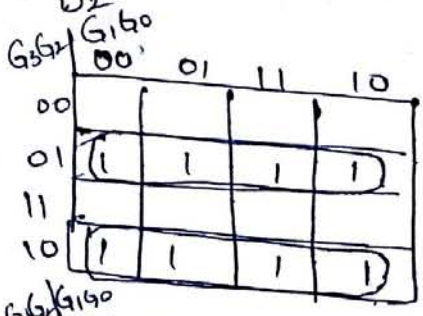


→ Design a 4-bit Gray to Binary code converter.

G_3	G_2	G_1	G_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

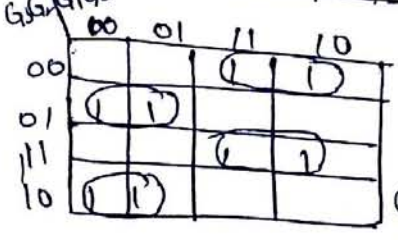


$B_3 = G_3$

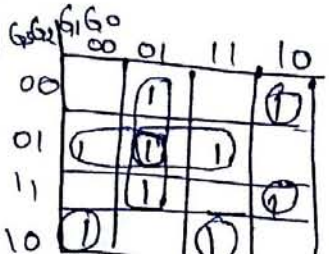


$B_2 = G_3\bar{G_2} + \bar{G_3}G_2$

$= G_2 \oplus G_3$



$B_1 = G_3 \oplus G_2 \oplus G_1$

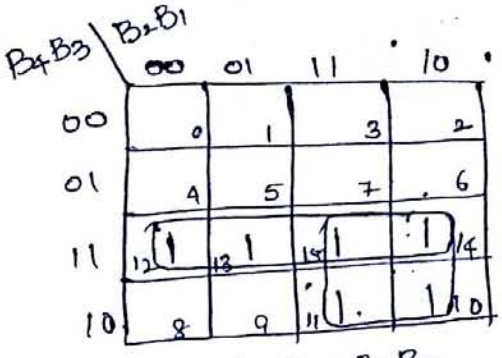


$B_0 = (G_3 \oplus G_2) \oplus (G_1 \oplus G_0)$

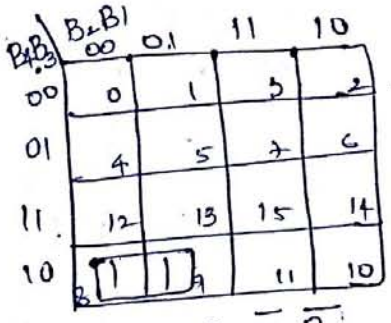
gn of a 4-bit Binary-to-BCD code converter.

Decimal	4-bit Binary $B_4\ B_3\ B_2\ B_1$	BCD output A B C D E
0	0 0 0 0	0 0 0 0 0
1	0 0 0 1	0 0 0 0 1
2	0 0 1 0	0 0 0 1 0
3	0 0 1 1	0 0 0 1 1
4	0 1 0 0	0 0 1 0 0
5	0 1 0 1	0 0 1 0 1
6	0 1 1 0	0 0 1 1 0
7	0 1 1 1	0 0 1 1 1
8	1 0 0 0	0 1 0 0 0
9	1 0 0 1	0 1 0 0 1
10	1 0 1 0	1 0 0 0 0
11	1 0 1 1	1 0 0 0 1
12	1 1 0 0	1 0 0 1 0
13	1 1 0 1	1 0 0 1 1
14	1 1 1 0	1 0 0 0 0
15	1 1 1 1	1 0 1 0 1

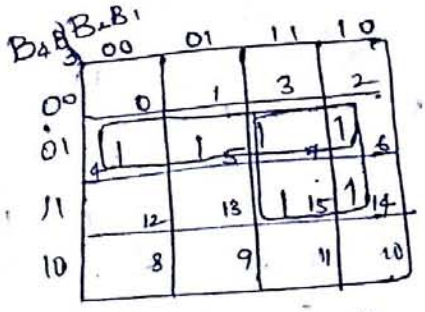
Block diagram



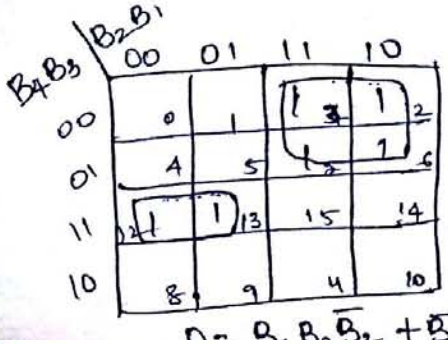
$A = B_4B_3 + B_4B_2$



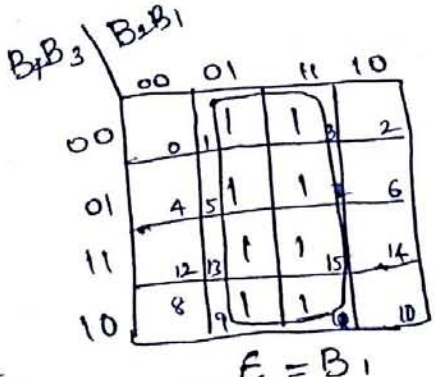
$B = B_4\bar{B}_3\bar{B}_2$



$C = \bar{B}_4B_3 + B_3B_2$



$D = B_1\bar{B}_2\bar{B}_3 + \bar{B}_4B_2$



$E = B_1$

By — P.Jayaratham and B.kalpana (Asst. Prof.)

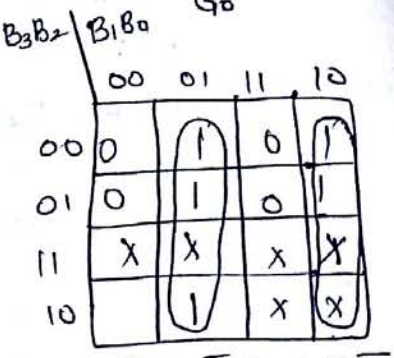
Design of a BCD-to-Gray code converters

K-map Simplification

B_3	B_2	B_1	B_0	D	C	B	A
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

dont cases.

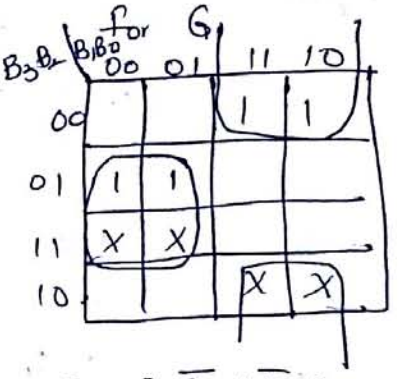
for G_0



$G_0 = \bar{B}_1 B_0 + B_1 \bar{B}_0$

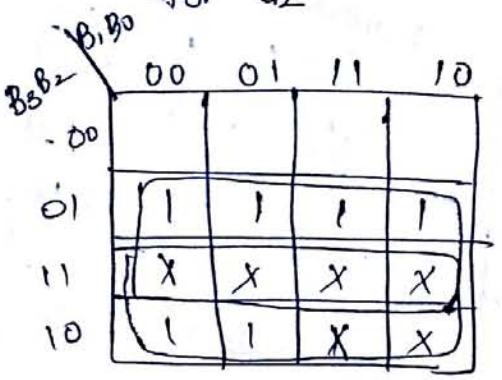
$B_1 \oplus B_0$

for G_1



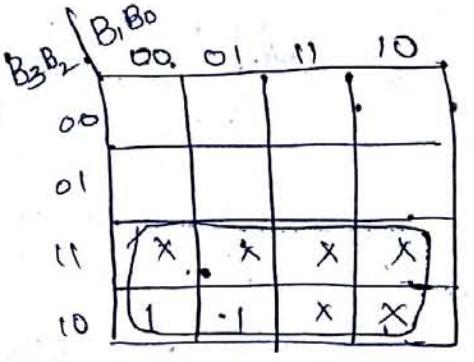
$G_1 = B_2\bar{B}_1 + \bar{B}_2 B_1$

for G_2



$G_2 = B_2 + B_3$

for G_3



$G_3 = B_3$

Design of full - Subtractor:-

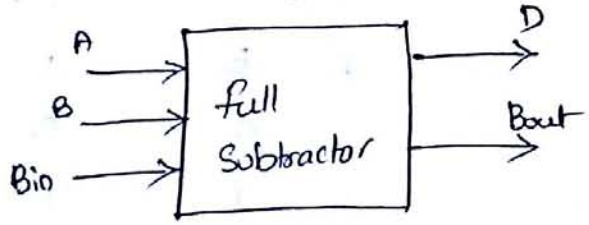
1. f.s Subbracts two bits and considering borrow from preceeding column produces difference D, Borrow (Bout)

2. A, B, Bin → i/p's

   D, Bout → o/p's

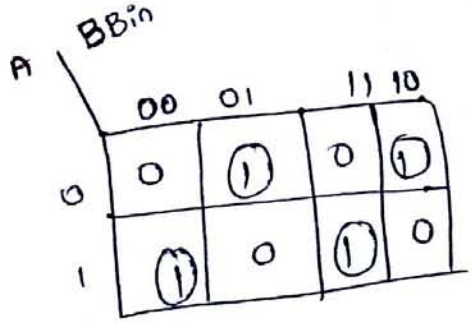
A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Block diagram



4. Simplify o/p function variables.

D

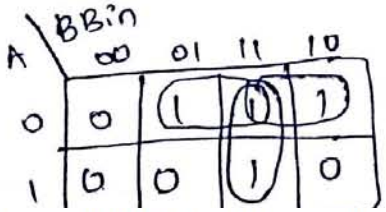


$D = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + ABB_{in}$

$= \bar{A}(B \oplus B_{in}) + A(\overline{B \oplus B_{in}})$

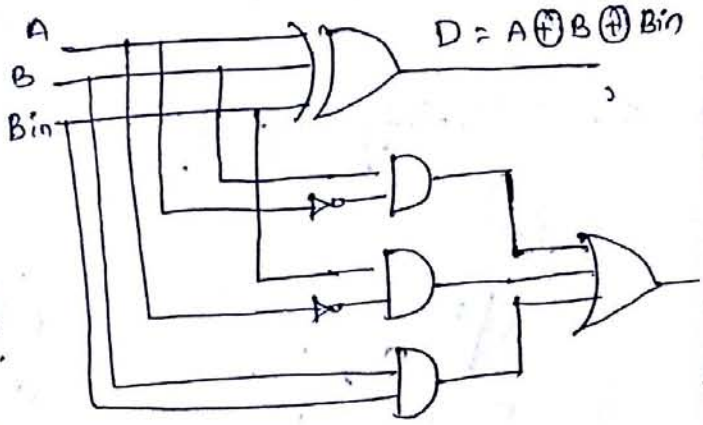
$= A \oplus B \oplus B_{in}$

Bout



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

5. logic diagram



Subtractors: Half - Subtractor → Subtracts one bit from other bit

full - Subtractor → Subtrahend is subtracted from m and considering borrow from preceding colu

Design of Half - Subtractor: (LSB Subtraction)

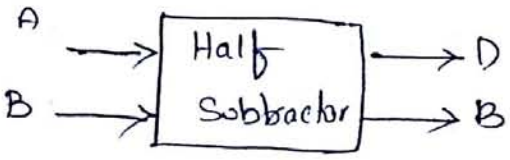
1. H.S is Combinational ckt that subtracts one bit from other produces the difference D, borrow B.

2. A, B → i/p's ; D, B → o/p's.

3. T.T

I/p's		o/p's	
A	Bin	D	Bout
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Block diagram



4. Simplify o/p function Variables.

D

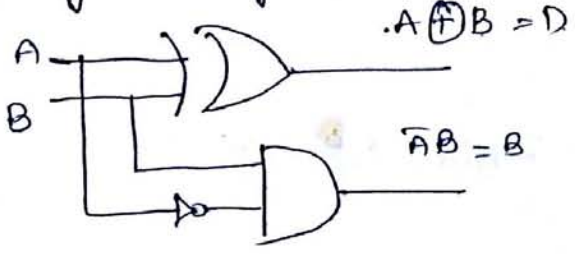
A \ B	0	1
0	0	(1)
1	(1)	0

B

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

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

5. logic diagram



Hazards and Hazard - free Realizations:-

→ Hazards are unwanted switching transients that may appear at o/p of ckt because different paths exhibit different delays. Such a transient is also called a glitch or Spurious Spike; which is caused by hazards behaviour of logic ckt.

→ A hazard in a combinational ckt is a condition where a single variable change produces a momentary o/p change when no o/p change should occur.

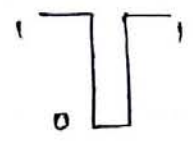
→ Hazards are of two types
(i) Static Hazard (ii) Dynamic Hazard

↳ a) static 1- hazard
b) static 0- hazard.

Static - 1 hazard:

Suppose all the i/p's are assigned some level and only i/p say x changes from 0 to 1 or 1 to 0. If o/p is Expected to be at 1 regardless of changing variable, the Spurious 0 level for a short interval is called static 1 hazard.

Static 0- hazard: If o/p is Expected to be at '0' regardless of changing variable, the Spurious 1 Level for a short interval is called a static 0 hazard.

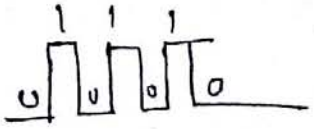


Static - 1 - hazard

static 0- hazard.

→ Static hazards can be eliminated by using redeundant gates.

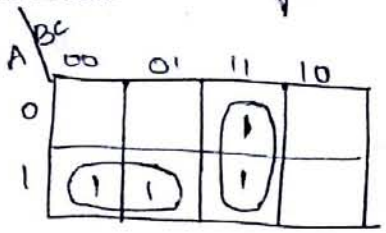
→ When o/p changes three or more times when it should change from 1 to 0 or 0 to 1 only once, it is called the dynamic hazard.



* → When a ckt is implemented in SOP with AND-OR gates or with NAND gates, Removal of Static 1(0) hazard generates that no static 0(1) hazards or dynamic hazards will occur.

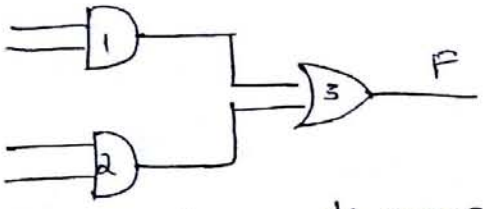
Static hazards :—

Consider the fn $F(A,B,C) = \sum m(3,4,5,7)$

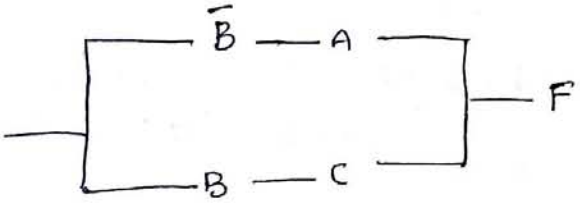


a) K-map.

$f = A\bar{B} + BC$



b) Logic diagram



c) Contact networks.

* let A=1, B=1, C=1, and only B is changing from 1 to 0. The o/p f has to remain at logic 1

(i) When B=1, o/p of gate 2 is 1, o/p of gate 1 is 0 and o/p F=1.

(ii) When B changes to 0, o/p of gate 2 is 0, o/p of gate 1 is 1 and o/p f remains at 1.

(iii) for the change in B from 1 to 0, if gate 1 responds faster than G2, f will be 1 as expected.

let $G_1 = 10ns$, $G_2 = 20ns$ $\quad B=0,\ \bar{B}=1,\ A=1 \Rightarrow G_1=1 \rightarrow F=1$

if gate G2 is faster than G1, the o/p becomes 0 before the o/p of G1 changes to 1, and for a very short time the o/p of G3 and G4 will be '0' resulting in an o/p of 0.

Let G1 = done, G2 = runs, A = 0 G2 = 0, C = 0 all 0 = runs

after some G1 = 1, G1 = 0 for 1

A little later of course the o/p gate to 1. This erratic behaviour is known as static 1 hazard

static 1 hazard.

with contact who it is called the 1st hazard.

If we consider the pos evaluation of error fn, then

$F = \pi M(0,1,2,6)$

$F = (A+B)(B+C)$

Let A = 0, B = 0, C = 0 and only B is changing from 0 to 1.

The o/p F has the remain at '0'

(ii) when B = 1, G1 = 0, G2 = 1 and F = 0 when B changes to 1,

o/p of G1 = 1, G2 = 0 and F remains at 0.

this the change in B from 0 to 1. If G2 responds faster than G1

F will be 0 as expected.

If G1 is faster than G2, the o/p becomes 1 before the o/p of G2 changes to 0 and for very short time o/p's of both

G_1 and G_2 will be 1 resulting in an o/p of 1.

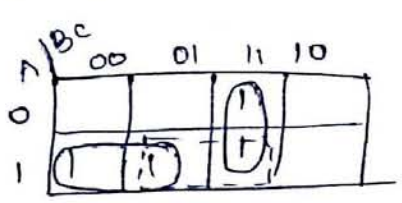
This Erratic behaviour is known as static-0 hazard. with free contact networks it is called cut set hazard.

Hazard free - Realization :—

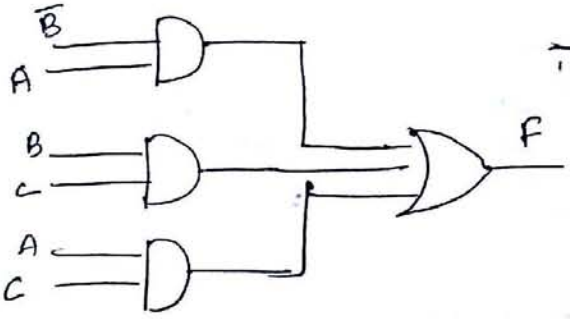
→ static -1 hazards arises because two adjacent 1's are covered by different subcubes.

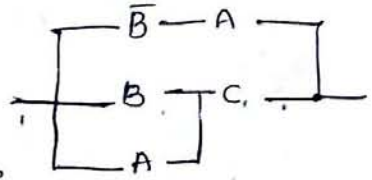
If we want to Ensure that this pair of adjacent 1's is covered by same subcube shown marked on map, we need to add one more AND gate.

The Corresponding contact n/w will have one more path. This realization will now have no static -1 hazards but the fn contains a redundant term BC



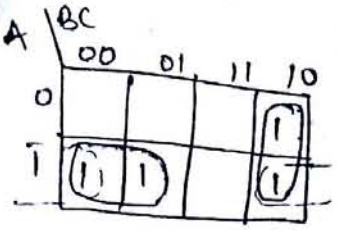
$f = A\bar{B} + BC + AC$





→ The removal of static 1 hazards in a fn by addition of subcubes does not guarantee the removal of static 0 hazards in fn.

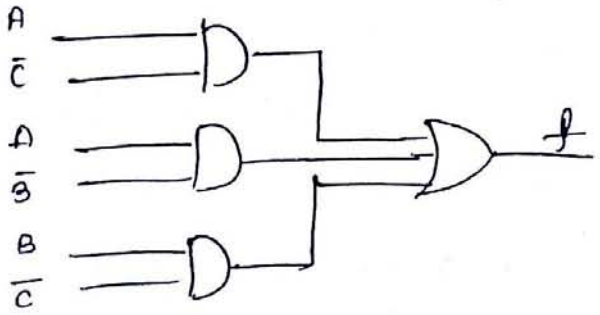
Realize the Switching fn $F(A,B,C) = \Sigma m\ (2,4,5,6)$ by a hazard free logic gate network.



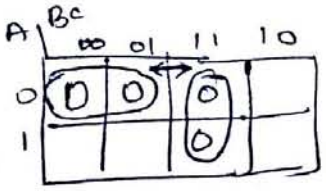
$f = A\bar{C} + A\bar{B} + B\bar{C}$

If we consider Sop form, $F = A\bar{C} + A\bar{B} + B\bar{C}$

This Sop form is hazard-free because Every pair of adjacent 1 is covered by some Subcube, and in order to satisfy this constraint, a Redaundant subc $A\bar{C}$ has tobe added.



→ Suppose we wish to Realize the same fn in Pos



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

There will be a hazard marked by arrow if we Realize it as $f = (A+B)(B+\bar{C})$

→ By Expanding pos form & ignoring term $B \cdot \bar{B} = 0$ results in Same Sop form which is hazard free.

The hazard in Pos form must be attributed to ignoring terms like $B \cdot \bar{B} = 0$

→ we can conclude that hazard free fn has to be Realized in its original form without Resorting to Simplification or factoring.

Essential Hazards:

- Essential hazard that occur in asynchronous Sequential ckts.
- It is caused due to unequal delays, along two or more paths that originate from same input.
- Excessive delay through an inverter ckt in comparision to delay associated with feedback path may cause such a hazard.
- Essential hazards cannot be corrected by adding redundant gates as in static hazard.

  This can be corrected by adjusting the amount of delay in effected path.

→ To avoid essential hazards, each feedback loop must be handled with individual care to ensure that the delay in feedback loop is long enough compared to delays of other signals that originate from input terminals.

Binary Subtractor (parallel subtractor)