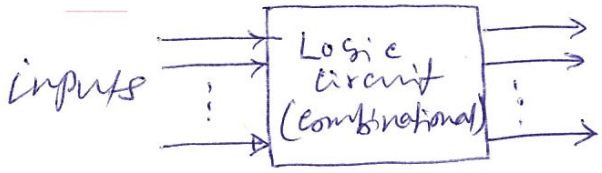
II Combinational Logic Circuits

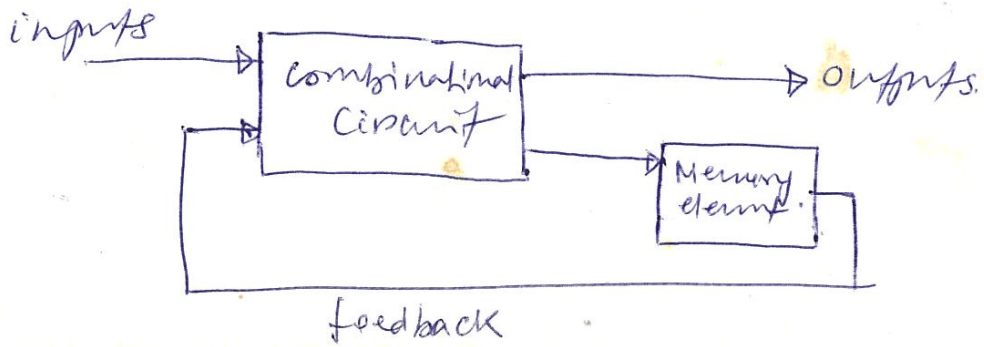
Combinational logic ⇒ A combinational circuit consists of logic gates whose outputs at any time are determined from the present combination of inputs.

A combinational logic circuit is defined as a circuit whose output(s) at any instant of time is a function of the combination of the inputs at that instant of time only without regards to the previous inputs.



Sequential logic circuits :

A sequential logic circuit is defined as a circuit whose output depends not only on the values of inputs at that instant of time, but also on past inputs.



Combinational Circuits (Standard IC)

* There are several combinational circuits ⇒
  * that are employed extensively in the design of digital systems.
  * These are available as IC and classified as Standard Digital Elements.

* Such IC's are ⇒ (Medium Scale Integration).
  * Adders
  * Subtractors
  * Comparators
  * Decoders
  * Encoders
  * Multiplexers
  * De-multiplexers etc.

Analysis of Digital Logic Circuits:

~~Conversion of a given DIGITAL LOGIC CIRCUIT Development of a~~

The analysis of a combinational circuit means ⇒

* Determination of the function that a given Logic circuit implements.

Procedure ⇒

* start with the LOGIC DIAGRAM (Circuit)
  ↓ Finally arrive at
* Boolean function corresponding to the given Logic circuit.

Design of a Logic Circuit:

The design of a combinational circuit means ⇒

* The determination of a LOGIC CIRCUIT DIAGRAM

* From the specification of the problem

Procedure:

1. From the specifications of the problem ⇒
   * determine the no. of inputs and outputs required ⇒
   * And assign symbols to each (such as, x, y, z, F_1, F_2 -- etc).
2. Derive the corresponding truth table of the problem in terms of inputs (x, y, z) and outputs (F_1, F_2, --)
3. Derive the simplified Boolean functions for each outputs (F_1, F_2, --) as a function of inputs.
4. Draw the logic circuit diagram
5. Verify the correctness of the design.

DESIGN

① BCD to Excess-3 Code Converter Logic Circuit :-

Solu: Solution:

Step 1.

BCD : 4-bit number (0 – 9), assign A, B, C, D

Excess-3 code : 4-bit number [BCD + 3]., assign: W, x, y, z

Step-2

Truth Table:

Inputs 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

Step-3 Simplified Boolean Functions :-

For output z :

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	0	1
$\bar{A}B$	1	0	0	1
AB	X	X	X	X
$A\bar{B}$	1	0	X	X

$z = \bar{D}$

For output y :

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	1	0
$\bar{A}B$	1	0	1	0
AB	X	X	X	X
$A\bar{B}$	1	0	X	X

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

$= \overline{C \oplus D}$

For output x :

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	1	1
$\bar{A}B$	1	0	0	0
AB	X	X	X	X
$A\bar{B}$	0	1	X	X

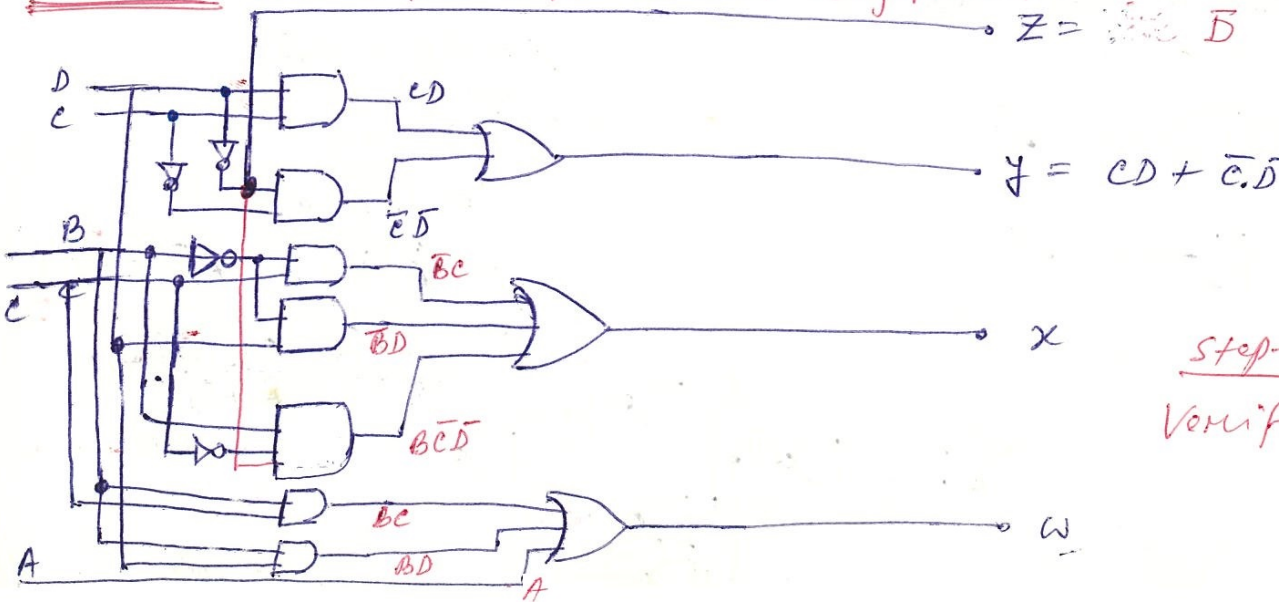
$x = \bar{B}C + \bar{B}D + B\bar{C}\bar{D}$

For output w :

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	0	0
$\bar{A}B$	0	1	1	1
AB	X	X	X	X
$A\bar{B}$	1	1	X	X

$w = A + BC + BD$

Step-4. Draw the logic circuit diagram :-



Step-5. Verification

HALF ADDER

* Addition of two one bit numbers.
* Two outputs ⇒ Sum, Carry
* It is a simple ⇒
  * multiple input
  * multiple output
  } combinational circuit.

Truth table :

Inputs		outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

S = ?

A \ B	$\bar{B}$	B
$\bar{A}$	0	(1)
A	(1)	0

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

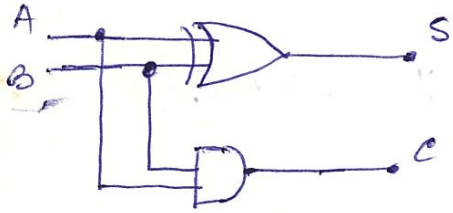
C = ?

A \ B	$\bar{B}$	B
$\bar{A}$	0	0
A	0	(1)

$$C = AB$$

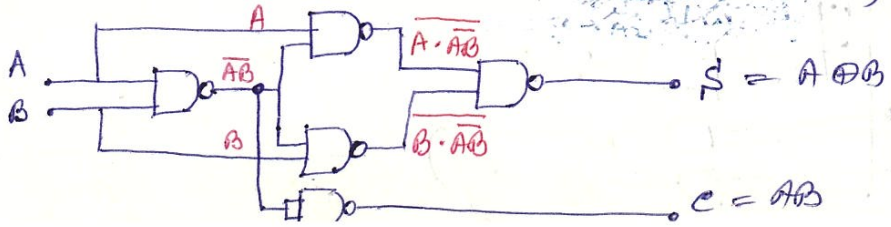
Circuits :

① Circuit with XOR and AND.

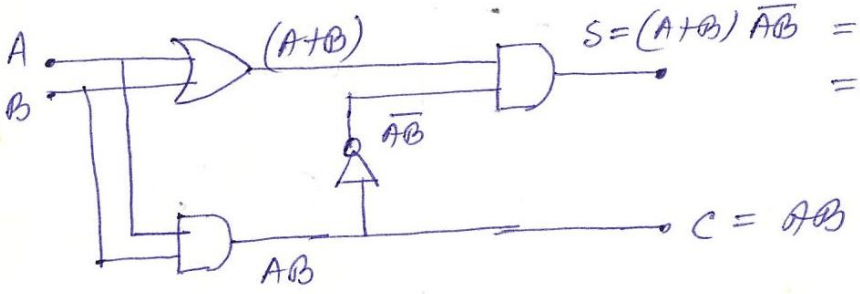


② Circuit with NAND gate :-

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

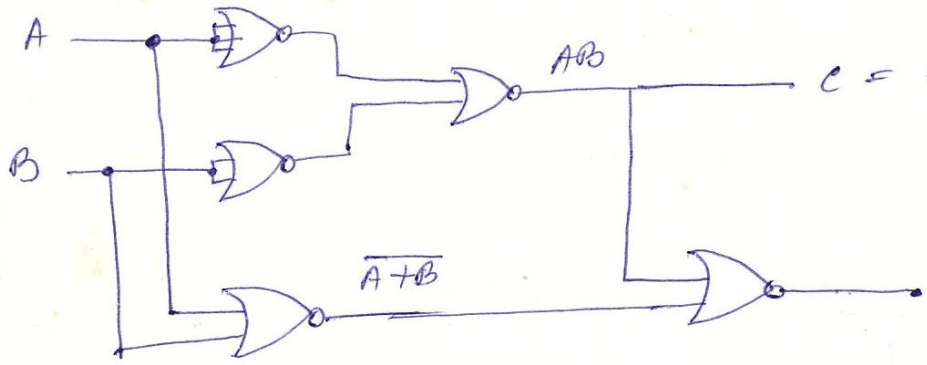


③ Circuit using AND, OR, and NOT :-



$S = (A+B)\,\overline{AB} = (A+B)(\bar{A}+\bar{B})$
$= A\bar{B} + \bar{A}B = A \oplus B$

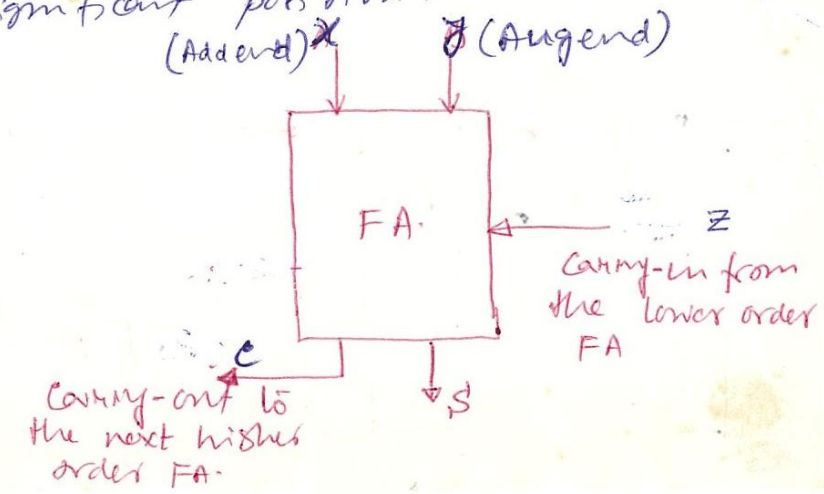
④ Circuit using NOR gates :-

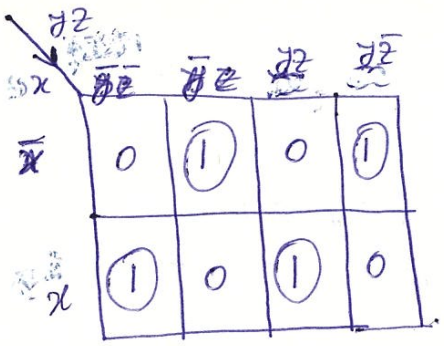


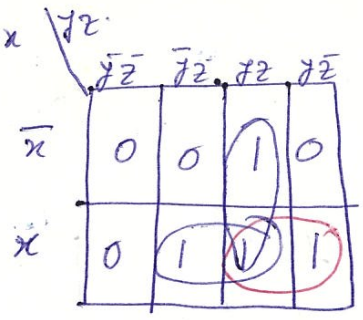
FULL ADDER :

* It performs the addition of three bits. ⇒
  * Two significant bits.
  * one previous carry from the lower significant position.

X	Y	Z	S	C
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







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

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

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

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

$$= x \oplus y \oplus z \quad \text{——— (1)}$$

$$C = yz + \bar{x}z + \bar{x}y$$

$$= xy + yz + zx \quad \text{——— (2)}$$

① FA using XOR, AND & OR Gates :-

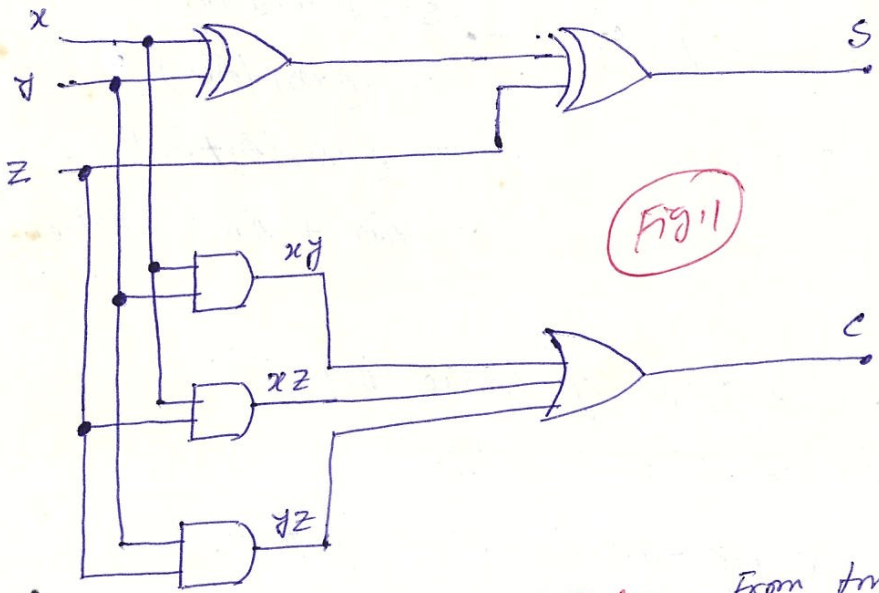


Fig:1

② FA using XOR and NAND :-

From truth table we get SOP expression for C as $C = \bar{x}yz + x\bar{y}z + xy\bar{z} + xyz$

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

$$= (x \oplus y)z + xy$$

$$= \overline{\overline{(x \oplus y)z + xy}}$$

$$= \overline{(x \oplus y)z} \cdot \overline{xy} \quad \text{——— (3)}$$

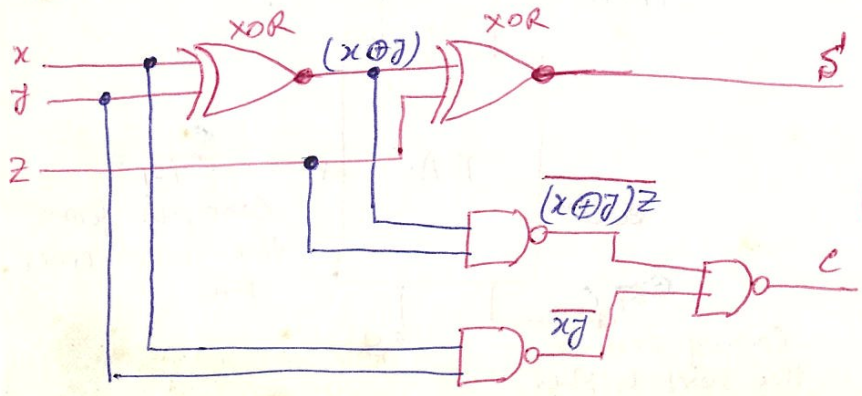


Fig:2

(3) FA using only NAND gates :-

Using eqns. (1) and (3) we may implement the FA circuit using all NAND gate as follows:-

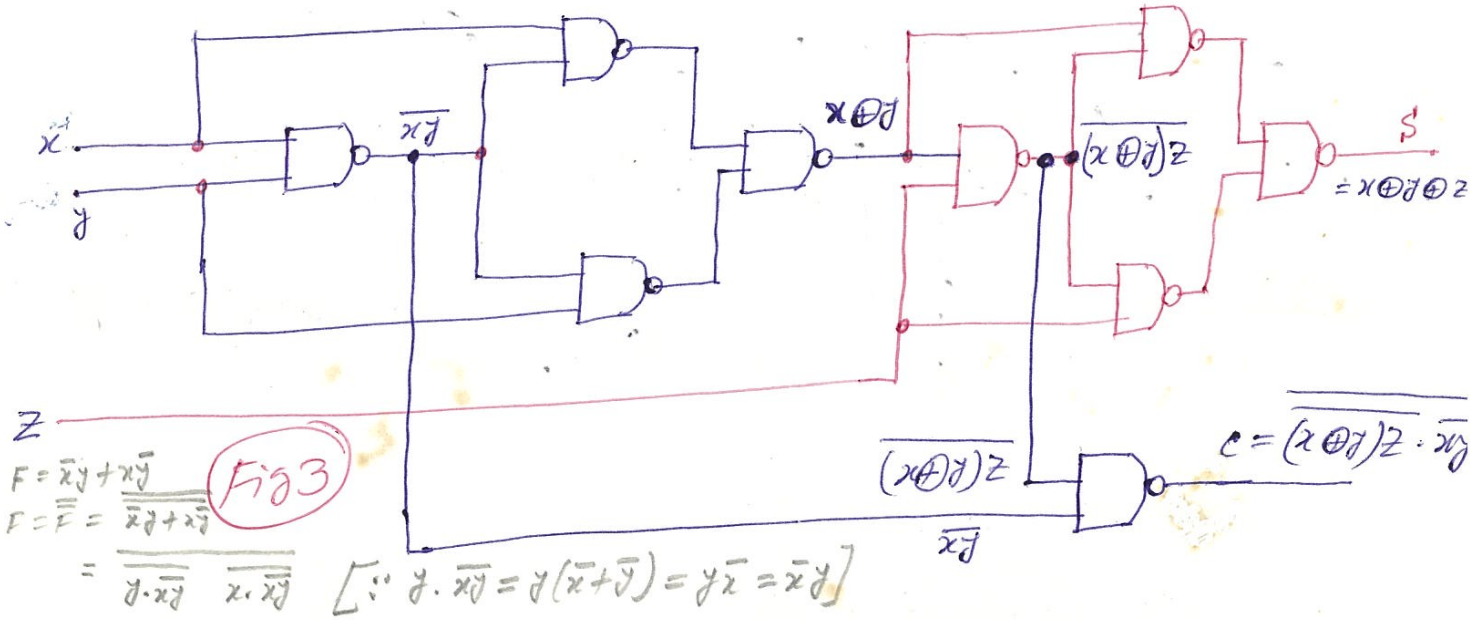


Fig 3

$F = \bar{x}y + x\bar{y}$

$F = \bar{\bar{F}} = \overline{\bar{x}y + x\bar{y}}$

$= \overline{y \cdot \overline{xy}} \;\; \overline{x \cdot \overline{xy}}$ $\left[\because y \cdot \overline{xy} = y(\bar{x} + \bar{y}) = y\bar{x} = \bar{x}y\right]$

(4) FA using all NOR Gates :-

From (3) of the truth table we get,

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

$$= (\bar{x}\bar{y} + xy)\bar{z} + (\bar{x}y + x\bar{y})z = (x \odot y)\bar{z} + (x \oplus y)z.$$

$$= (x \odot y)\bar{z} + (\overline{x \odot y})z = (x \odot y) \oplus z$$

$$S = \bar{\bar{S}} = (x \odot y) \oplus z \qquad \text{(4)}$$

Similarly, $\bar{C} = \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z}$

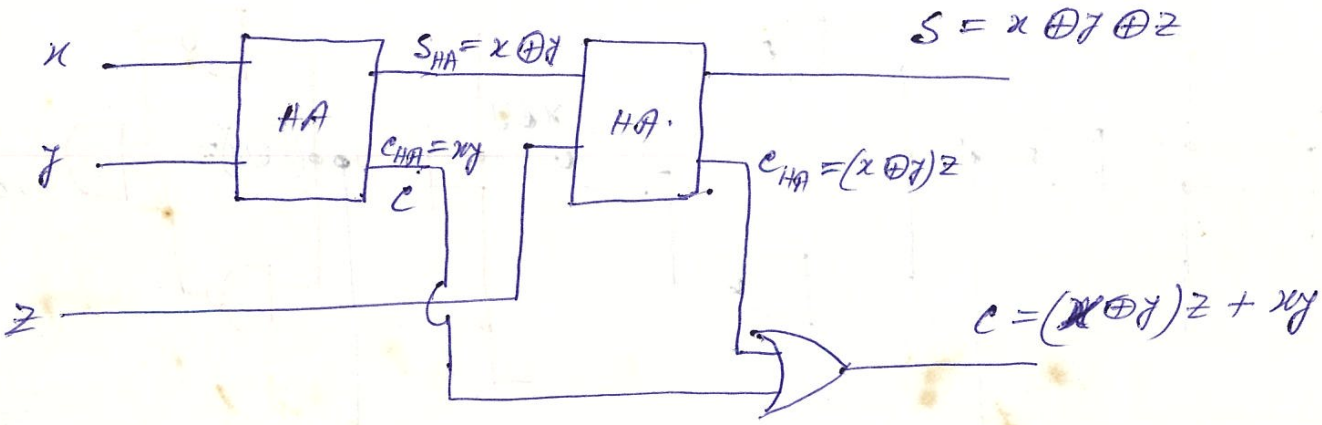
$$= (x \oplus y)\bar{z} + \bar{x}\bar{y}(z + \bar{z})$$

$$= \overline{(x \odot y)}\,\bar{z} + \bar{x}\bar{y} = \overline{x \odot y + z} + \overline{(x + y)}$$

$$C = \overline{\overline{x \odot y + z} + \overline{(x + y)}} \qquad \text{(5)}$$

NOR circuit : The one-to-one replacement of NAND gates by NOR gates of fig. 3 will give the NOR implementation of FA.

(5) Full Adder (FA) using two Half Adder (HA) :-



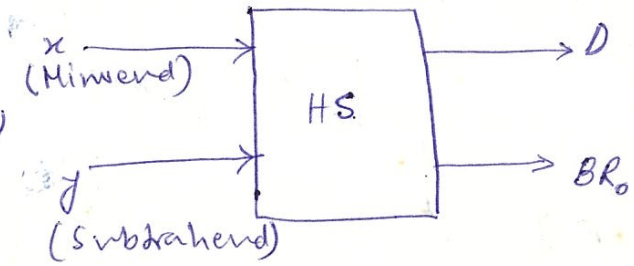
From eqn. (3) we get,

$$C = \overline{(x \oplus y)z} \cdot \overline{xy}$$

$$= (x \oplus y)z + xy$$

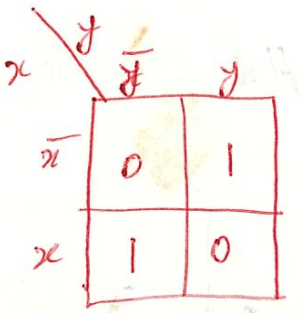
HALF SUBTRACTOR

* subtraction of two one bit numbers.
* Two outputs — Difference (D), Borrow (BR)



Truth table:

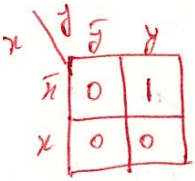
Minuend x	Subtrahend y	Difference D	Borrow out BR_0
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



$D = \bar{x}y + x\bar{y}$

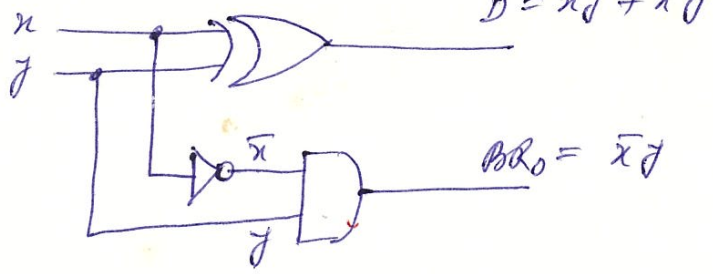
Logic expression:

$\therefore D = x\bar{y} + \bar{x}y = x \oplus y$



$BR_0 = \bar{x}y$

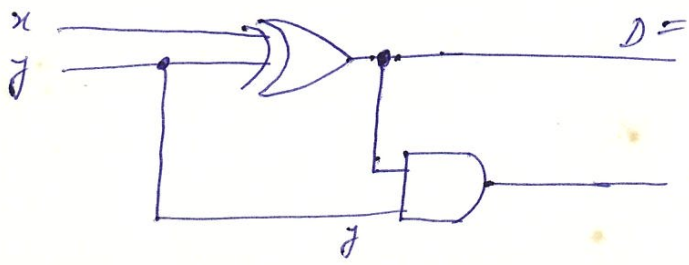
① circuit with XOR, AND, NOT gates:-



$D = x\bar{y} + \bar{x}y$

$BR_0 = \bar{x}y$

② circuit with XOR, AND gates:-

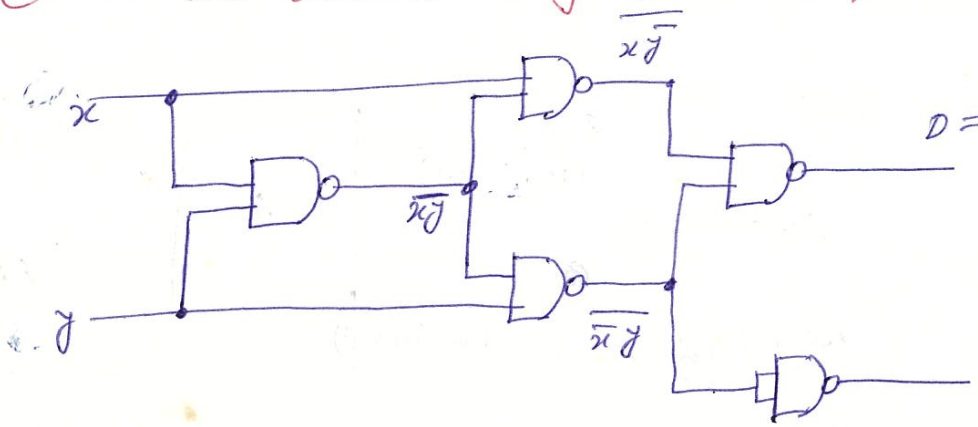


$D = \bar{x}y + x\bar{y}$

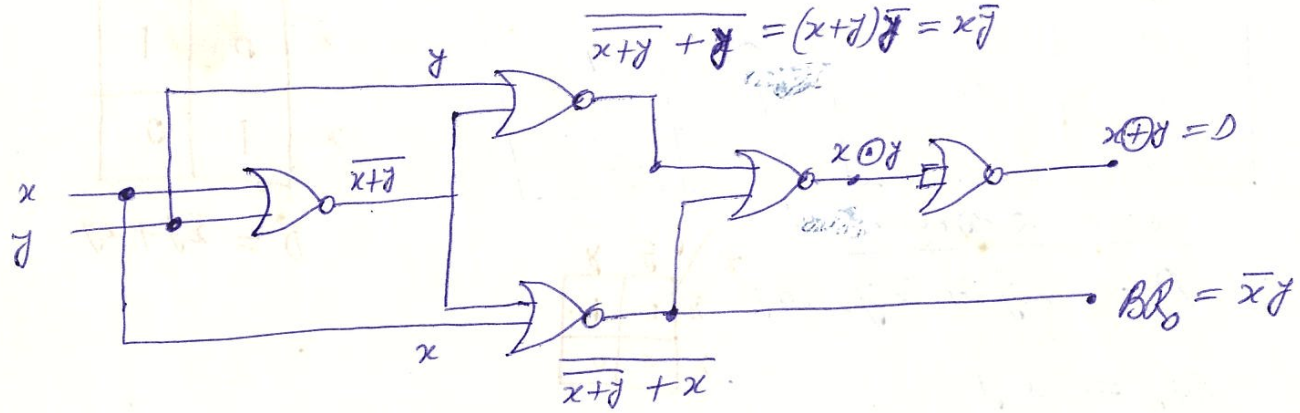
$BR_0 = (\bar{x}y + x\bar{y})y$

$= \bar{x}y$

(3) HS circuit using all NAND gates :-



(4) HS circuit using all NOR Gates :-

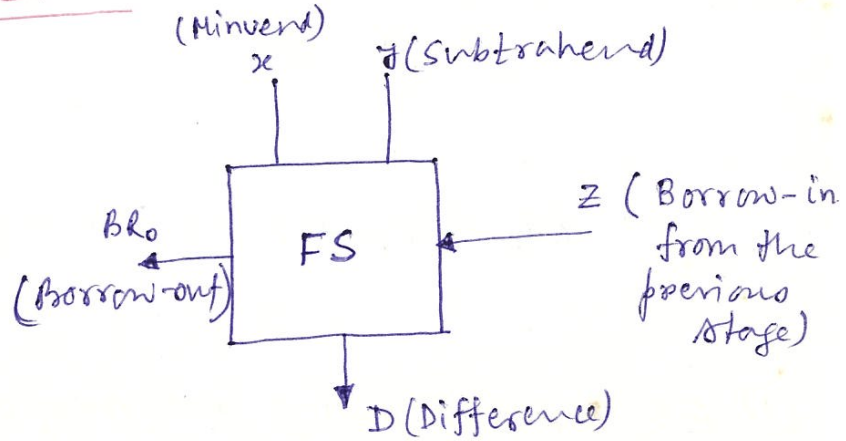


$= (x+y) \cdot \bar{x}$

$= \bar{x}y$

FULL SUBTRACTOR

	1	14	
	2	4	16
Minuend	2	5	6
subtrahend	1	5	7
	0	9	9



Minuend x	Subtrahend y	Borrow-in from the adjacent lower significant stage i.e z	Difference D	Borrow out to the next higher significant stage BR_0
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

* comparing FA truth table with FS ⟹.

* we find $S = D$

$\therefore D = x \oplus y \oplus z$ ——— (1)

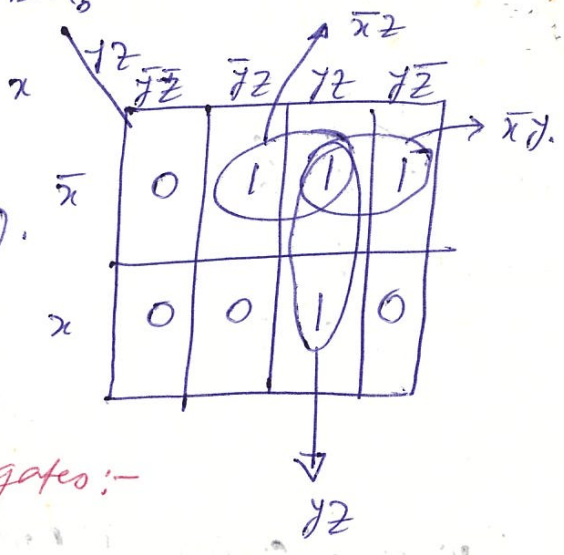
* SOP form of BR_0 as obtained from truth table ⟹

$BR_0 = \bar{x}\bar{y}z + \bar{x}y\bar{z} + \bar{x}yz + xyz$ ——— (2)

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

$= \bar{x}y + (\overline{x \oplus y})z = \overline{\bar{x}y \cdot (x \odot y)z}$ ——— (3)

Again, simplified expression for BR_0 as obtained from K-map ⇒.



$\therefore BR_0 = \bar{x}y + yz + \bar{x}z$ — (4).

(1) FS circuit using XOR, AND, OR and NOT gates :-

[Using eqns (1) & (4)]

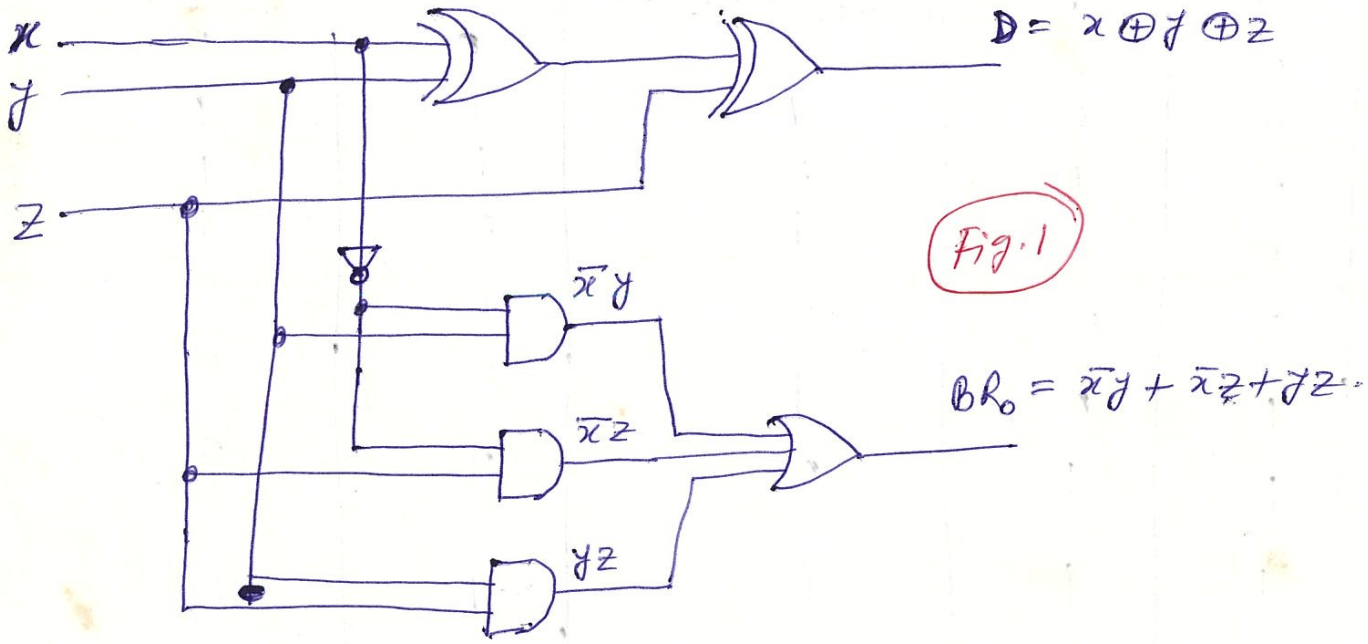


Fig. 1

(2) FS using all NAND gates :-

[Using eqns. (1) & 3]

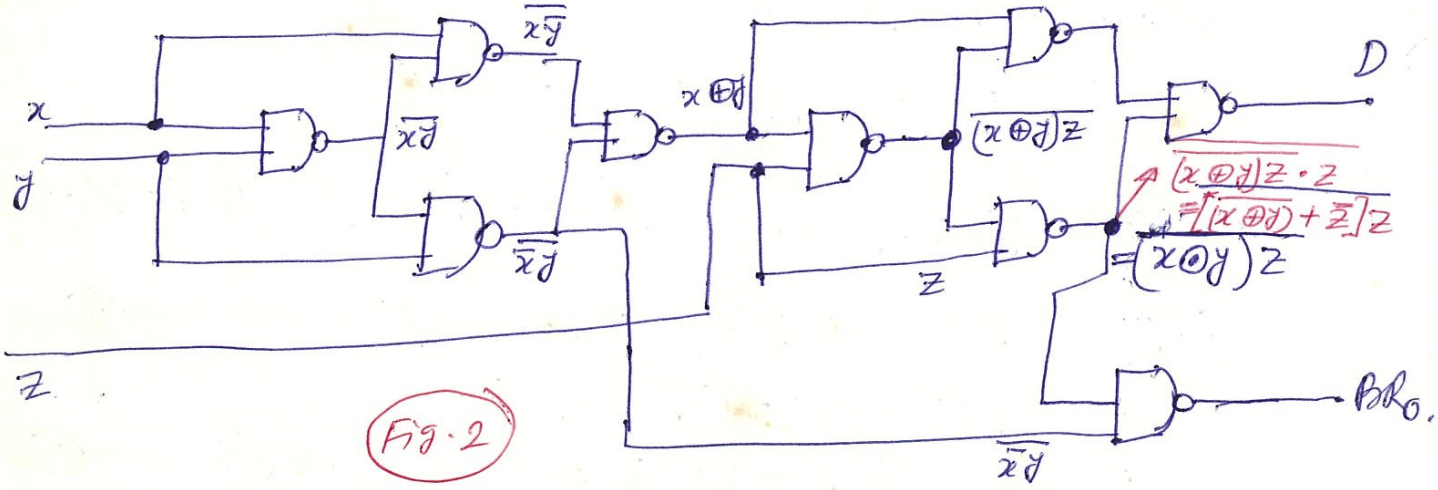


Fig. 2

③ FS Circuit using all NOR gates :-

* To design with NOR gates
* The entries 0s are to be used to get ⇒
  * Boolean expressions for "D" and "BR_0".

For D :

From truth Table we get, (by expressing standard SOP form)

$$\bar{D} = \bar{x}\bar{y}\bar{z} + \bar{x}y\bar{z} + x\bar{y}z + xy\bar{z}$$
$$= (\bar{x}\bar{y} + xy)\bar{z} + (\bar{x}y + x\bar{y})z$$
$$= (x \odot y)\bar{z} + \overline{(x \odot y)}z$$
$$= (x \odot y) \oplus z$$
$$= \overline{(x \odot y) \odot z}$$

$$\therefore \boxed{D = x \odot y \odot z}$$

For BR_0 : In standard SOP form for Zeros (0s).

$$\overline{BR_0} = \bar{x}\bar{y}\bar{z} + x\bar{y}\bar{z} + x\bar{y}z + xy\bar{z}$$
$$= (\bar{x}\bar{y} + xy)\bar{z} + x\bar{y}(\bar{z} + z)$$
$$= (x \odot y)\bar{z} + x\bar{y}$$
$$= \overline{x \odot y} + z + \bar{x} + y$$

$$\therefore BR_0 = \overline{\overline{x \oplus y} + z + (\bar{x} + y)}$$

NOR circuit : The one-to-one replacement of NAND gates by NOR gates of fig. 2 will give the NOR implementation of FS.

(4) FS using HS :-

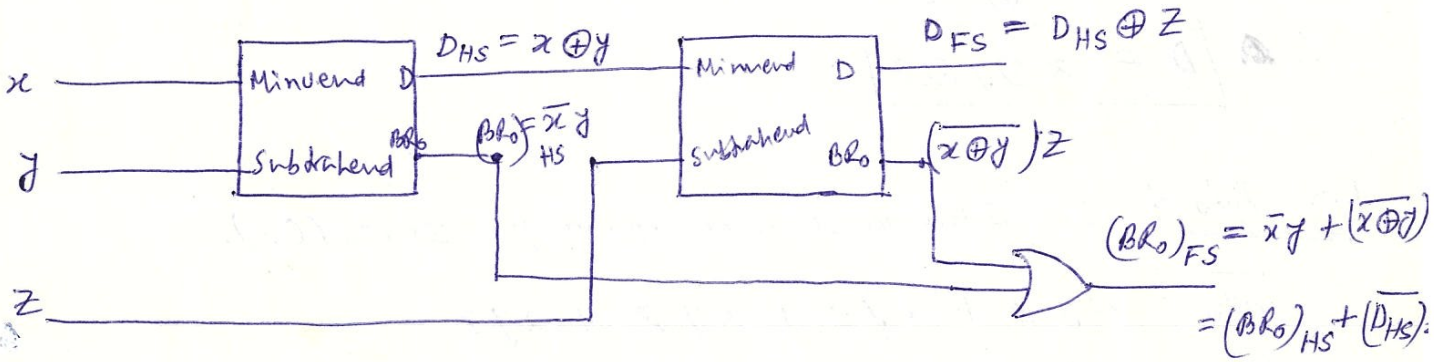
$$D_{FS} = x \oplus y \oplus z = D_{HS} \oplus z$$

~~$(BR_0)_{FS} = x \oplus (x \oplus y) z + \bar{x} y$~~

$$(BR_0)_{FS} = \bar{x} y + (\bar{x}\bar{y} + xy) z \quad [\text{From eqn. (3)}]$$

$$= (BR_0)_{HS} + (\overline{D_{HS}}) z$$

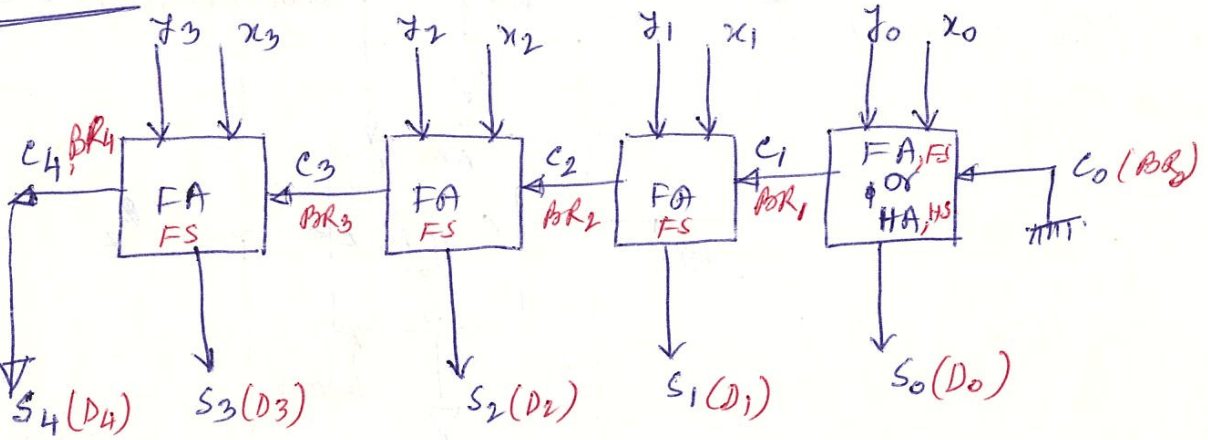
∴ circuit is ⇒.



Multi-bit Adder : (Subtracter) [Ripple carry Adder] [Ripple Borrow Subtracter]

* It is a digital circuit that produces the arithmetic sum of two binary numbers.
* It is constructed ⇒
  * with a number of 1-bit full adder connected in cascade.
  * with the carry output from each FA connected to the input of the next FA in the chain.
  * And the inputs bits (Augend (Minuend) & Addend (Subtrahend)) are connected parallelly.

4-bit Adder:



* An n-bit parallel Adder has ⇒
  * (2n+1) ⇒ inputs OR "2n" inputs if first Adder is a HA. (least significant)
  * (n+1) ⇒ outputs
* There is a delay to propagate the carry from the least significant position to Most significant position.
* Total delay will be ⇒ n x delay of a single Adder.
* It is also called "RIPPLE CARRY ADDER".

FULL SUBTRACTOR USING FULL ADDER:

Full Adder

$$\begin{cases} S = x \oplus y \oplus z \\ C = xy + yz + zx \end{cases}$$

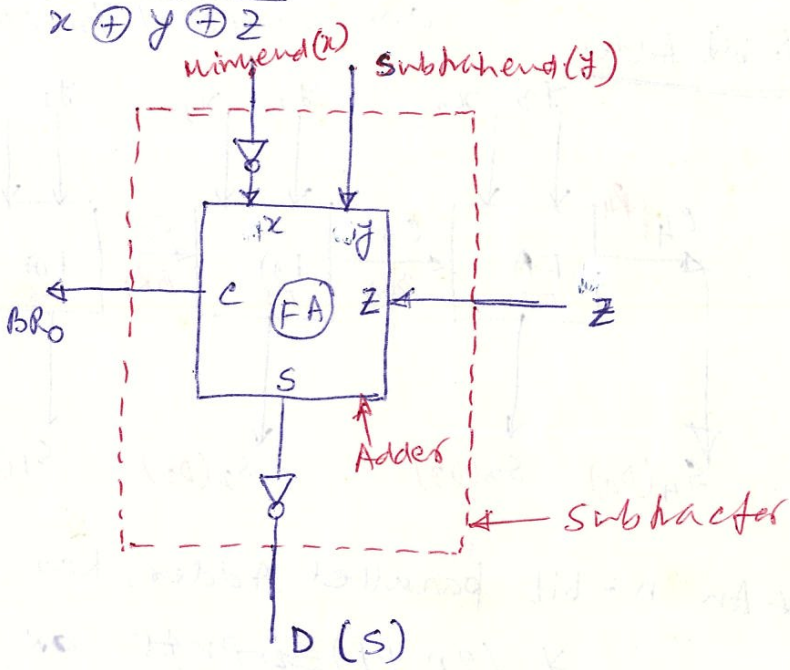
Full Subtractor

$$\begin{cases} D = x \oplus y \oplus z \\ BR_0 = \bar{x}y + yz + \bar{x}z \end{cases}$$

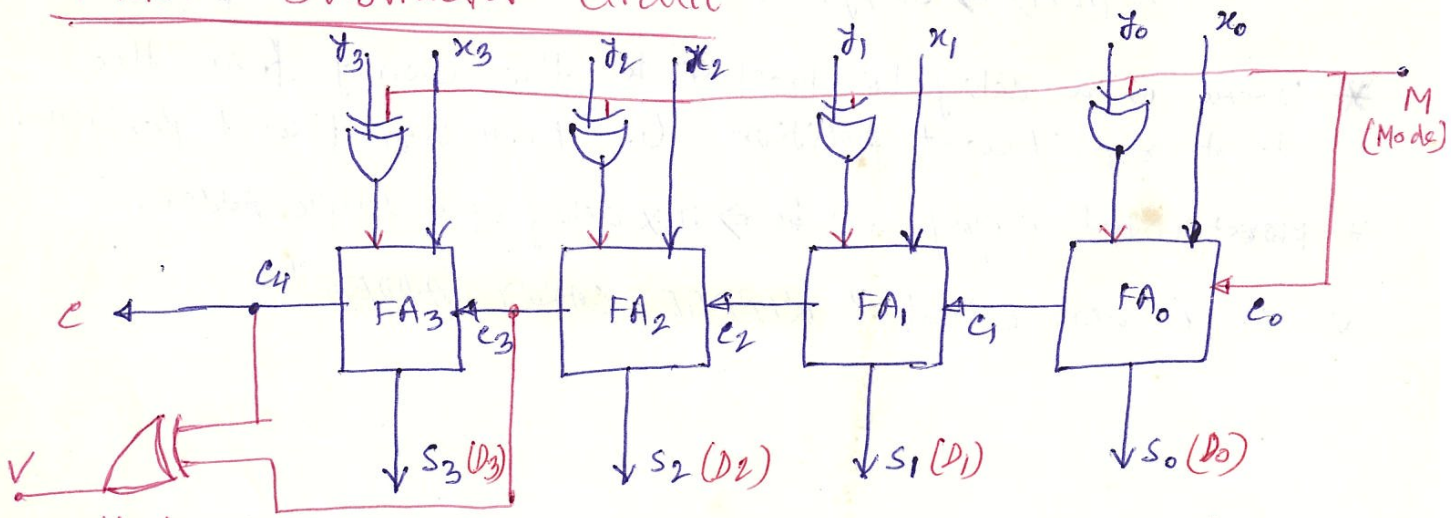
∴ Again,

$$\bar{x} \oplus y \oplus z = \overline{x \oplus y \oplus z}$$

∴ the circuit is



Binary Adder - Subtractor Circuit.



(overflow detection)
V = 1 (overflow)
V = 0 (no overflow)

When M = 1, then y becomes 2' complement
Hence, we get, x − y

M = 0 ⇒ Adder Mode
M = 1 ⇒ Subtractor Mode

Adders with Carry Lookahead Generator

Disadvantages of
* The Ripple Carry Adder: [illegible] ⇒.
  * One has to wait till the carry propagate from Least significant position to Most significant position.
  * Hence the system becomes slow.

* This problem may be avoided by the technique of carry Lookahead Generator (CLA).

* By this technique all the carries are generated together (i.e. at the same time)

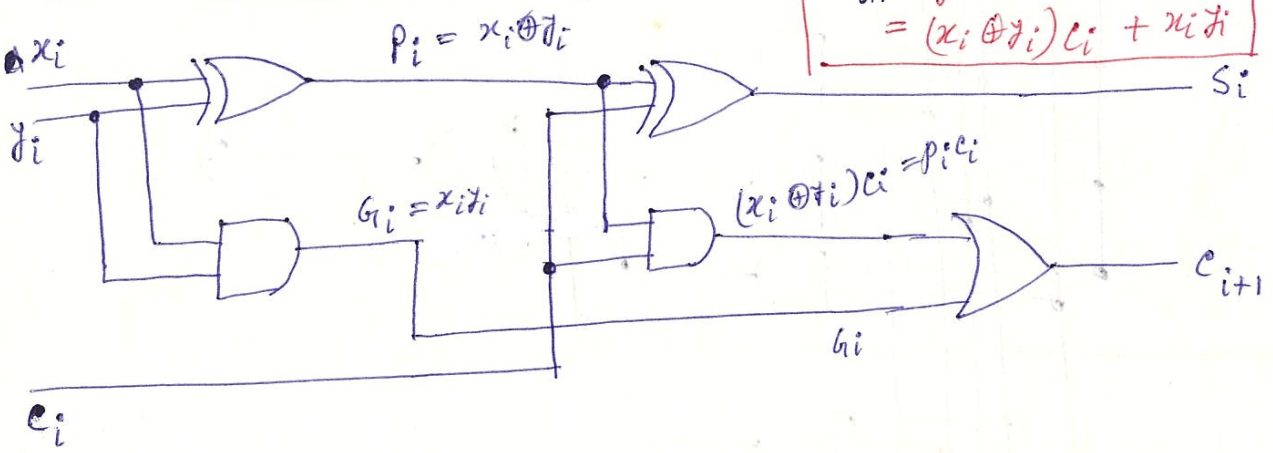
We knows
For FA:

$S_i = x_i \oplus y_i \oplus c_i$

$c_{i+1} = x_i y_i + y_i c_i + c_i x_i$

$= (x_i \oplus y_i) c_i + x_i y_i$

Single stage of a FA i



* P_i ⇒ carry Propagate
* G_i ⇒ Carry Generate

$P_i = x_i \oplus y_i$

$G_i = x_i y_i$

We know:

$S_i = x_i \oplus y_i \oplus c_i = P_i \oplus c_i$

$c_{i+1} = G_i + P_i c_i$

* Now let us write the Boolean expression for the carry output of each stage :

C_0 = input carry

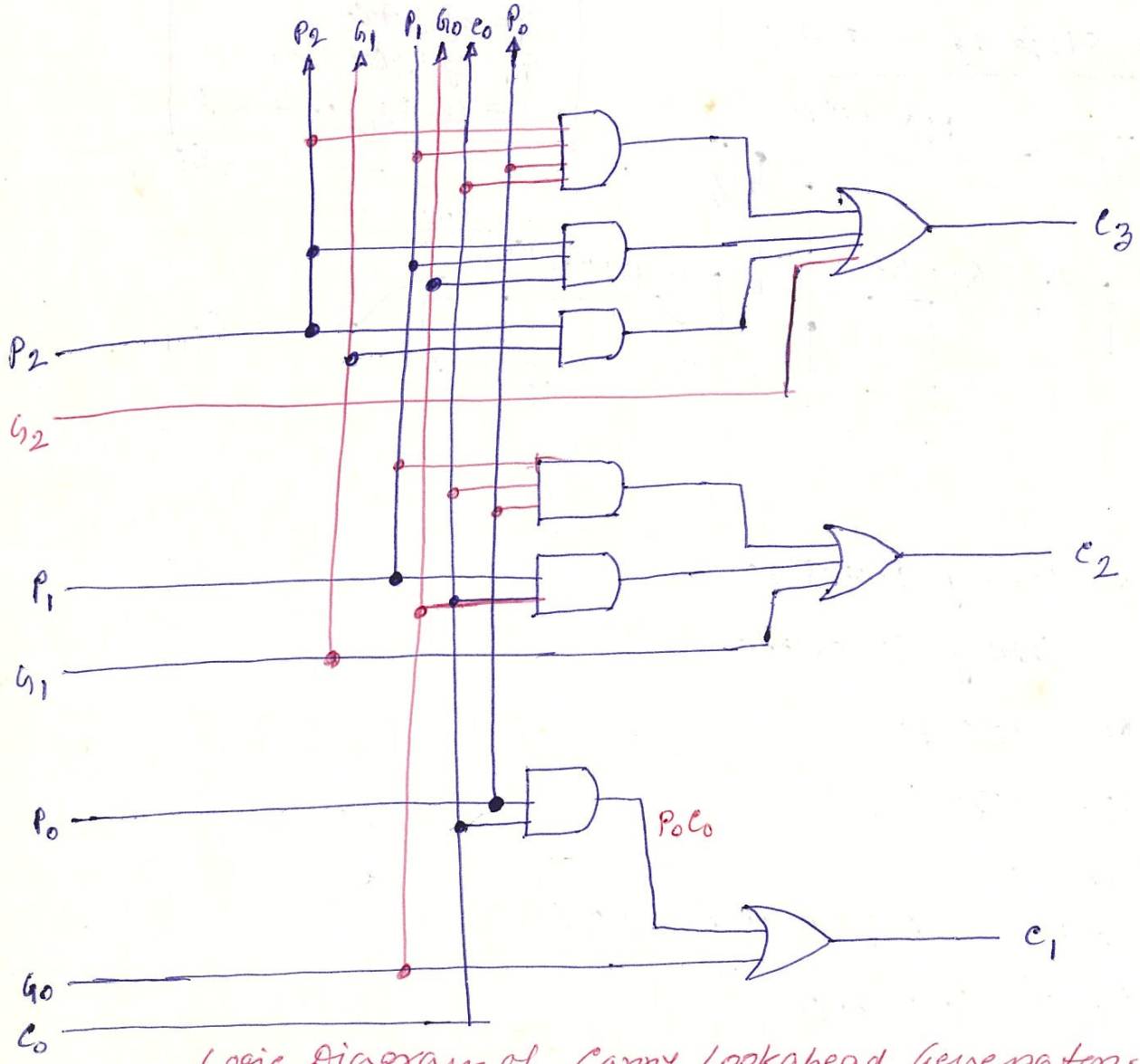
$C_1 = G_0 + P_0 C_0$ (carry output of the first stage)

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

$= G_1 + P_1 G_0 + P_1 P_0 C_0$ [carry o/p of the 2nd stage]

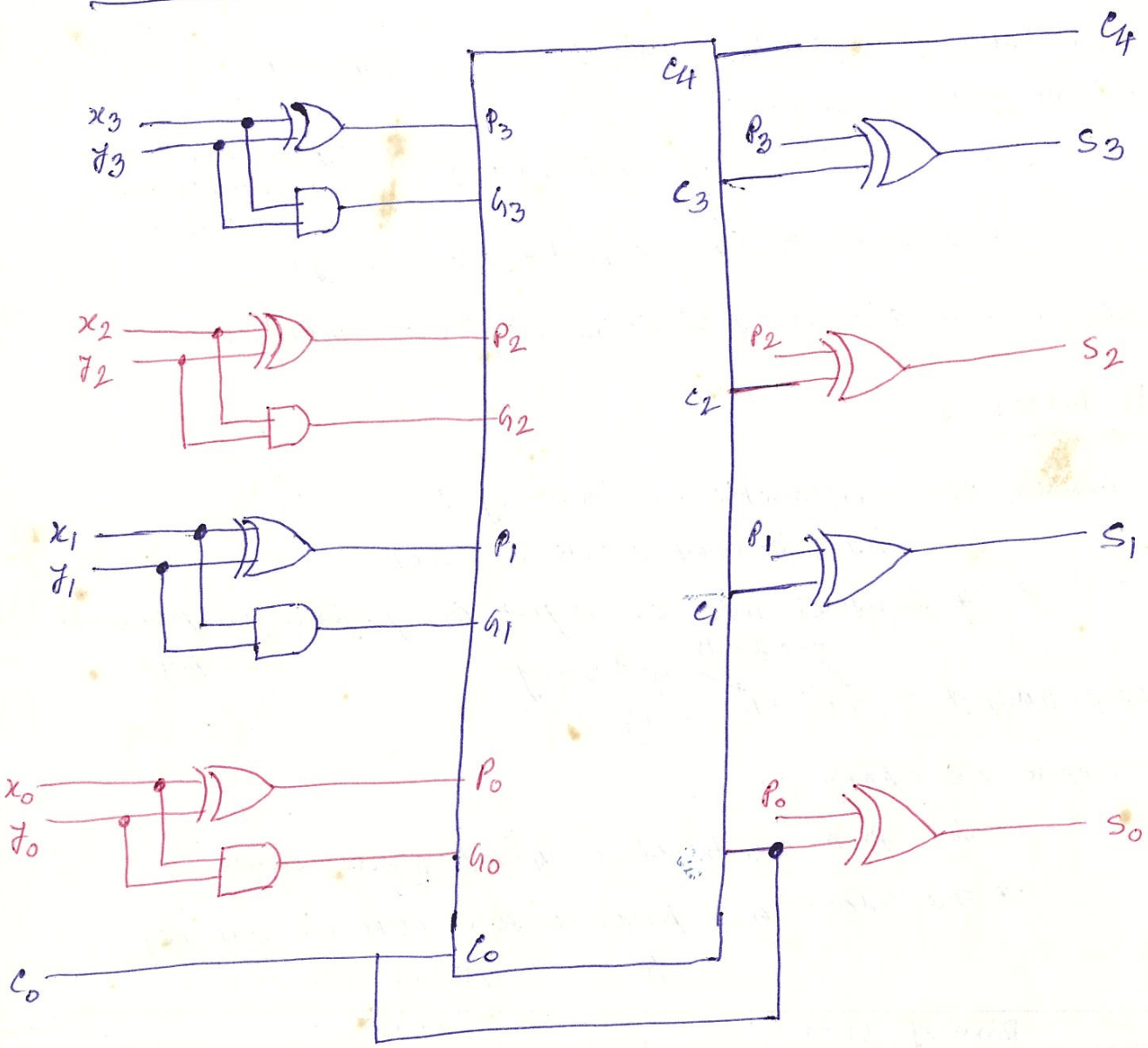
$C_3 = G_2 + P_2 C_2 = G_2 + P_2 (G_1 + P_1 G_0 + P_1 P_0 C_0)$

$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$.



Logic Diagram of Carry Lookahead Generator.

4-bit Adder with Carry Lookahead :



4-bit Adder with CLA

$$C_4 = G_3 + P_3 C_3 = G_3 + P_3(G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0)$$

$$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$$

DECIMAL ADDER (Page-662, DRC)

* Decimal codes: BCD (8421), 2421, Excess-3, 8 4 -2 -1
* A decimal adder requires:
  * a minimum of NINE (9) inputs (4-bit for each decimal digit + carry-in)
  * 5-outputs (4-bit output + 1 carry out)
* Here we will discuss BCD coded decimal Adder.

BCD Adder :

* Consider the arithmetic addition of :-
  * TWO decimal digits in BCD
  * together with an input-carry from the previous stage.
* Max. output = $\overbrace{9+9}^{\text{Two digits}} + \overbrace{1}^{\text{input carry}} = 19$
* Suppose we apply ⇒
  * two BCD digits to a 4-bit binary adder
  * The adder will produce the SUM in BINARY from 0 to 19.

Decimal NO.	Binary SUM					BCD Sum					Decimal N
	(carry) K	Z_4	Z_3	Z_2	Z_1	(carry) C	S_3	S_2	S_1	S_0	
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1	1
2	0	0	0	1	0	0	0	0	1	0	2
3	0	0	0	1	1	0	0	0	1	1	3
4	0	0	1	0	0	0	0	1	0	0	4
5	0	0	1	0	1	0	0	1	0	1	5
6	0	0	1	1	0	0	0	1	1	0	6
7	0	0	1	1	1	0	0	1	1	1	7
8	0	1	0	0	0	0	1	0	0	0	8
9	0	1	0	0	1	0	1	0	0	1	9
10	0	1	0	1	0	1	0	0	0	0	10
11	0	1	0	1	1	1	0	0	0	1	11
12	0	1	1	0	0	1	0	0	1	0	12
13	0	1	1	0	1	1	0	0	1	1	13
14	0	1	1	1	0	1	0	1	0	0	14
15	0	1	1	1	1	1	0	1	0	1	15
16	1	0	0	0	0	1	0	1	1	0	16
17	1	0	0	0	1	1	0	1	1	1	17
18	1	0	0	1	0	1	1	0	0	0	18
19	1	0	0	1	1	1	1	0	0	1	19

* By inspection ⇒

From 0 — 9 ⇒ BCD Sum & Binary Sum are identical

From 10 — 19 ⇒ Addition of 6(0110) with binary sum gives correct BCD representation

* Development of logic circuit for Detection and Necessary Correction from the Content of Table :-

* Here we will develope a circuit to detect when 6(0110) to be added with Binary sum

Situations when ⇒ we Need Correction are :-

First ⇒ * When $K=1$ (16 — 19)

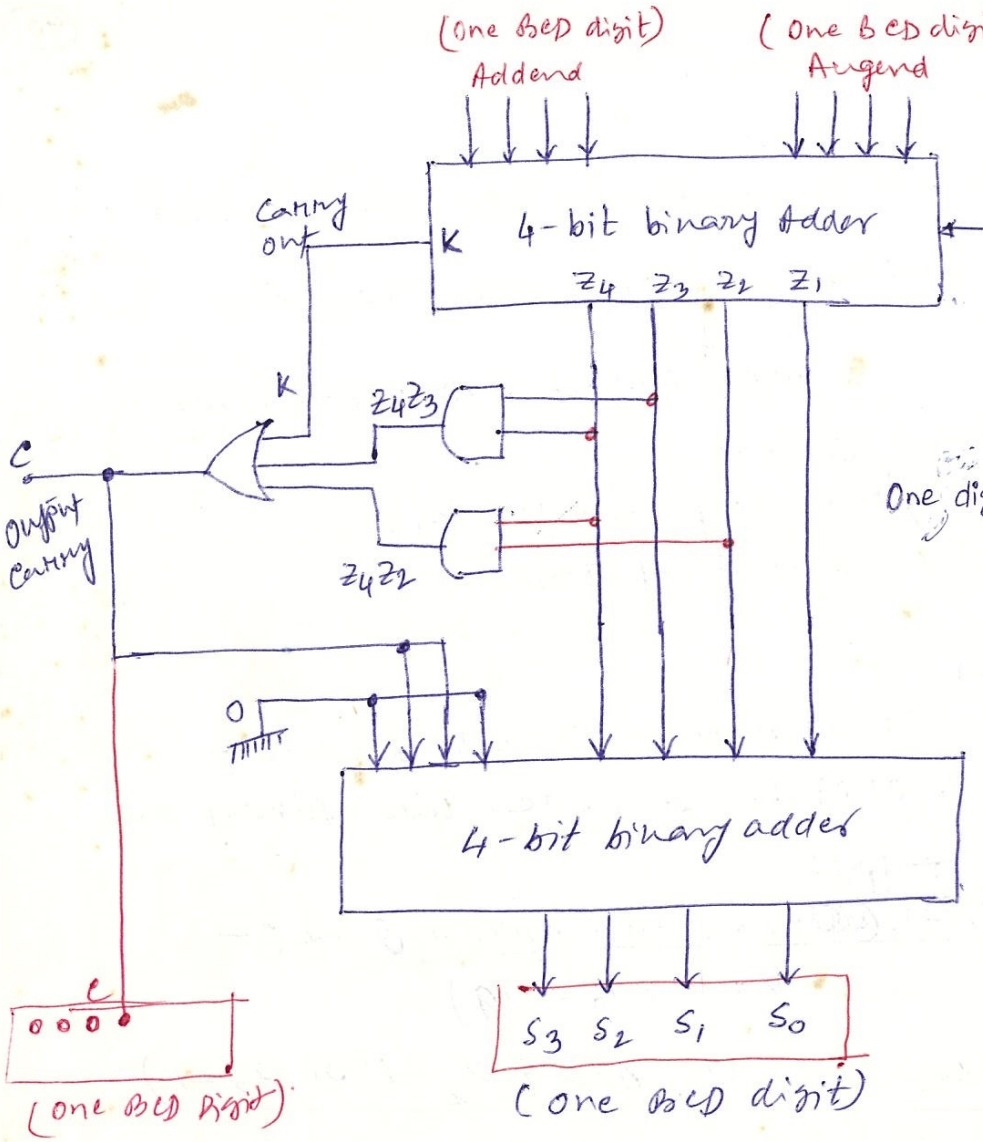
Second * When $Z_4 = 1$ And ($Z_2 = 1$ or $Z_3 = 1$)

The corresponding Boolean expression for C ⇒

$$C = K + Z_4 Z_3 + Z_4 Z_2.$$

* When $C=1$ ⇒ * then 6(0110) has to be added with the binary sum

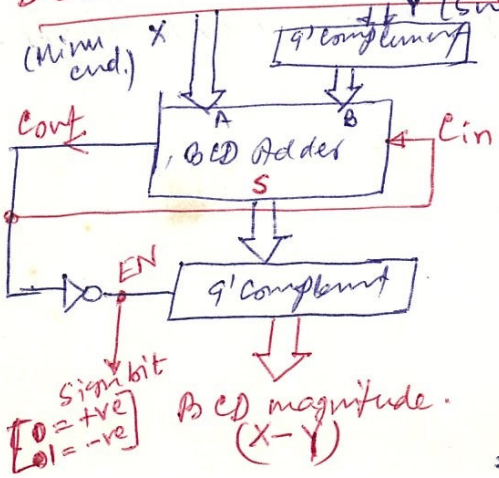
* And this $C=1$ will provide a Carry output to the next stage.



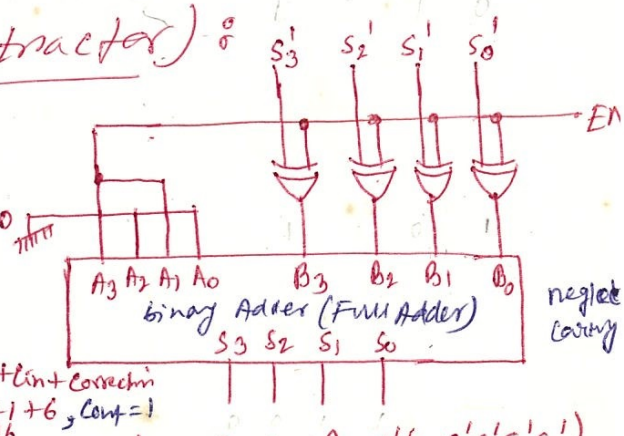
Logic diagram of One digit BCD Adder

* For addition of n-digit decimal number ⇒
  * we require n-BCD adder (or 2n Binary adder) in cascade.
  * The output carry of one stage must be connected to the input carry of the next higher order stage.

Decimal subtractor (BCD subtractor)



X = in BCD
Y = in BCD.



Case-I
$X > Y$, $X+(9-Y)+Cin+Correction$
$= X+9-Y+1+6$, Cout = 1
$= X-Y+16$

Case-II
$X \leq Y$
$X+(9-Y)+Cin+correction$
$= X+9-Y \leq 9$, Cout = 0, Correction = 0

9' complement of $S'(=S_3'S_2'S_1'S_0')$

9' complement Enabled.

Problem. Assume that the time required to generate the carry-out bit from 1-bit full adder is 25 nsec and that of sum bit is 45 nsec. What is the maximum rate of addition when eight 1-bit full adders are cascaded.

Soln:

$7 \times 25 = 175$ nsec.

Carry+Sum = $175 + 45 = 220$ nsec.

$8 \times 25 = 200$ nsec.

∴ Worst case time requirement = 220 nsec.

∴ maximum rate of addition = $\frac{1}{220 \text{ nsec}}$.

$= \frac{1}{220} \times 10^9$ sec.

$= \frac{1}{2.2} \times 10^7 / \text{sec}$.

Problem Modify the ckt of BCD adder so that same ckt may act as a BCD adder + BCD subtractor with external command input ADD/SUB.

Problem How can the subtractor(decimal) ckt discussed above be used for multi-digit subtraction.

BINARY MULTIPLIER

* Multiplication of binary number is performed in the same way as in decimal numbers.

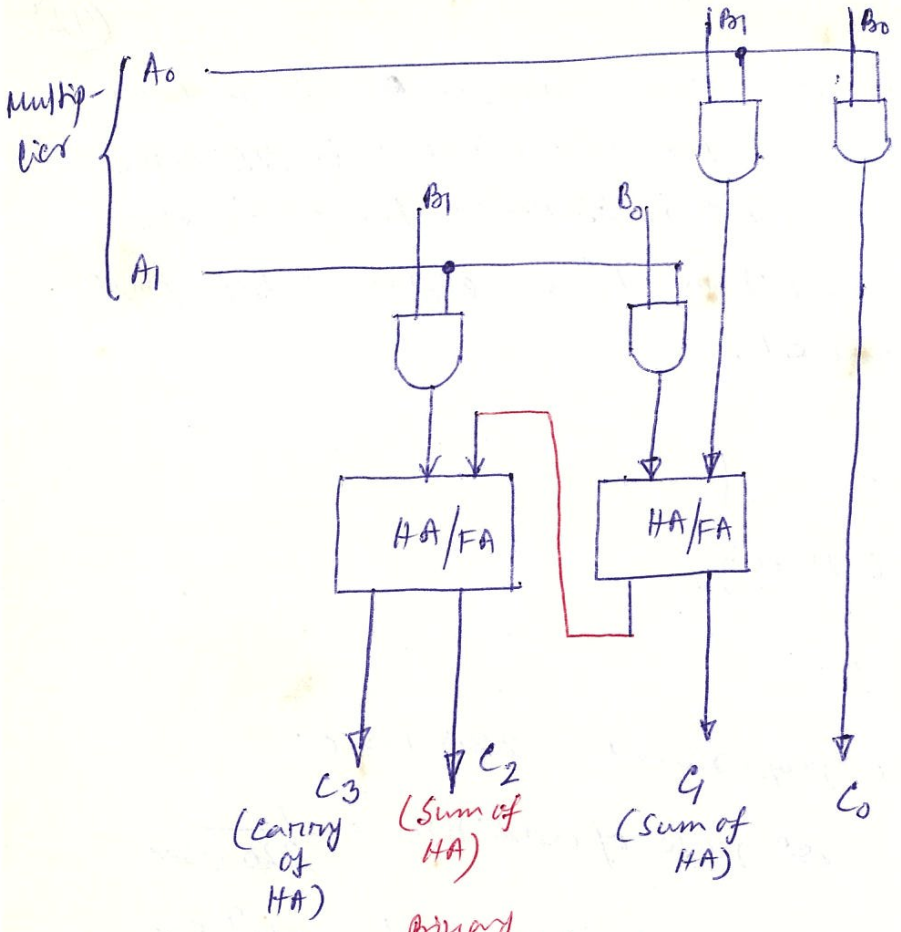
* Multiplicand ⇒ B_1 B_0

  Multiplier ⇒ A_1 A_0

B_1A_0 A_0B_0 ← Partial Products

shifted partial product → A_1B_1 A_1B_0 (+)

C_3 C_2 C_1 C_0



2-bit X 2-bit Binary Multiplier

4-bit X 3-bit Binary Multiplier

B_3 B_2 B_1 B_0 – Multiplicand

A_2 A_1 A_0 – Multiplier

A_0B_3 A_0B_2 A_0B_1 A_0B_0

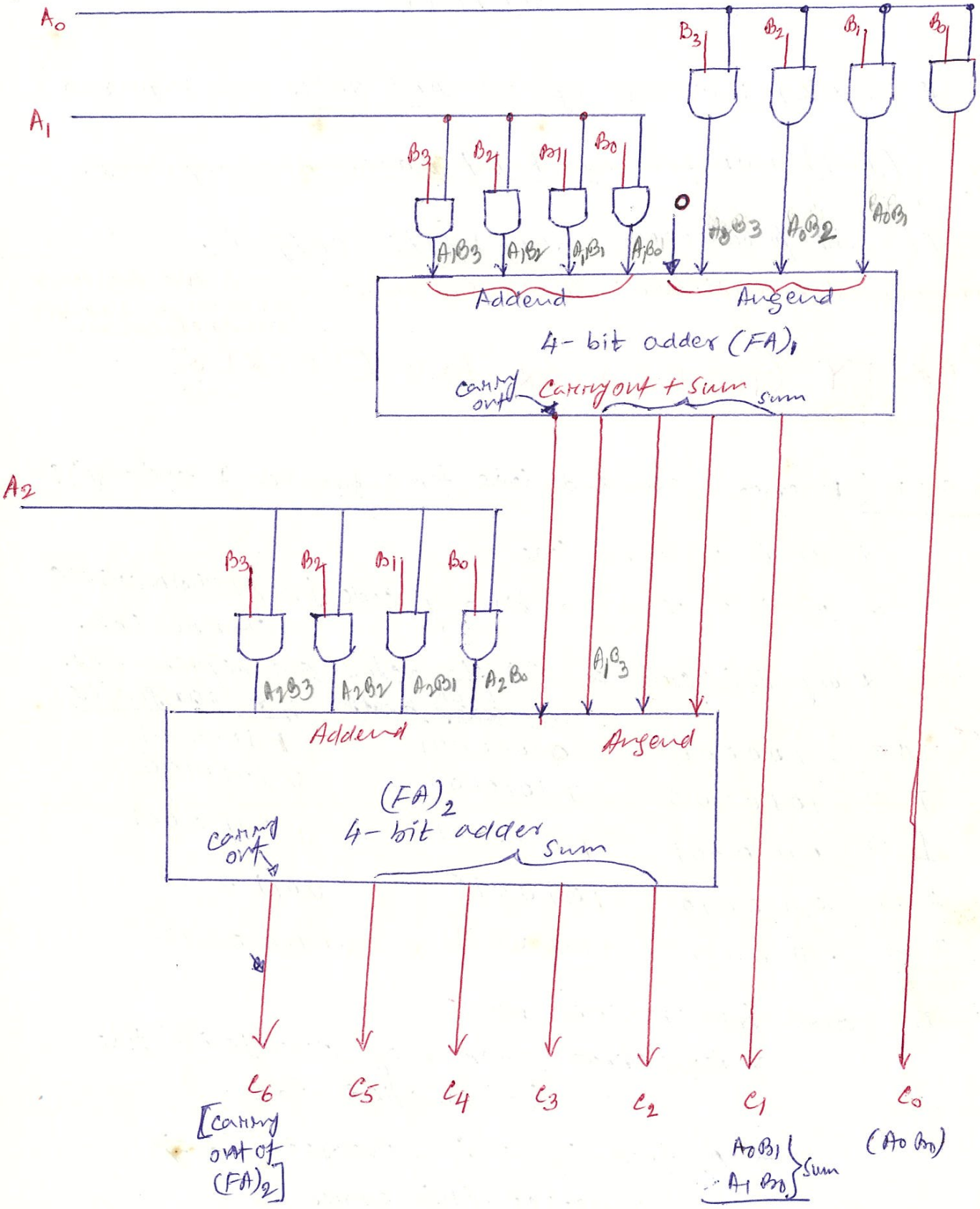
A_1B_3 A_1B_2 A_1B_1 A_1B_0

A_2B_3 A_2B_2 A_2B_1 A_2B_0

~~A_3B_3 A_3B_2 A_3B_1 A_3B_0~~

C_6 C_5 C_4 C_3 C_2 C_1 C_0

Logic circuit for 4 × 3-bit Binary Multiplier ⇒



* For K-bit Multiplicand
  J-bit Multiplier.

* (J X K) numbers of ⇒ AND gates are required
  (J-1) numbers of K-bit adders are required.

* It will produce a (J+K) bits output.

Commercial multiplier: IC 74284/74285 ⇒ This two together performs a 4X4-bit multiplication.

PARITY GENERATOR AND CHECKER

ASCII (American Standard Code for Information Interchange):-

* It is a 7-bit code
* It is a standard binary code for ALPHANUMERIC CHARACTERS.
* By this code a 128 characters are represented.

		with EVEN parity	with odd parity
A ⇒	1000001	01000001	11000001
T ⇒	1010100	11010100	01010100
1 ⇒	0110001	10110001	00110001
2 ⇒	0110010	10110010	00110010
3 ⇒	0110011	00110011	10110011

* In data transmission ⇒
  * Errors may develope in the transmitted data.
  * A technique is developed to ⇒ detect this error
  * In this technique an extra bit is used with the transmitted data ⇒.
  * This extra bit is called "PARITY BIT".

PARITY BIT :

(may be 0 or 1)

This is a bit which is included with the every character of the message to make total number of 1's either EVEN or ODD.

ODD PARITY : If total number of 1's after including the parity bit becomes ODD, then this parity bit is called ODD PARITY.

EVEN PARITY: Other wise EVEN PARITY.

* The parity bits are always included at the MSB position.

Error Detection and Remedy.

* Transmit 8-bit character which includes parity bit (say Even parity)
* Receive at the destination
* Check the parity of the received data at the receiving end.
* If no error is detected ⇒ then the receiver sends back an ACK (acknowledge) control character 00000110.

* If any of the received character is not even i.e. an error is detected ⇒ then the receiver sends back NAK (negative acknowledge) control character 10010101. with even parity.

* The sending end will respond to an NAK ⇒
  * By transmitting the message again until the correct parity is received

Parity Generator: The circuit that generates the parity bit in the transmitter.

Parity checker: The circuit that checks the parity in the receiver.

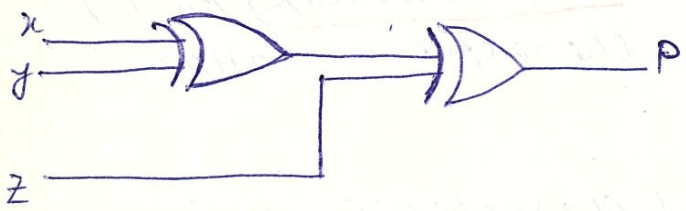
Design of Parity Generator:

x \ yz	$\bar{y}\bar{z}$	$\bar{y}z$	$y\bar{z}$	yz
$\bar{x}$	0	1	0	1
x	1	0	1	0

$$\therefore P = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$$
$$= (\bar{x}\bar{y} + xy)z + (\bar{x}y + x\bar{y})\bar{z}$$
$$= \overline{(x \oplus y)}z + (x \oplus y)\bar{z}$$
$$= x \oplus y \oplus z.$$

Even Parity.

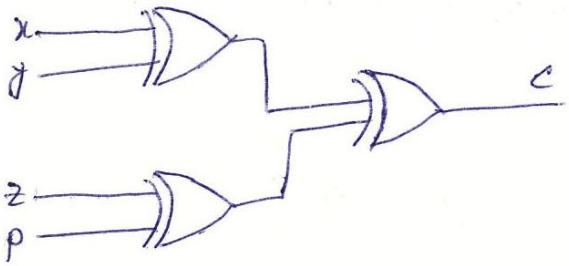
Three bit Message			Parity bit
x	y	z	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



Parity Checker

xy \ zp	$\bar{z}\bar{p}$	$\bar{z}p$	$z\bar{p}$	zp
$\bar{x}\bar{y}$	0	1	0	1
$\bar{x}y$	1	0	1	0
$x\bar{y}$	0	1	0	1
xy	1	0	1	0

$$C = x \oplus y \oplus z \oplus p$$



Even Parity checker.

Four Bits Received				Parity Error checker.
x	y	z	p	C
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

* This checker may be converted to 3-bit parity generator by putting $p=0$ and $C=P$.

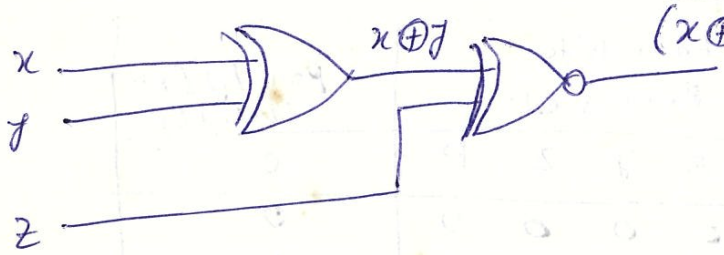
ODD Parity Generator & Checker

* From the truth table of Even parity generater and checker, we observe that
* the odd parity generator or checker is
* Parity bit (P) for generater } will be
* Parity Error checker bit (C) } first complement of the Even parity case.

* Hence It is implemented with XOR gates and the gate at the output must be EX-NOR gate.

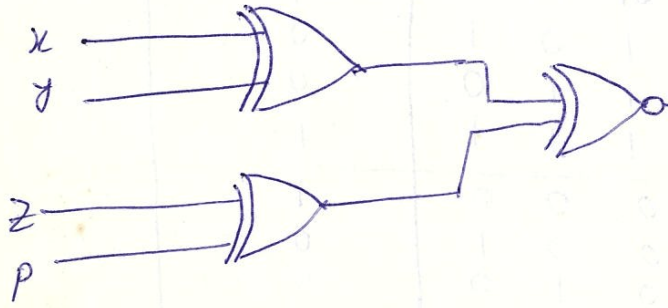
ODD Parity.

i.e. Generator :



$$\overline{(x \oplus y) \oplus z} = x \oplus y \odot z = p$$

ODD Parity checker.



$$c = \overline{(x \oplus y) \oplus (z \oplus p)}$$

$$= (x \oplus y) \odot (z \oplus p)$$

Commercial Parity Generator/checker Circuits :-

IC 74180 — 14 Pins } 9-bit parity

IC 74280 — 14 Pins }

MAGNITUDE COMPARATOR

* A magnitude comparator is a combinational circuit that compares two numbers A & B and determines their relative magnitude.
* The outcome of comparator are three binary variables ⇒ that indicates
  * $A > B$
  * $A = B$
  * $A < B$.

Design technique:

$A = B$.

Let, $A = A_3\ A_2\ A_1\ A_0$, $B = B_3\ B_2\ B_1\ B_0$ } four bits number in binary.

* $A = B$, only when all the corresponding bits of A & B are equal.
* i.e, $A_3 = B_3$, $A_2 = B_2$, $A_1 = B_1$, & $A_0 = B_0$.
* The equality relation of each pair of bits can be expressed by ~~XOR function~~ EX-NOR function as

(EX-NOR) ⇒ $x_i = A_i B_i + \bar{A_i}\,\bar{B_i}$, for $i = 0, 1, 2, 3$

* $\boxed{F_{A=B} = x_3\, x_2\, x_1\, x_0}$ i.e. when all the x_i's are 1 then $A = B$, and $F_{A=B} = 1$.

A > B and A < B

* To determine A > B, or A < B
* we inspect the relative magnitude of pairs of significant bits ⇒
    * starting from the MSB.
* If two bits are equal ⇒
    * we compare the next lower significant pair of bits.
* This comparison continues until a pair of unequal bits is reached. ⇒
* If the corresponding bit of A = 1 and B = 0 ⇒
    * then A > B.
* But if the corresponding bit of A = 0 and B = 1 ⇒
    * Then A < B.
* Logical expression of the sequential comparison are :

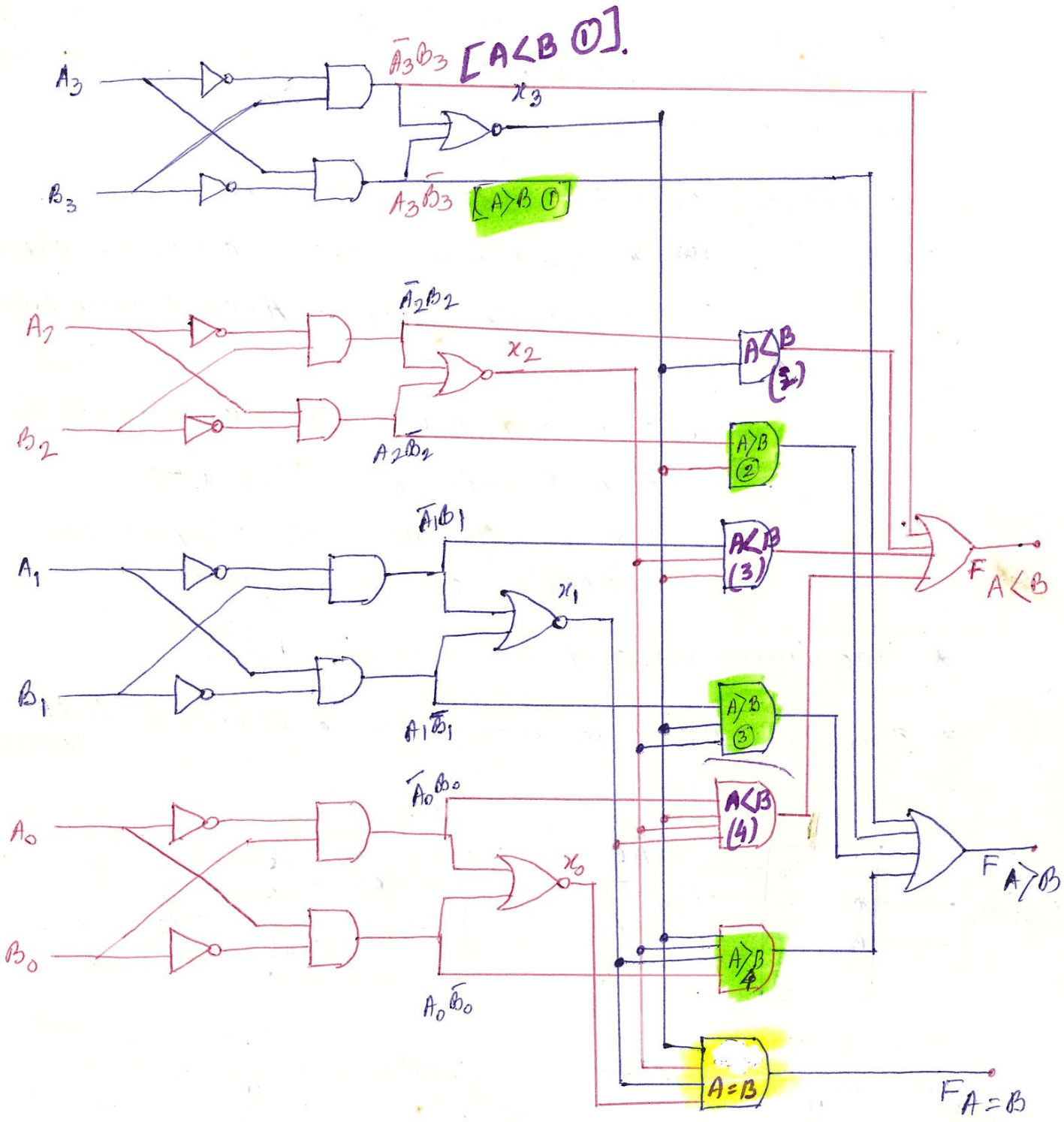
$$F_{A>B} = \underset{(1)}{A_3 \bar{B}_3} + \underset{(2)}{x_3 A_2 \bar{B}_2} + \underset{(3)}{x_3 x_2 A_1 \bar{B}_1} + \underset{(4)}{x_3 x_2 x_1 A_0 \bar{B}_0}$$

$$F_{A<B} = \underset{(1)}{\bar{A}_3 B_3} + \underset{(2)}{x_3 \bar{A}_2 B_2} + \underset{(3)}{x_3 x_2 \bar{A}_1 B_1} + \underset{(4)}{x_3 x_2 x_1 A_0 \bar{B}_0}$$

If, $F_{A>B} = 1$, then A > B

$F_{A<B} = 1$, then A < B.

Logic Circuit of 4-bit Magnitude Comparator:



Commercially available comparator IC:

IC 7485 ⇒ 4-bit magnitude comparator

IC 74682/74683/74684/74685/74686 ⇒ 8-bit comparator.

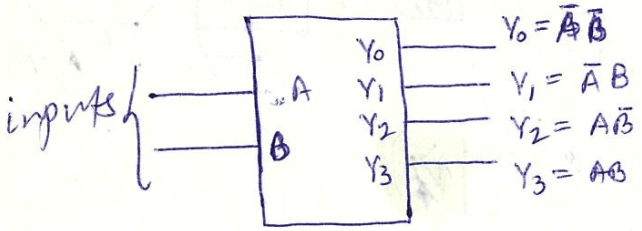
DECODER

* It is a combinational circuit that converts binary information from n-bit input to m number of mutually exclusive outputs.

* Mutually exclusive outputs ⇒ means ⇒
  * ⇒ that the decoder circuit makes only one output Active (active Low or Active High) ⇒
  * ⇒ Among the m-possible outputs ⇒
  * ⇒ And this is unique ⇒
  * ⇒ And depends on the input code applied at the input.

* maximum value of m may be 2^n.

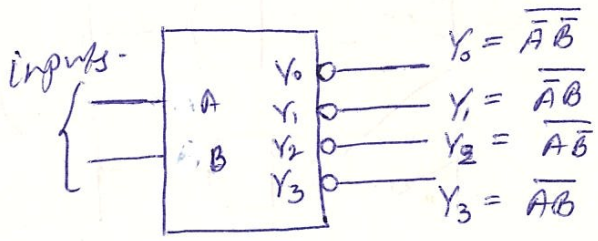
* This is called ... n-to-m-line DECODER. (n×m Decoder)



2-to-4 Decoder with ACTIVE HIGH outputs.

Truth Table:

inputs		outputs			
A	B	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



2-to-4 Decoder with ACTIVE LOW outputs.

Truth Table:

inputs		outputs			
A	B	Y_3	Y_2	Y_1	Y_0
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

* **Binary Decoder** : * A decoder with n-input and 2^n outputs is called Binary Decoder.
  * Because this type of decoder generates all the minterms ⇒
  * which are possible with n-bit binary inputs.

* 2-to-4
* 3-to-8
* 4-to-16

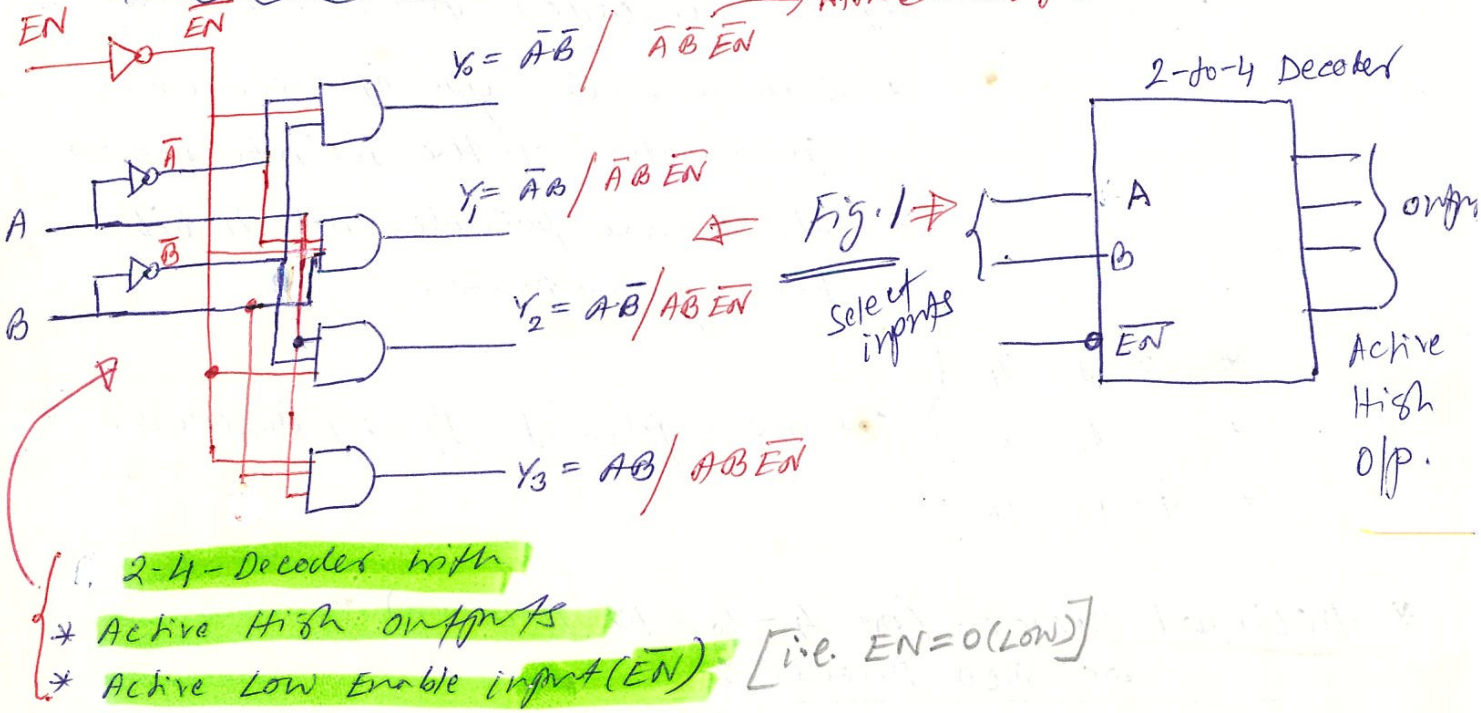
} are examples of Binary decoders.

* **Decimal Decoder (or 4-to-10 Decoder) : or BCD Decoder :-**
  * Here BCD codes are used as the input and a maximum of 10 valid outputs are obtained.
    * For six non-BCD codes (1010 to 1111) used as inputs ⇒
    * The outputs will be in the inactive state.
      * i.e * all 0s for Active High output.
        * all 1s for Active Low output.

* ~~Excess-3 decoder~~
* ~~Binary-to-Excess-3 Decoder~~

LOGIC CIRCUIT

2-to-4-bit Decoder for LOGIC HIGH OUTPUTS:

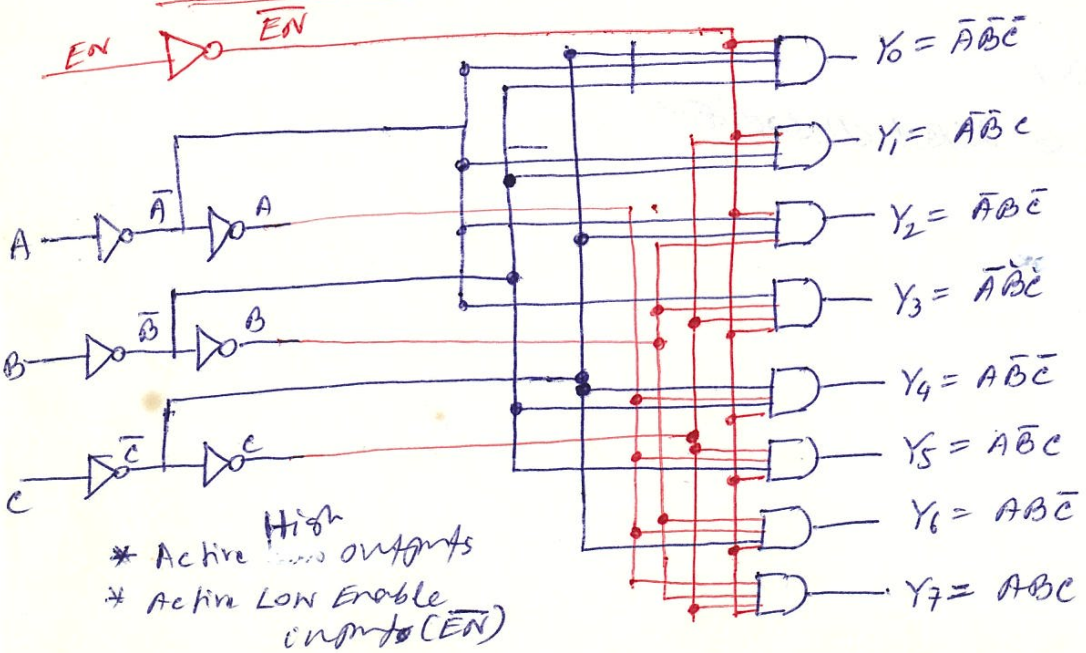


1. 2-4-Decoder with
   * Active High outputs
   * Active Low Enable input ($\bar{EN}$) [i.e. EN = 0 (LOW)]

2-to-4-Decoder with.
* Active LOW outputs
* Active LOW Enable input ($\bar{EN}$)

} This circuit is obtained by first inverting the output of fig. 1 i.e by putting NAND gate instead of AND in fig. 1.

3-to-8 Line Decoder.

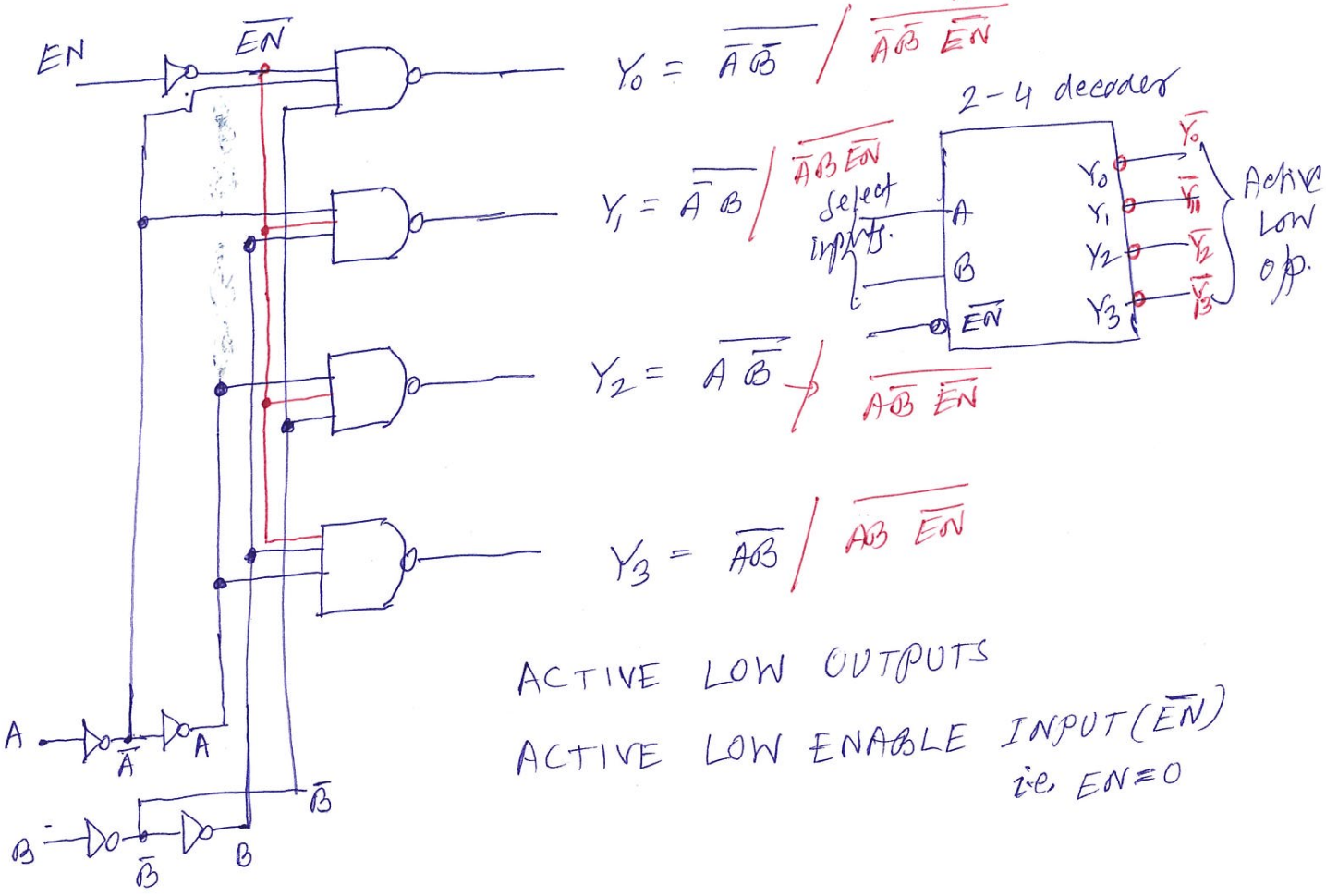


* Active High outputs
* Active LOW Enable input ($\bar{EN}$)

* Two inverters in series for each Decoder inputs A or B have been used to reduce loading to the external circuit driving the A and B inputs
* To get active Low outputs ⇒ the AND gates are replaced by NAND gates.

~~Decoder~~

2 to 4 Decoder For LOGIC LOW OUTPUTS



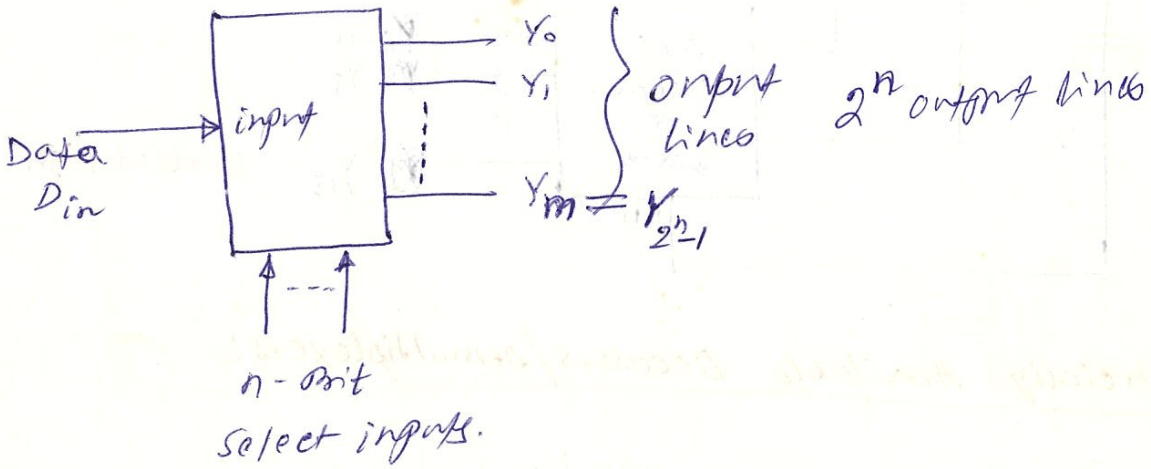
ACTIVE LOW OUTPUTS

ACTIVE LOW ENABLE INPUT ($\overline{EN}$)
i.e. EN = 0

DEMULTIPLEXER :

* This is a circuit that receives data on an input line and routes this received data on one of 2^n possible output lines.

* Hence, a Decoder with Enable Input can function as a Demultiplexer (DMUX).



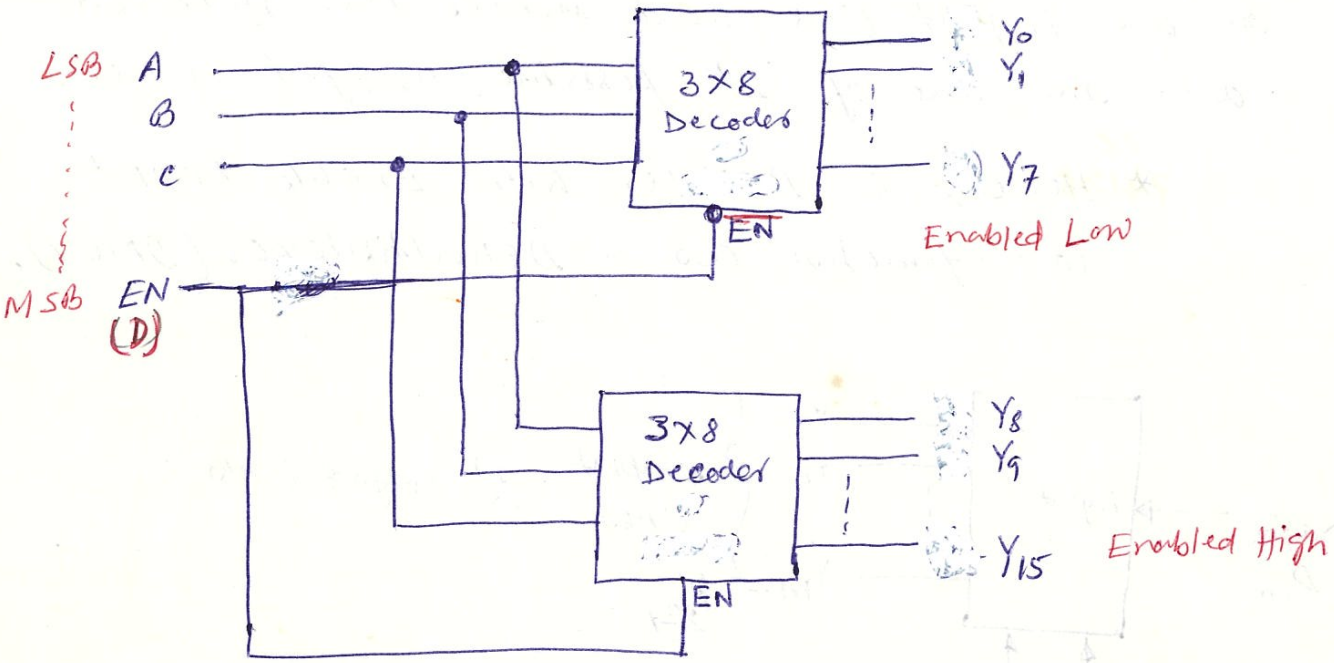
* Hence any decoder having Enable input can be used as DEMULTIPLEXER.

Cascading of Decoders/Demultiplexers

* commercially available decoders/ ... Demultiplexers have ⇒
  * limited numbers of outputs (2^n maximum)
  * no. of outputs may be increased by cascading the available Decoders/DMUX.
  * This can be done with the help of Enable inputs or other suitable Decoders.

4X16 Decoder with TWO 3X8 Decoder

D C B A
1 0 0 0 → 8



Commercially Available Decoders/Demultiplexers:

IC 74139 — 16 Pins — contains:
* 2 fully independent 2X4 Decoder/DMUX
* Active low separate Enable input: $\overline{EN1}$ / $\overline{EN2}$
* 4-Active LOW outputs (Separate)
* 2-Active High inputs (Separate).

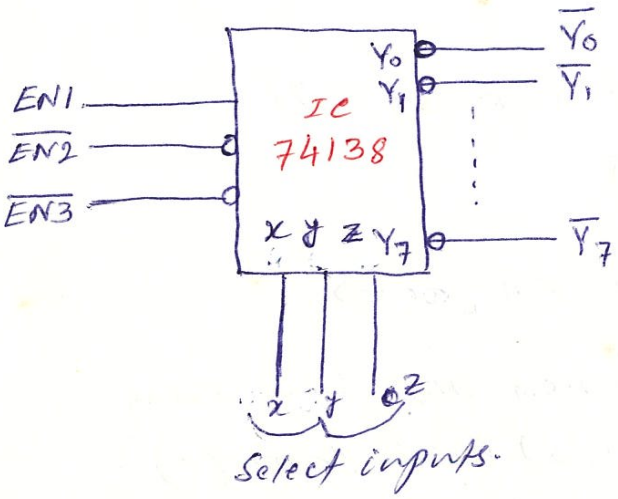
Ic 74155 — 16 Pins. Contains:
* 2 units of 2X4 Decoder/DMUX
* 2 common address inputs
* Each unit has ⇒ 1 active Low ($\overline{EN}$) / 1 active High (EN) Enable inputs.
* 4 separate outputs of each units ⇒ Active LOW.

Ic 74156 — 16 Pins ⇒ Same as 74155, but the outputs are open collector type.

IC 74239 — 16 Pins ⇒ Same as Ic 74139, but outputs are Active High.

IC 74138 16 pins contains:

* 3×8 Decoder / DMUX.
* Select Inputs : 3
* Outputs : 8 ⇒ Active Low
* Enable inputs : 3 ⇒ one High Enable
  2 - Low Enable.



* To make this chip Enable ⇒ The Enable inputs together have to be as follows:-
  $EN1 = 1$ (High)
  $\overline{EN2} = \overline{EN3} = 0$ (LOW).
* Otherwise all the outputs will remain in High State.

Combinational Logic Implementation Using DECODER:

* A decoder produces 2^n minterms for n-input variables.
* As Any Boolean function ⇒
  * can be expressed in the sum-of-minterms.
  * Hence, we get any combinational logic implemented by
    * Using DECODER to generate Minterms
    * And One OR gate at the output to form the each logic Sum.
* Hence any combinational circuit ⇒ With
  no. of inputs = n
  no. of outputs = m
  * can be implemented with ⇒
    * 1 ⇒ ~~n~~ $n \times 2^n$ line Decoder

    And * ~~m~~ 1 ⇒ m-input OR gate.

Ex: ~~Binary Subtractor~~ Design of 1-bit Full Adder With DECODER

* It has ⇒.
  * 2-significant bit inputs (x, y)
  plus * 1-bit Carry (z).

∴ From the truth table of FA ⇒.

* We have obtained the functions for the sum (s) and carry (c) in term of ... MINTERMS as

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

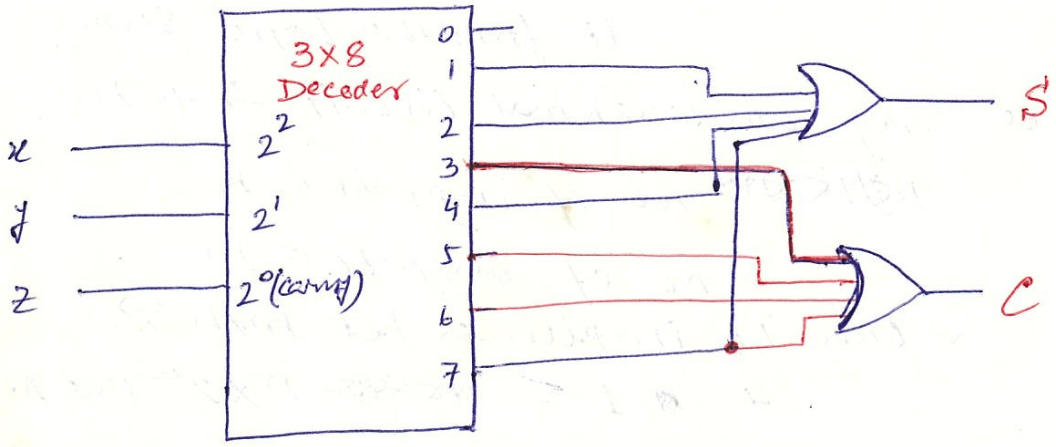
$C(x,y,z) = \sum m(3, 5, 6, 7)$

* Since here ⇒

No. of inputs = 3 (x, y, z)

No. of Minterms require to implement S & C = 7

* ∴ * We require a 3 X 8 Decoder and * ~~one~~ two 4-inputs OR Gate.

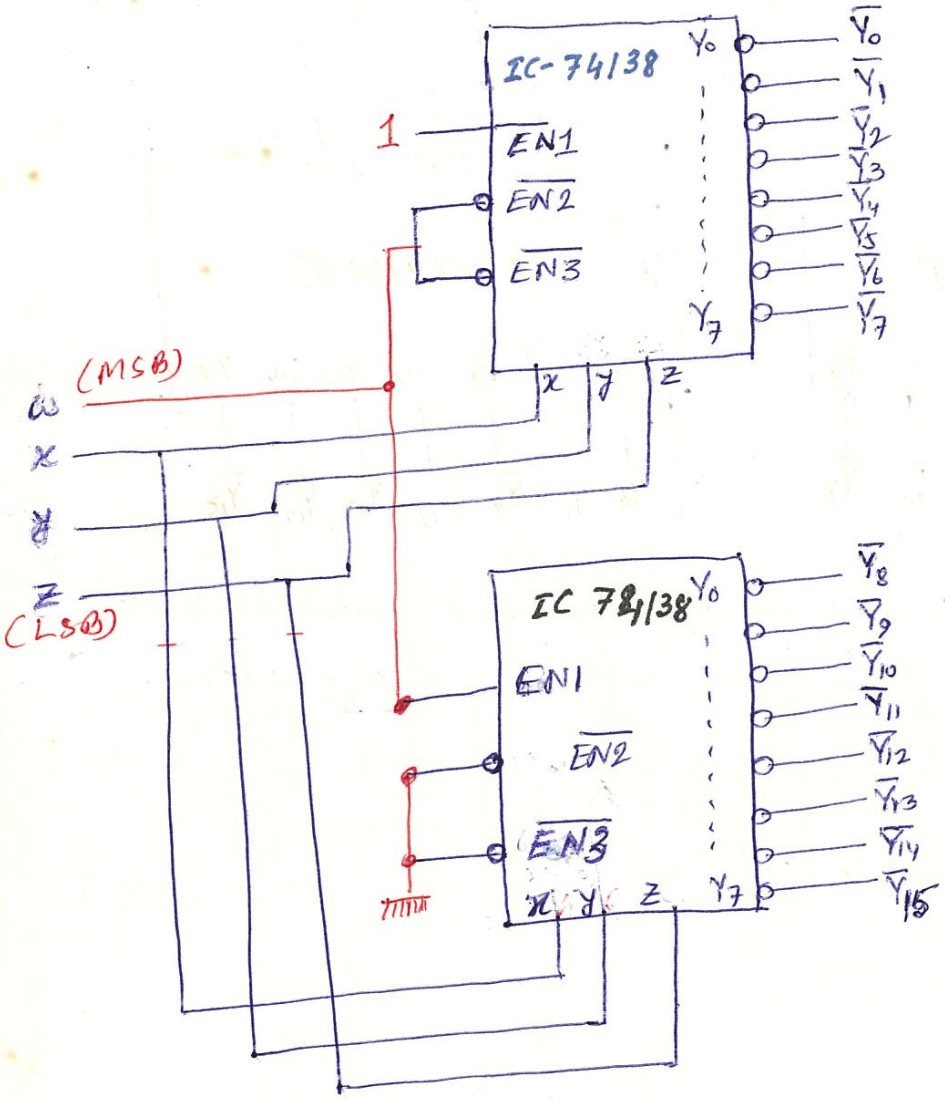


Problem:

Design an equivalent 4×16 decoder by cascading two IC-74138 without using any other gates.

IC74138 ⇒ * 3×8 Decoder/DMUX * 8- Low Active o/ps.
* Three enable inputs ⇒ EN1, $\overline{EN2}$, $\overline{EN3}$

Soln:



IC 7442 · 16 pins

(BCD to Decimal Decoder)

contains: * 10 : Active low output
* 4 : Inputs
* No Enable Input

IC-7445 16 pins
(BCD to Decimal)

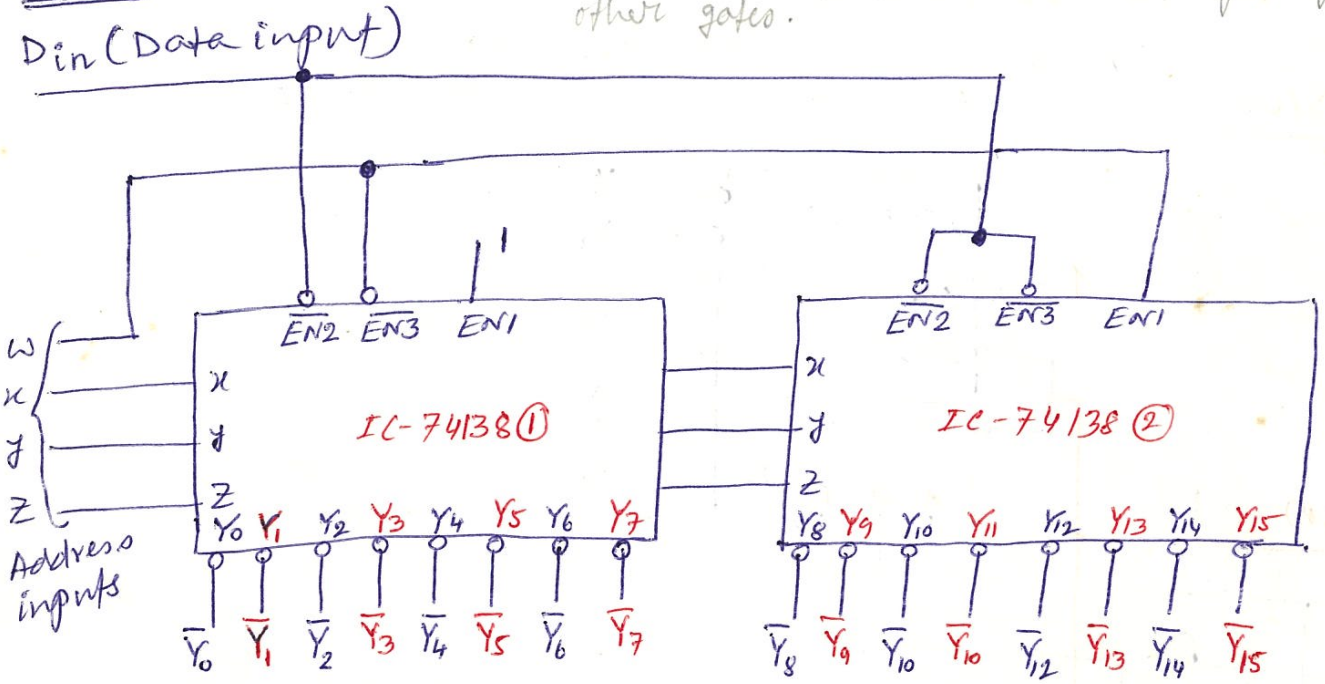
IC-7443 ⇒ Excess-3-to-Decimal Decoder

IC-7444 ⇒ Gray-to-Decimal Decoder

Problem -2

cascade 74138 ICs to design a 1 x16 demultiplexer. with Active low output without using any other gates.

Soln:



Data input Din is always Low. from external sources. It decides when data will be sent to selected address.