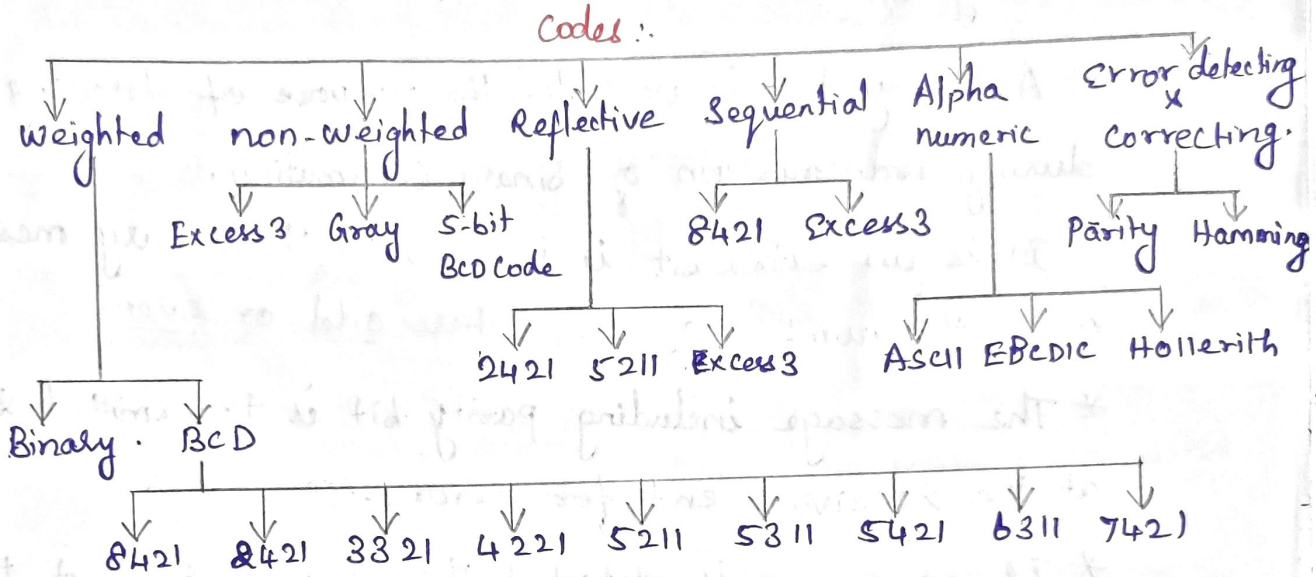
Binary Codes :-

* Digital data is represented, stored and transmitted as group of binary digits. The group of bits is also known as binary code - which represents both numeric or alphanumeric

Classification of binary codes.



Error detecting and Correcting Codes :-

Introduction :-

* when digital information in binary form is transmitted from one circuit to another circuit error may occur (ie 0 may change to 1 & viceversa) due to presence of noise.

* To avoid error b/w transmitter & receiver - extra bit or more than one bit are added in the data called as parity bit

* The extra bit allow the detection & sometime correction of error in the data.

* The data along with the parity bit forms the code.
* Codes which allow only error detection are called error detecting codes.
* Codes which allows error detection & correction are called error detecting & correcting codes.

Parity Bit :-

* A parity bit is used for the purpose of detecting errors during transmission of binary information.
* It is an extra bit included with the binary message to make the number of 1's either odd or even.
* The message including parity bit is transmitted & checked at the receiving end for errors.
* If an error is detected the receiver request the transmitter to retransmit the Message again.
* The circuit that generates the parity bit in the transmitter is called parity generator.
* The circuit that checks the parity in the receiver is called parity checker.
* The parity can be even parity or odd parity
* In even parity the added parity bit will make the total numbers of 1's an even amount.
* In odd parity the added parity bit will make the total number of 1's an odd amount.

3 bit message	Message with odd parity (Parity bit ↓ | msg bit)	Message with even parity
0 0 0	1 | 0 0 0	0 0 0 0
0 0 1	0 0 0 1	1 0 0 1
0 1 0	0 0 1 0	1 0 1 0
0 1 1	1 0 1 1	0 0 1 1
1 0 0	0 1 0 0	1 1 0 0
1 0 1	1 1 0 1	0 1 0 1
1 1 0	1 1 1 0	0 1 1 0
1 1 1	0 1 1 1	1 1 1 1

Example:-

Write the message 0111001 with an even parity. Place the parity bit in the MSB.

Added parity bit [0] 0111001 (The message contains even number of 1's so the parity bit is 0)

Example:-

The received code is 1000 0001 check whether code is correctly received or not if the parity is odd.

10000001 – (The Message contains even number of 1's, but the parity used is odd parity – Therefore the code is <u>not</u> received <u>correctly</u>)

* The parity error detection system just detects any odd number of error – It cannot detect even number of errors.

Block parity :-

* Several binary words are transmitted & received continuously. It can be regarded as block of data having rows & columns.

for example :-

→ Consider a four eight bit word (4 × 8 block)

→ Parity is assigned to both rows & columns. The scheme is known as Block parity.

→ possible to correct single error & detect any two errors in the word.

Data	Data	
1011	0101	1
0101	0101	0
1100	1101	1
1001	1011	1
1011	0110	Even Parity. 1 ↑ Parity columns.

Transmitted Data

1000	0101	1
0101	0101	0
1100	1101	1
1001	1011	1
10[1][1]	0110	1

↑ ↑ Parity error – Two errors have occured in 3rd & 4th columns.

1011	0101	1
01(1)1	0101	[0] ← Parity error
1100	1101	1
1001	1011	1
10[1]1	0110	↑ parity error.

Received Data.

The error bit is at 2nd row – 3rd column.

* The parity bit checks all bit locations having 1's in the same location in the binary location numbers.

* Therefore P_1 checks for 3, 5 and 7 ---- and assigns P_1 according to even or odd parity.

Bit designation.	<------>	D_5	P_4	D_3	P_2	P_1
Bit location	<------>	5	4	3	2	1
Binary location number	<------	101	100	011	010	001
Information bit (D_n)	<------					
Parity bit (P_n)	<----					

Assignment of P_2 :-

* P_2 bit has '1' in the middle bit of the binary location num
* P_2 checks for bits having 1 in the middle bit
* Parity bit P_2 checks bit location 2, 3, 6 & 7 ------

& assigns P_2 according to even or odd parity.

Assignment of P_4 :-

* P_4 has '1' in the Left most bit
* P_4 checks for bits having '1' in the LSB
* Parity bit P_2 checks bit location 5, 6, 7 --- &

assigns P_4 according to even or odd parity.

Hamming code :-

* R.W Hamming developed a system that provided a way to add one or more parity bits to a data character in order to detect and correct errors.

Step : 1 Number of parity bits.

→ Assign the number of parity bits depending on the number of Information bits.

→ The no. of parity bits is determined by.

$2^P \geq x + P + 1$ where x - no. of information bits
P - no. of parity bit.

→ The value of p is determined by trial & error Method.

Step : 2 Location of parity bits in the code.

→ The bits are designated from the right most bit as bit 1, bit 2, bit 3 and so on.

$\cdots\cdots\cdots Bit_4\ Bit_3\ Bit_2\ Bit_1$

→ The parity bits are located in the positions corresponding to $2^0, 2^1, 2^2, 2^3 \cdots\cdots = [1, 2, 4, 8 \cdots\cdots]$

$\cdots\cdots\cdots D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$

P_n → designation of parity bit ; D_n → designation of Information bit

Step : 3 Assigning values of parity bit.

→ Write the binary number of each decimal location number.

Assignment of P_1 :-

* The binary location number of parity bit P_1 has a '1' for its right most digit.

Problem:

Encode the binary word 1011 into seven bit even parity Hamming code.

Solution:-

Step: 1 find the number of parity bits required.

Let P = 2 then

$2^P \geq x + P + 1$

$2^2 \geq 4 + 2 + 1$

$4 \geq 7$ ✗

Condition not satisfied.

So Let P = 3 then

$2^P \geq x + P + 1$

$2^3 \geq 4 + 3 + 1$

$8 \geq 8$

Condition satisfied.

So $\boxed{P = 3}$ Total length of encoded Message = x + P

= 4 + 3

= 7 //.

Step: 2 construct the bit location table.

Bit designation	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit location	7	6	5	4	3	2	1
Binary location number	111	110	101	100	011	010	001
Information bit	1	0	1		1		
Parity bits				0		0	1

Step: 3

Assign P_1 → check for location 3, 5, 7 → The location have 3 1's so for even parity $P_1 = 1$

Assign P_2 → check for location 3, 6, 7 → The location have 2 1's so for even parity $P_2 = 0$

Assign P_4 → check for location 5, 6, 7 → The location have 2 1's for even parity $P_4 = 0$

Step: 4

$\boxed{\text{The encoded Hamming code} = 1010101}$

Problem:- Determine the single error correcting code for the information code 10111 for odd parity.

Solution:-

Step: 1 find the number of parity bits.

Let P = 3 then

$2^P \geq x + P + 1$

$2^3 \geq 5 + 3 + 1$

$8 \geq 9$ ✗

condition not satisfied.

So let P = 4 then

$2^P \geq x + P + 1$

$2^4 \geq 5 + 4 + 1$

$16 \geq 9$

condition satisfied

So $\boxed{P = 4}$.

Total length of encoded message

$= x + P$

$= 5 + 4 = 9$ ✓.

Step: 2 Construct the bit location table.

Bit designation	D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit location	9	8	7	6	5	4	3	2	1
Binary location no	1001	1000	0111	0110	0101	0100	0011	0010	0001
Information bit	1		0	1	1		1		
Parity bits.		0				1		1	0

Step: 3

Assign P_1 – check location 3, 5, 7, 9 → The location have 3 1's ✗
so odd parity $P_1 = 0$

Assign P_2 – check location 3, 6, 7 → The location have 2 1's ✗
for odd parity $P_2 = 1$

Assign P_4 – check location 5, 6, 7 → The location have 2 1's ✗
for odd parity $P_4 = 1$

Assign P_8 – check for location 9 → The location have 1 1's
for odd parity $P_8 = 0$

Step: 4 **The encoded Message is 1 0 0 1 1 1 1 1 0**

Detecting & correcting an error:-

* Each parity bit, along with its corresponding group of bits must be checked for proper parity.
* The correct result of individual parity check is marked by 0
* The wrong result is marked by 1
* After all parity checks, binary word is formed taking result bit for P_1 as LSB - The word gives bit location where error was occured.
* If word has all bits 0 there is no error in the Hamming code.

Problem:- Assume that the even parity Hamming code. (0110011) is transmitted & that 0100011 is received. The receiver does not know what was transmitted. Determine bit location where error has occured using received code.

Solution:-

Step: 1 construct the bit location table.

Bit designation	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit location	7	6	5	4	3	2	1
Bit location no.	111	110	101	100	011	010	001
Received code.	0	1	0	0	0	1	1

Step: 2 check for parity bits.

for P_1 : P_1 checks location 1, 3, 5, & 7

1 0 0 0

* Binary number → 1

Since the given parity is even & there is one 1 in the group.

∴ Parity check for even parity is wrong ------ 1 (LSB)

for P_2 : P_2 checks location 2, 3, 6 & 7

1 0 1 0

Since the given parity is even & parity check for even parity is correct ---------- 0

for P_4 P_4 checks location 4, 5, 6, & 7

0 0 1 0

Parity check for even parity is wrong ------ 1

Step: 3 form binary word from the result.

101 → 5 It means that bit location 5 has an error. It is 0 but it should be 1

∴ The correct code is 0 1 1 0 0 1 1

Problem:- The Hamming code 10110101 is received. Detect if any error has occured & correct it using odd parity.

Solution:-

Step: 1 construct a bit location table.

Bit designation	D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit location.	9	8	7	6	5	4	3	2	1
Bit location no.	1001	1000	0111	0110	0101	0100	0011	0010	0001
Received code.	1	0	1	1	0	1	1	0	1

Step: 2 check for parity bits.

for P_1 - check location 1, 3, 5, 7 & 9

1 1 0 1 1 → 4 1's

Parity check for odd parity is wrong ----- 1 (LSB)

for P_2 - check location 2, 3, 6, 7

0 1 1 1

Parity check for odd parity is correct ----- 0

for P_4 - check location 4, 5, 6, 7

1 0 1 1

Parity check for odd parity is correct ---- 0

for P_8 - check location 8 9

0 1

Parity check for odd parity is correct --- 0

A error had occured. in bit 0001 so change the bit value. Therefore the transmitted message code is 101101100

Assignment :-

Given that a frame with bit sequence 1101011011 is transmitted, it has been received as 1101011010 Determine the method of detecting the error using any one error detecting code.

Arithmetic operation :-

* Arithmetic operations in a computer are done using binary numbers & takes place in arithmetic unit.

* The Arithmetic operations are Addition, Subtraction, multiplication & Division

Addition	Subtraction	multiplication	Division
0 + 0 = 0	0 - 0 = 0	0 x 0 = 0	0 ÷ 1 = 0
0 + 1 = 1	1 - 0 = 1	0 x 1 = 0	1 ÷ 1 = 1
1 + 0 = 1	1 - 1 = 0	1 x 0 = 0	0 ÷ 0 = not allowed
1 + 1 = 10 (↑ carry)	10 - 1 = 1	1 x 1 = 1	1 ÷ 0 = not allowed.

Binary Addition :- Two binary numbers can be added in same way as decimal numbers are added → start adding from LSB.

Problem :- Add 1010 and 1111

```
  (1)(1)
   1010        10
  +1111       +15
  -----       ---
  11001       25
```

Ans : 11001

Binary Subtraction :-

Problem : Subtract 1001 & 1101

```
   1101        13
  -1001        -9
  -----       ---
   0100        4
```

Ans : 0100

Problem : Subtract 1001 & 0111

```
   0 10
   1001        9
  -0111       -7
  -----       ---
   0010        2
```

Ans : 0010

Binary multiplication :-

Problem: multiply the binary numbers.

(i) 1011 & 1101 (ii) 100110 & 1001

(iii) 1.01 & 10.1

Solution :-

(i) 1011 & 1101

```
  1011
  1101
------
  1011
 0000
1011
1011
--------
10001111
--------
```

(ii) 100110 & 1001

```
  100110
    1001
----------
  100110
 000000
0000000
100110
----------
1010110
----------
```

(iii) 1.01 x 10.1

```
   101
    101
-------
   101
  000
 101
-------
 11.001
-------
```

Binary division :-

Problem : Divide the following

(i) 11001 ÷ 101 (ii) 11101 ÷ 1100

(i) 11001 ÷ 101

```
        101
    ------
101 | 11001
      101
    ------
       101
       101
    ------
         000
```

11001 ÷ 101 = 101

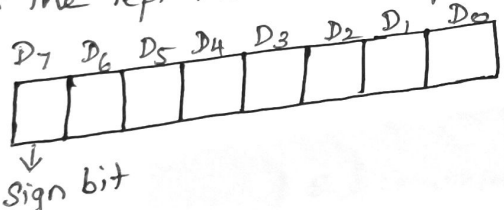
(ii) 11101 ÷ 1100

```
        10.011
    ------------
1100 ) 11101
       1100
       -----
        10100
         1100
        ------
        10000
         1100
        ------
         0100
        ------
```

11101 ÷ 1100 = 10.011

Signed Binary Numbers :-

* + Sign → positive number & − sign → negative number.
* Due to H/w limitations both +ve & −ve numbers are represented in binary numbers.
* The left most bit represents the sign of the number



Sign bit 0 → no. is positive
1 → no. is negative.

* unsigned 8-bit binary number ranges from 0 to 255
* Signed 8-bit binary number ranges from +127 to −128

Complements :-

* 1's complement * 2's complement * 9's complement.
* 10's complement * 15's complement * 16's complement.
* In digital computers to simplify the subtraction operation complements are used.
* There are two types of complement.
  → Radix complement − r complement
  → diminished Radix − r−1 complement.

* Binary number → base 2 → 2's complement - radix
  (2-1) → 1's complement - diminished.
* decimal number → base 10 → 10's complement
  (10-1) → 9's complement.
* hexadecimal → base 16 → 16's complement
  (16-1) → 15's complement.

1's complement :-

* can be obtained by changing all 0's to 1 and all 1's to 0

example :- find 1's complement of 0110

0110 → 1001

2's complement :-

* can be obtained by adding 1 to its 1's complement.

example :- find 2's complement of 1010

1010 → 1's complement → 0101

+ 1

2's complement → 0110

9's complement :-

* obtained by subtracting each digit in the number from "9"

example :- find the 9's complement of 4397

```
  9999
- 4397
------
  5602
```

4397 → 9's complement → 5602.

10's complement :-

* 10's complement is equal to its 9's complement + 1

example:- find the 10's complement of 739

9's complement → 999
739
260.

Add '1' with 9's complement

260 + 1 = 261

739 → 10's complement → 261

15's complement :-

* obtained by subtracting each digit from 15

example:- find 15's complement of A9B.

15 15 15
A 9 B
5 6 4

A9B → 15's complement → 564

16's complement :-

* Obtained by adding 1 to the LSB of 15's complement.

example:- find the 16 complement of A9C

15's complement.

15 15 15
A 9 C.
5 6 3.

16's complement.

563 +
1
564.

* A9C → 16's complement → 564.

BCD Arithmetic Simplification :-

Binary code decimal (BCD)

* BCD is a numeric code in which each digit of a decimal number is represented by a separate group of bits.
* The most common BCD code is 8-4-2-1 BCD.
* It is represented by 4-bit binary number - BCD.
* The numbers 8421 indicate the binary weights of the four bits.
* In BCD binary number from 0000 to 1001 is only used [0-9].

example :- 7 is represented as 0111

12 is represented as $\underset{1}{0001}$ $\underset{2}{0010}$

Decimal	8421 BCD	4221 BCD	5421 BCD
0	0000	0000	0000
1	0001	0001	0001
2	0010	0010	0010
3	0011	0011	0011
4	0100	0110	0100
5	0101	0111	0101
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100
10	0001 0000		
11	0001 0001		

Valid BCD.

BCD Addition :-

Step : 1 Add the two numbers using the rules for binary addition

Step : 2 If a four bit sum is equal to or less than 9
It is a valid BCD number.

Step : 3 If a four bit sum is greater than 9 (or) if a carry out the group is generated. It is a invalid result
Add 6 (0110) to the four bit sum in order to skip the six invalid states & return to BCD.

Problem :- Add the following BCD numbers.

(i) 1001 and 0100

```
  1001
  0100
  ----
  1101   → Invalid BCD.
  0110   Add 6.
  ----
1 0011   Valid BCD.
-----
1  3
```

(ii) 00011001 & 00010100

```
  00011001
  00010100
  --------
  00101101
     0110
  --------
  00110011
  ----  ----
   3     3.
```

Valid BCD.

BCD Subtraction :-

* Addition of 9's complement of the subtrahend to minuend.

Problem :- subtract the following using 9's complement Method.

(a) 649 - 387

(b) 891 - 786.

Solution:-

(a) 9's complement of subtrahent 387 → 999 − 387 = 612

Add the result with → 649 + 612 = 1261 → 1 + 261 = 262

649 − 387 = 262

(b) 891 − 786.

9's complement of subtrahent 786 → 999 − 786 = 213

Add the result with → 891 + 213 = 1104 → 1 + 104 = 105

891 − 786 = 105

Non-weighted codes:-

* Excess 3 code - The excess 3 represents a decimal number in a binary form as a number greater than 3

* An excess 3 code is obtained by adding 3 to a decimal number.

Excess 3 code.

Decimal	8421 BCD	Excess 3 code.
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Problem :- perform the addition is excess - 3 code.

103 + 287.

Solution :-

1	0	3
0001	0000	0011
0011	0011	0011
0100	0011	0110
0101	1011	1010
1001	1111	0000
		0011
1001	1111	0011
0011	0011	
0110	1100	0011

Gray code :-

* The Gray code belongs to a code called minimum change codes.
* Only one bit in the group changes when moving from one step to the next.
* It is also called unit distance code:

Conversion of binary number to gray code :-

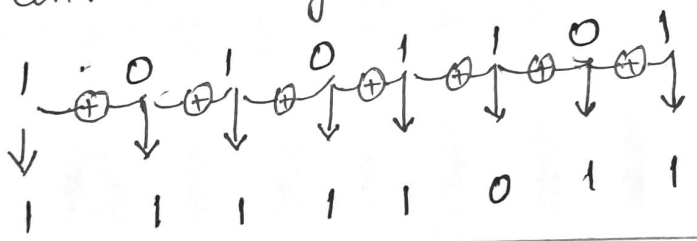
Step: 1 MSB of gray code is the same as the first bit of binary number.

Step: 2 The second bit of gray code equals the Exclusive OR of first & second bit of binary number.

Step: 3 The third bit of gray code equals the exclusive OR of second & third bit of binary number & soon.

Problem: Convert binary 10101101 to its gray code.

Solution:

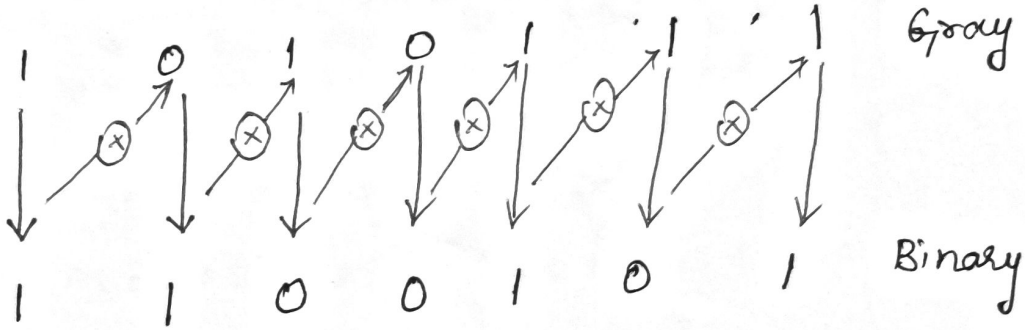


Conversion of gray code to binary number :-

Step: 1 MSB is same as that of gray code.

Each binary bit is the exclusive OR of corresponding gray code & the previous binary bit.

Problem :- convert 1010111 into binary.



Standard Representation for logical Functions :-

* Logical functions are expressed in terms of logical variables.
* The logical functions can be represented in two forms
  - → Sum of products form
  - → product of sums form.

Sum of Products (SOP) :-

* The form consists of two or more AND terms that are ORed together.

(eg) $Y = AB + AB\bar{C} + \bar{B}$.

Minterm :- $\leq m$

* product term containing all the variables → standard sum of products form (or) canonical sum of product form → minterm.

A	B	C	minterm	
0	0	0	$\bar{A}\bar{B}\bar{C}$	m_0
0	0	1	$\bar{A}\bar{B}C$	m_1
0	1	0	$\bar{A}B\bar{C}$	m_2
0	1	1	$\bar{A}BC$	m_3
1	0	0	$A\bar{B}\bar{C}$	m_4
1	0	1	$A\bar{B}C$	m_5
1	1	0	$AB\bar{C}$	m_6
1	1	1	ABC	m_7

* The minterm can be stated as

$Y(A,B,C) = \leq m(0,5,6)$

$= m_0 + m_5 + m_6$.

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

* To get the standard sop form.

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

* missing terms C in 1st term.
  AC in 2nd term.

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

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

$= \underset{m_7}{ABC} + \underset{m_6}{AB\bar{C}} + \underset{m_5}{A\bar{B}C} + \underset{m_1}{\bar{A}\bar{B}C} + \underset{m_4}{A\bar{B}\bar{C}} + \underset{m_0}{\bar{A}\bar{B}\bar{C}}$

$Y(A,B,C) = \Sigma m\,(0,1,4,5,6,7)$.

Product of Sum (POS) :- ΠM

* The form consists of OR terms which are AND ed together.

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

A	B	C	Maxterm.	
0	0	0	$A+B+C$	M_0
0	0	1	$A+B+\bar{C}$	M_1
0	1	0	$A+\bar{B}+C$	M_2
0	1	1	$A+\bar{B}+\bar{C}$	M_3
1	0	0	$\bar{A}+B+C$	M_4
1	0	1	$\bar{A}+B+\bar{C}$	M_5
1	1	0	$\bar{A}+\bar{B}+C$	M_6
1	1	1	$\bar{A}+\bar{B}+\bar{C}$	M_7

* The maxterm can be stated as

$Y(A,B,C) = \Pi M\,(0,2,4)$

$= M_0 * M_2 * M_4$.

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

To get the standard POS form.

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

$\Rightarrow (A+B+0)(A+B+\bar{C})(0+\bar{B}+0)$.

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

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

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

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

$\Rightarrow M_0 \cdot M_1 \cdot M_2 \cdot M_3 \cdot M_6 \cdot M_7$

$Y(A,B,C) = \Pi M\,(0,1,2,3,6,7)$

Problem :-

Convert the following Boolean function into standard SOP and express it in terms of minterms.

$$Y(A,B,C) = AB + A\bar{C} + BC.$$

Solution :-

* C term is missing in 1st term. add $C+\bar{C}$
  B term " " " 2nd term add $B+\bar{B}$
  A term " " " 3rd term. add $A+\bar{A}$

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

$\Rightarrow \underline{ABC} + \underline{AB\bar{C}} + \underline{AB\bar{C}} + A\bar{B}\bar{C} + \underline{ABC} + \bar{A}BC$

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

$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$

$m_7 \quad m_6 \quad m_4 \quad m_3.$

$$Y(A,B,C) = \Sigma m(3,4,6,7)$$

Problem :-

Convert the following Boolean function into standard pos and express in terms of max terms.

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

Solution :-

* add $C\bar{C}$ in 1st term, $A\bar{A}$ in 2nd term & $B\bar{B}$ in 3rd term.

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

$\Rightarrow \underline{(A+B+C)}(A+B+\bar{C})(A+B+\bar{C})(\bar{A}+B+\bar{C})\underline{(A+B+C)}(A+\bar{B}+C)$

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

$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$

$M_0 \quad M_1 \quad M_5 \quad M_2$

$$Y(A,B,C) = \Pi M(0,1,2,5)$$

Simplification of Boolean Expression :-

* To reduce the requirements of hardware, it is necessary to simplify the Boolean expression.

* Boolean expressions can be simplified using:
  - → Algebraic simplification
  - → Karnaugh map method. (K-map)
  - → Quine - Mccluskey method.

Algebraic Simplification :-

Problem:-
Simplify the three variable expression using Boolean algebra.

$Y(A,B,C) = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C} + ABC$

Solution:-

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

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

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

$= \bar{A} + A$

$= 1.$

Drawback of algebraic Simplification :-

* No easy way to tell whether the simplified expression is in its simplest form or it can be simplified further.

Karnaugh map Simplification :-

* K-map contains cell.
* The number of cells in a K-map depends upon the number of variables

2 variables

$2^2 = 4$ cells.

A \ B	0	1
0	00	01
1	10	11

3 Variables

$2^3 = 8$ cells.

A \ BC	00	01	11	10
0	000 m_0	001 m_1	011 m_3	010 m_2
1	100 m_4	101 m_5	111 m_7	110 m_6

4 variables.

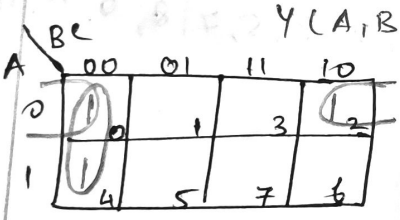
$2^4 = 16$ cells.

AB \ CD	00	01	11	10
00	0000 m_0	0001 m_1	0011 m_3	0010 m_2
01	0100 m_4	0101 m_5	0111 m_7	0110 m_6
11	1100 m_{12}	1101 m_{13}	1111 m_{15}	1110 m_{14}
10	1000 m_8	1001 m_9	1011 m_{11}	1010 m_{10}

Problem:-

Simplify the following expression using k-map:

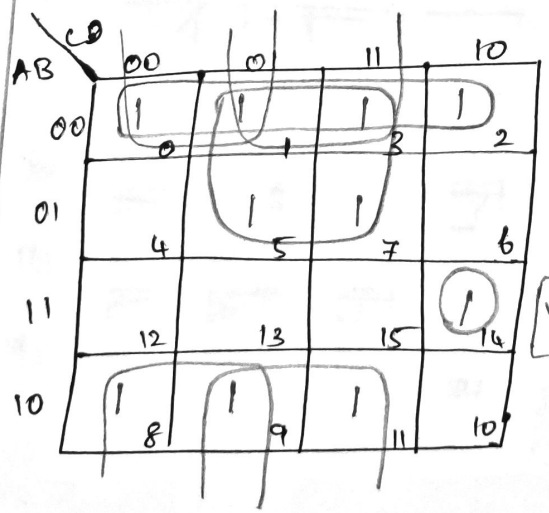
$$Y(A,B,C) = \Sigma m(0,2,4)$$



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

Problem:- Minimize the function using k-map :-

$$Y(A,B,C,D) = \Sigma m(0,1,2,3,5,7,8,9,11,14)$$

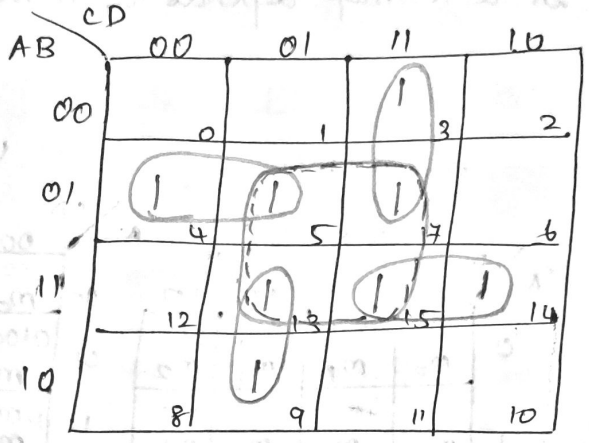


$$Y(A,B,C,D) = \bar{A}\bar{B} + \bar{A}D + B\bar{C} + \bar{B}D + ABC\bar{D}$$

$$\boxed{Y(A,B,C,D) = \bar{A}\bar{B} + \bar{A}D + \bar{B}\bar{C} + \bar{B}D + ABC\bar{D}}$$

Problem:-

Simplify the expression $Y = \Sigma m\,(3,4,5,7,9,13,14,15)$ using K map method :-

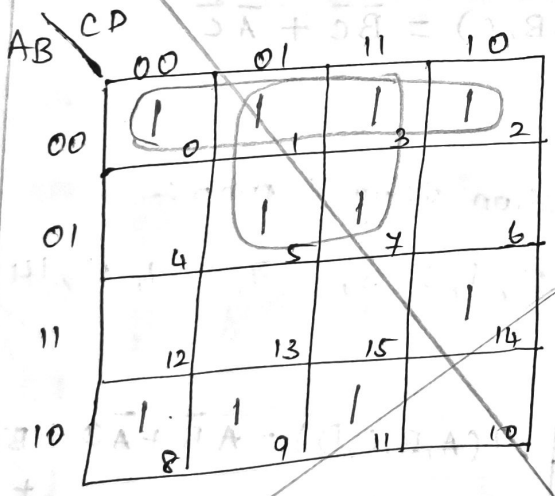


$$Y = \overline{BD} + \bar{A}B\bar{C} + \bar{A}\bar{C}D + A\bar{B}C + ABC$$

Redundant

Problem:- Simplify the logic function using K-map.

$$Y(A,B,C,D) = \Sigma m\,(0,1,2,3,5,7,8,9,11,14)$$



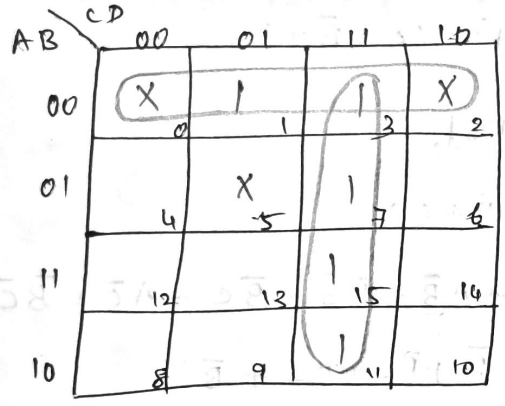
Don't care Condition :-

* The output level is not defined it can be either HIGH or LOW.
* These output levels are indicated by 'x' (cross) & called as don't care condition.

Problem :- Simplify the Boolean expressions.

$$Y = \Sigma m(1, 3, 7, 11, 15) + d(0, 2, 5)$$

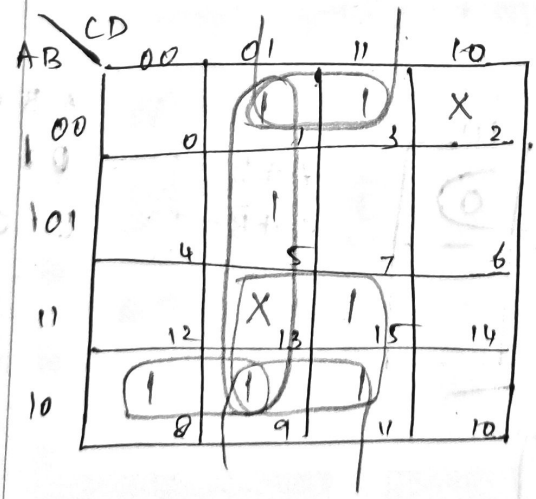
Solution :-



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

Problem :- Minimize the function.

$$Y(A, B, C, D) = \Sigma m(1, 3, 5, 6, 9, 11, 15) + d(2, 13)$$



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

Simplification of Product of Sums :-

4 Variable :-

AB \ CD	00	01	11	10
00	$A+B+C+D$	$A+B+C+\bar{D}$	$A+B+\bar{C}+\bar{D}$	$A+B+\bar{C}+D$
01	$A+\bar{B}+C+D$	$A+\bar{B}+C+\bar{D}$	$A+\bar{B}+\bar{C}+\bar{D}$	$A+\bar{B}+\bar{C}+D$
11	$\bar{A}+\bar{B}+C+D$	$\bar{A}+\bar{B}+C+\bar{D}$	$\bar{A}+\bar{B}+\bar{C}+\bar{D}$	$\bar{A}+\bar{B}+\bar{C}+D$
10	$\bar{A}+B+C+D$	$\bar{A}+B+C+\bar{D}$	$\bar{A}+B+\bar{C}+\bar{D}$	$\bar{A}+B+\bar{C}+D$

Problem :-

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

Solution using Algebraic Method.

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

$$= AA + A\bar{B} + AC + A\bar{B} + \bar{B}\bar{B} + \bar{B}C + A\bar{C} + \bar{B}\bar{C} + C\bar{C} \to 0$$

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

$$= A(1+\bar{B}) + A(C+\bar{C}) + \bar{B} + \bar{B}(C+\bar{C})$$

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

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

Solution using K-map POS.

A \ BC	00	01	11	10
0			0	0
1				

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

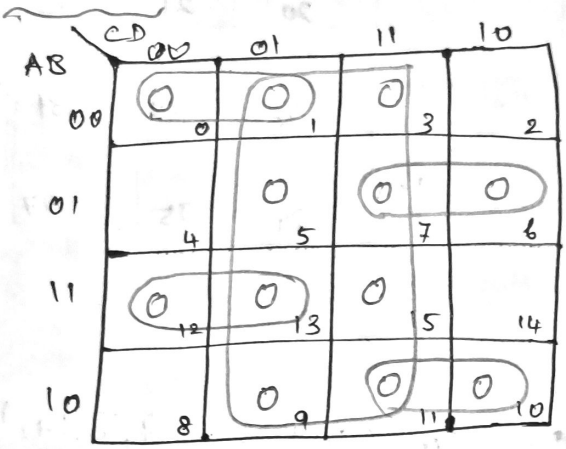
	A	B	C
$A+\bar{B}+\bar{C}$	0	1	1
$A+\bar{B}+C$	0	1	0

Problem :-

$Y(A,B,C,D) = \Pi M\ (0,1,3,5,6,7,9,10,11,12,13,15)$

Simplify using k-map.

Solution :-

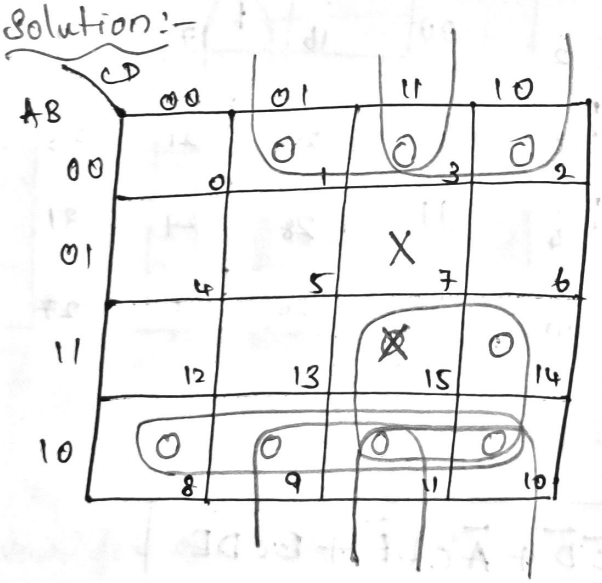


$Y(A,B,C,D) = \bar{D}(A+B+C)(A+\bar{B}+\bar{C})(\bar{A}+\bar{B}+C)(\bar{A}+B+\bar{C})$

Problem :-

Simplify $Y(A,B,C,D) = \Pi M\ (1,2,3,8,9,10,11,14) \cdot d(7,15)$

Solution :-



$Y(A,B,C,D)$

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

Five Variable :-

A = 0

BC \ DE	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

A = 1

BC \ DE	00	01	11	10
00	16	17	19	18
01	20	21	23	22
11	28	29	31	30
10	24	25	27	26

Problem :-

Simplify $Y(A,B,C,D,E) = \sum m(0,1,5,6,9,13,14,17,21,22,25,29)$.

Solution :-

A = 0

BC \ DE	00	01	11	10
00	1 0	1 1	3	2
01	4	1 5	7	1 6
11	12	1 13	15	1 14
10	8	1 9	11	10

A = 1

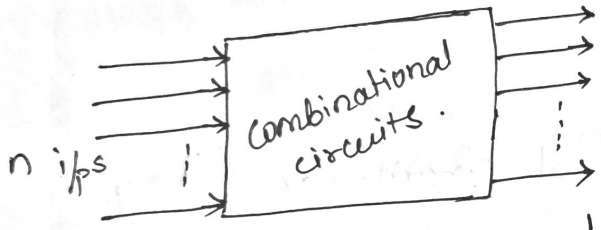
BC \ DE	00	01	11	10
00	16	1 17	19	18
01	20	1 21	23	1 22
11	28	1 29	31	30
10	24	1 25	27	26

G_3, G_4, G_2, G_1

$$Y(A,B,C,D) = \bar{D}E + \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}CD\bar{E} + \bar{B}CD\bar{E}$$

Combinational Circuits :-

* Logic circuits for digital systems may be combinational or sequential.
* Combinational circuit consists of an interconnection of logic gates whose o/p's at any time are determined from only the present combination of inputs.
* Combinational circuit performs an operation that can be specified logically by a set of Boolean function.



* The n i/p binary variables come from an external source. The m output variable are produced by the Internal combinational logic circuit and go to an external destination.
* for n input variables there are 2^n possible combinations of the binary inputs.
* for each possible Input combination, there is one possible value for each output variable.
* Combinational circuits can be specified with a truth table that lists the o/p values for each combinational of i/p variables.
* The Combinational circuits also can be described by m - Boolean function, one for each input variable.
* Each o/p function is expressed in terms of n i/p variables

Design Procedure :-

* From the specification of the circuit, determine the required no of i/p's & o/p's and assign a symbol for each.
* Derive the truth table that defines the required relationship b/w i/p's and o/p's
* obtain the simplified Boolean function for each o/p as a function of i/p variables
* Draw the logic diagram & verify the correctness of the design.

Analysis Procedure :-

* The diagram of a combinational circuit has logic gates with no feedback path or Memory elements.
* Label all gate outputs that are a function of input variables with arbitrary symbols - with meaningful names. Determine the boolean functions for each gate o/p.
* Label the gates that are a function of i/p variables & previously labeled gates with other arbitrary symbols. find the boolean functions for these gates.
* Repeat the process outlined in step-3 until the o/p's of the circuit are obtained.
* By repeated substitution of previously defined functions. obtain the o/p Boolean function in terms of i/p variables.

Adder.

* The most basic arithmetic operation is the addition of two binary digits.
* The simple addition consists of four possible elementary operations.

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 10$

* The first three operations produce a sum of one digit but when both augend and addend bits are equal to 1 the binary sum consists of two digits.
* The higher significant bit of this result is called a carry
* when the augend & addend numbers contain more significant digits, the carry obtained from the addition of two bits is added to the next higher order pair of significant bits.

Half Adder.

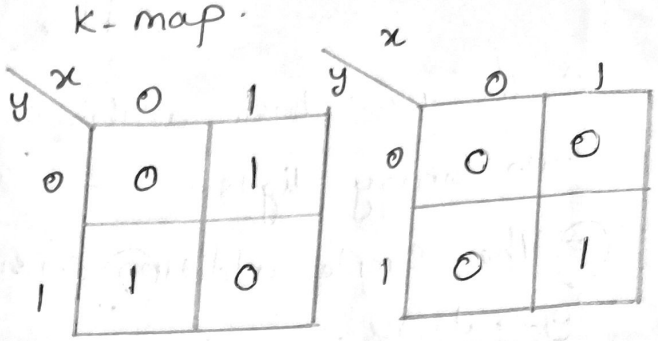
* A combinational circuit that performs the addition of two bits is called a half adder.

* I/p variables → augend & addend
  → x & y
* O/p Variables → Sum & Carry
  → S & C

Truth Table :-

Input		output	
x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

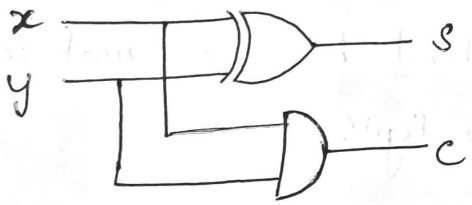
K-map.



$Sum = \bar{x}y + x\bar{y}$ $\quad$ $carry = xy$

$\boxed{Sum = x \oplus y}$ $\quad$ $\boxed{carry = xy}$

Logic diagram

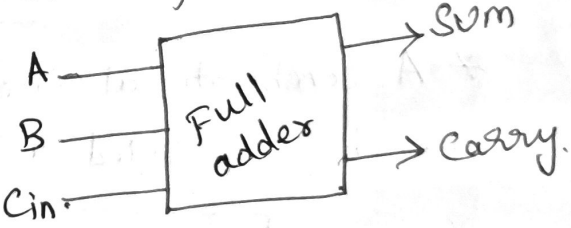


Full Adder.

* The circuit that performs the addition of three bits is a full adder.
* It consists of three inputs and two outputs.

I/p = A, B, Cin.

O/p = Sum, carry.



* Addition of n-bit binary numbers requires the use of full adder as the process of addition proceeds on a bit-by-bit basis right to left. beginning with the least significant bit.

Truth table:- 28, 33, 40, 43, 46, 51, 53
55, 67, 70, 81, 88

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

Sum:

A \ BCin	00	01	11	10
0	0	(1)	0	(1)
1	(1)	0	(1)	0

$Sum = A'B'C_{in} + A'BC_{in}' + AB'C_{in}' + ABC_{in}$.

$\Rightarrow (A'B' + AB)C_{in} + (A'B + AB')C_{in}'$

$= C_{in}(\overline{A \oplus B}) + C_{in}'(A \oplus B)$

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

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

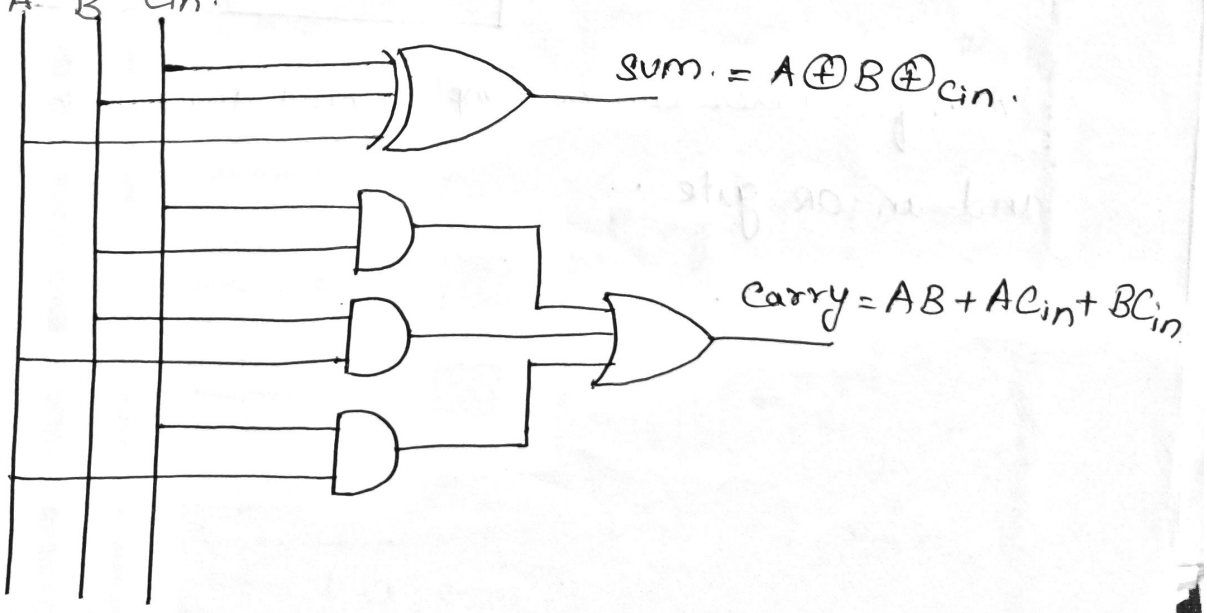
Carry:

A \ BCin	00	01	11	10
0	0	0	(1	0
1	0	(1	1)	(1

$carry = AB + BC_{in} + AC_{in}$

$\boxed{carry = AB + BC_{in} + AC_{in}}$

Logic diagram:-



Implementation of full adder with Half adder :-

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

$$Carry = AB + BC_{in} + AC_{in}.$$

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

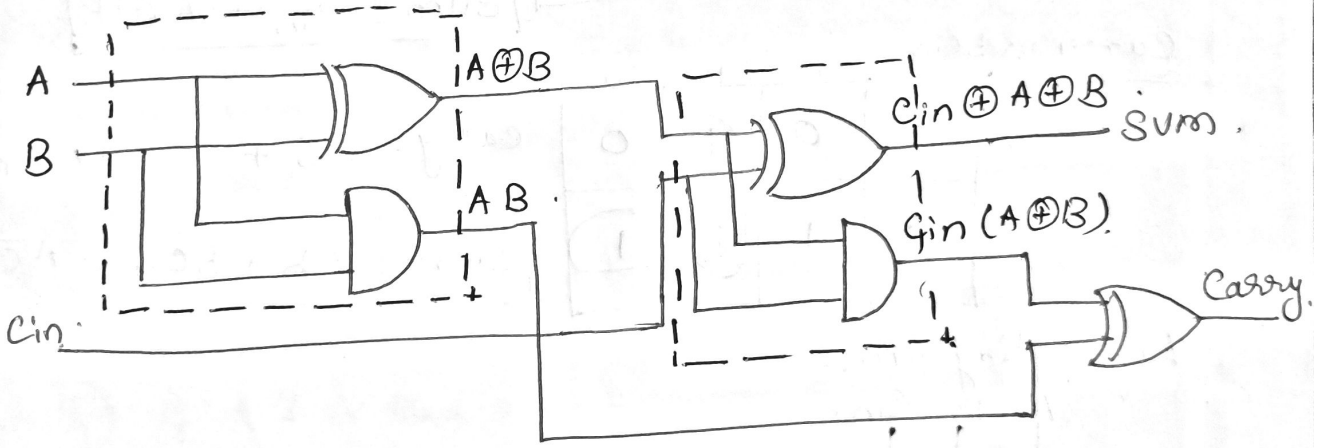
$$= AB + ABC_{in} + \bar{A}BC_{in} + ABC_{in} + A\bar{B}C_{in}.$$

$$= AB + ABC_{in} + \bar{A}BC_{in} + A\bar{B}C_{in}.$$

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

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

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



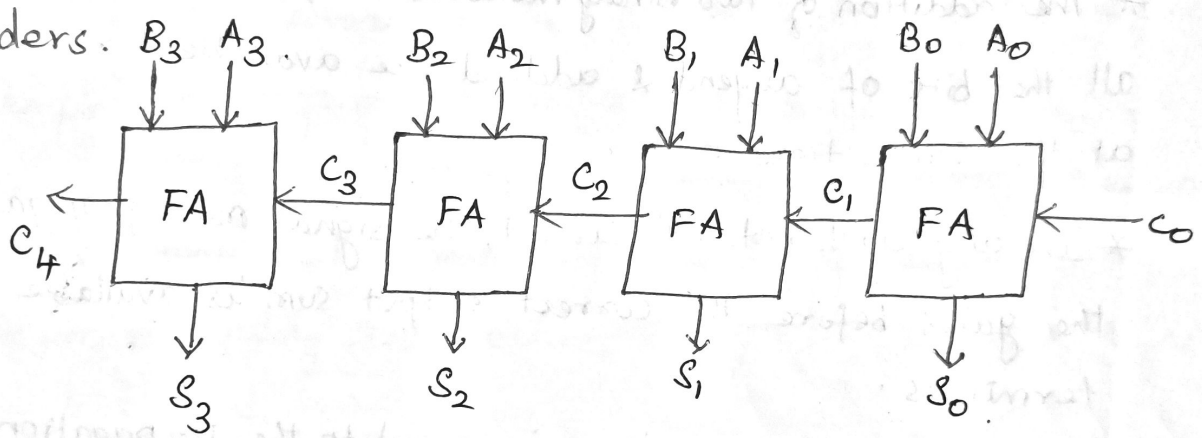
* A full adder can be implemented using two half adders and an OR gate.

Carry lookahead generator :-

* Binary adder is a digital circuit that produces arithmetic sum of two binary numbers.

* Constructed with full adders connected in cascade, with output of carry from each full adder connected to input carry of next full adder.

* Addition of n bit numbers require a chain of n full adders.



* Four bit binary ripple carry adder.

* Augent bits of A & Addend bits of B are designated by subscript numbers from right to left with subscript 0 denoting Least significant bit.

* carrys are connected in a chain through the full adders.

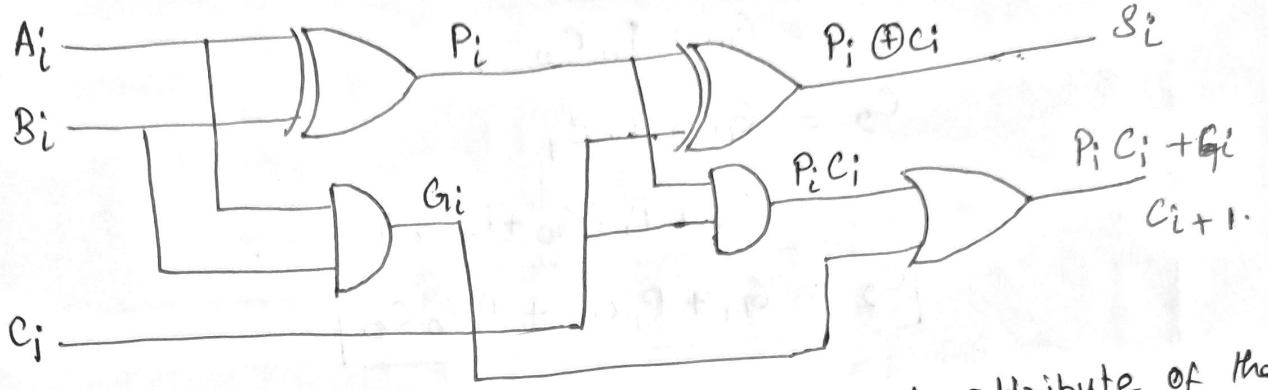
* Consider the two binary numbers A = 1011 and B = 0011

```
      1  1  <- carry
A     1  0  1  1
B     0  0  1  1
-------------------
      1  1  1  0   sum.
```

A	1	0	1	A_i
B	0	0	1	B_i
Carry in.	0	1	1	0 C_i
	1	1	1	0 Sum.
	0	0	1	1 carry.

Carry propagation :-

* The addition of two binary numbers in parallel implies that all the bits of augend & addend are available for computation at the same time.
* In any combinational circuit, the signal must propogate thro' the gates before the correct output sum is available in o/p terminals.
* Total propagation time is equal to the propagation delay of a typical gate, times the number of gate levels in the ckt.
* Consider output S_3 → Inputs A_3 & B_3 are available as soon as Input signals are applied to the adder. However Input carry C_3 does not settle to its final value until C_2 is available from previous stage.
* Illy. C_2 has to wait for C_1 and so on down to C_0.



* The carry propagation time is an important attribute of the adder bcoz it limits the speed with which two numbers are added.
* An solution for reducing the carry propagation delay time is to employ faster gates with reduced delays.
* Another solution is to increase the complexity of the equipment in such a way that carry delay time is reduced.
* The most widely used technique employs the principle of carry look ahead logic.

$$P_i = A_i \oplus B_i$$
$$G_i = A_i B_i$$

The output sum & carry.

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

G_i is called a carry generate & it produces a carry of 1 when both A_i and B_i are 1 regardless of input carry C_i

P_i is called carry propogate. bcoz it determines whether a carry into stage i will propegate into stage i+1

C_0 = input carry.

$$\boxed{C_1 = G_0 + P_0 C_0.}$$

$$C_2 = G_1 + P_1 C_1$$

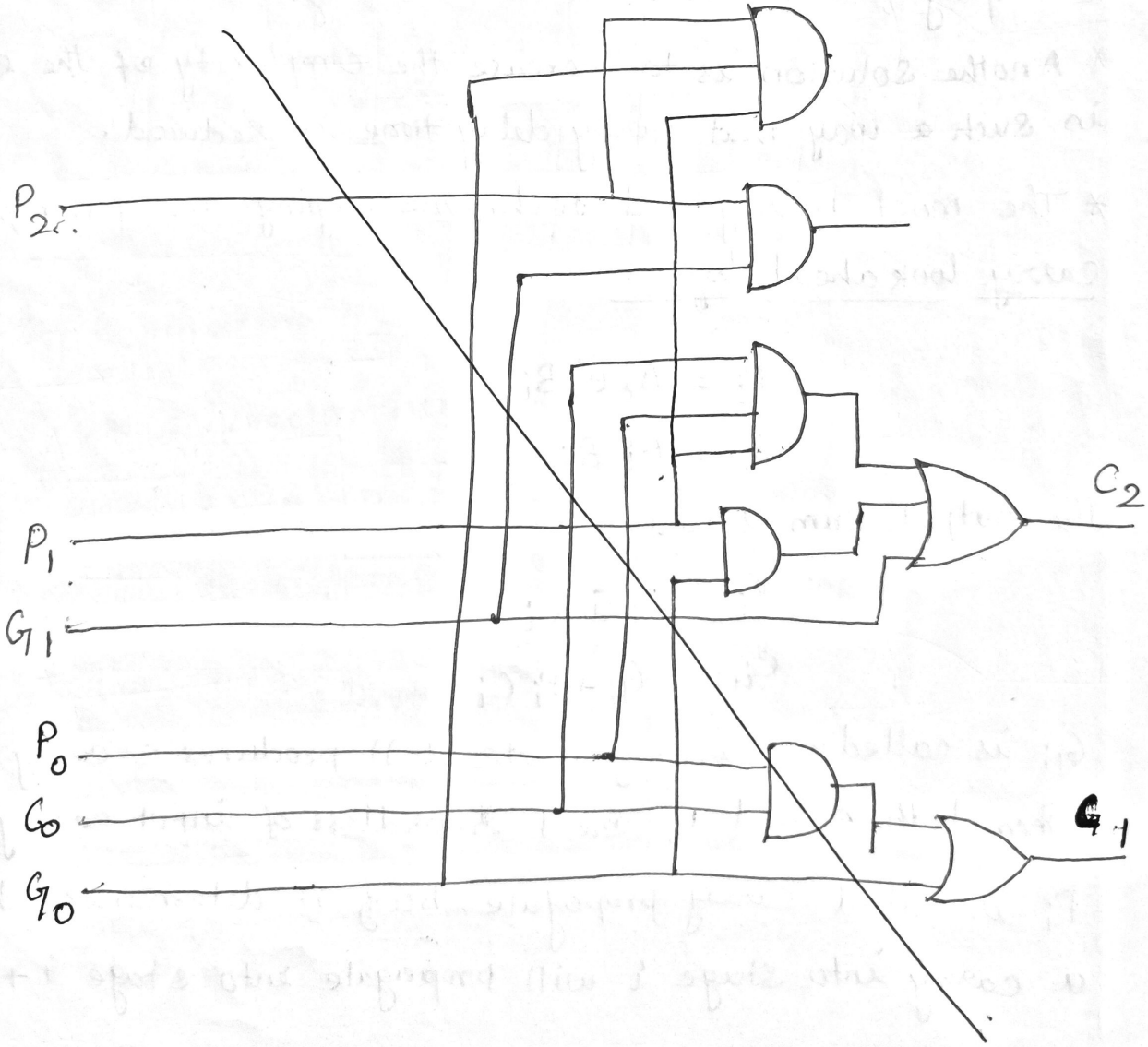
$$= G_1 + P_1 (G_0 + P_0 C_0)$$

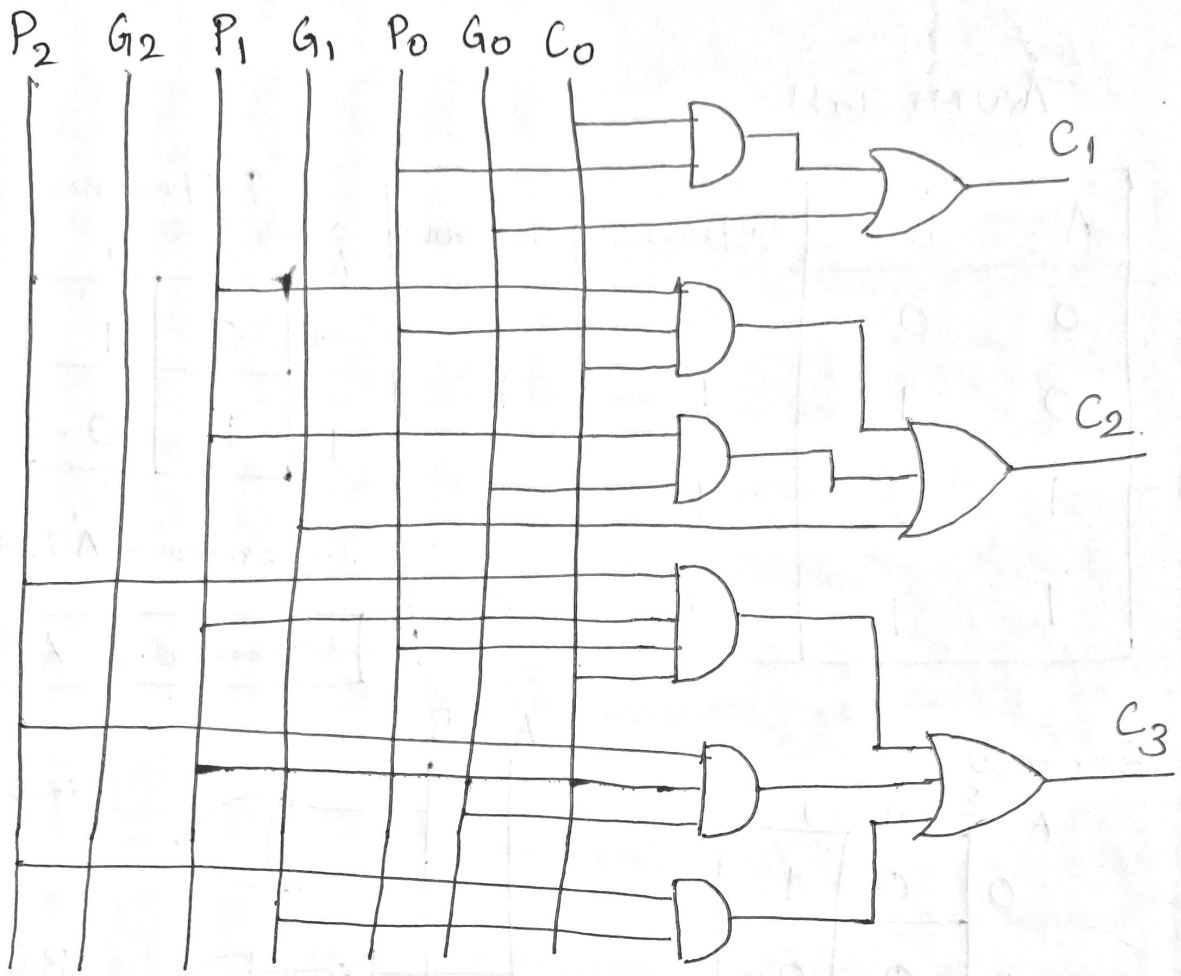
$$\boxed{C_2 = G_1 + P_1 G_0 + P_1 P_0 C_0.}$$

$$C_3 = G_2 + P_2 C_2$$

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

$$\boxed{C_3 = G_2 + G_1 P_2 + P_1 P_2 G_0 + P_0 P_1 P_2 C_0}$$

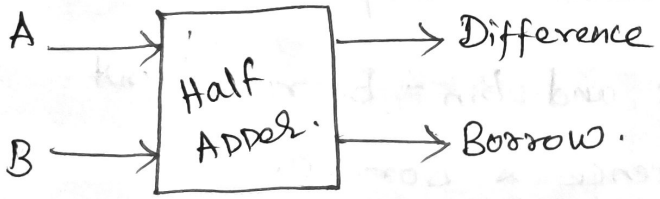




Subtractor :-

Half Subtractor :-

* Half subtractor has two inputs & two outputs.



Inputs – A & B.

Outputs – Borrow & Difference.

* The two inputs A and B form the minuend and the subtrahend.

* The two outputs D is the Difference output and B is the borrow output.

TRUTH Table.

A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Difference.

A \ B	0	1
0	0	1
1	1	0.

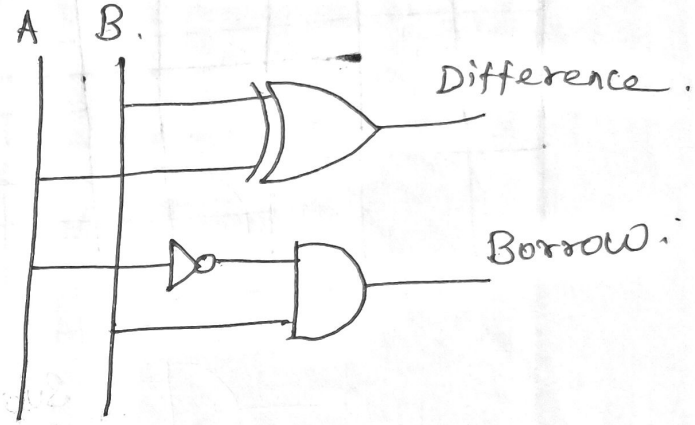
Difference = $A'B + AB'$

Difference = $A \oplus B$.

Borrow.

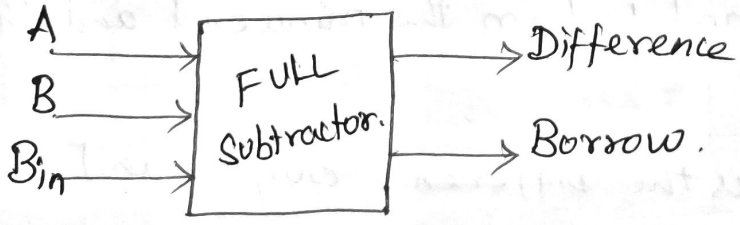
A \ B	0.	1
0	0	1
1	0	0.

Borrow $\neq A'B$.



Full Subtractor :-

* A full subtractor has three inputs and two outputs.
* Inputs = A, B and B_{in} = Borrow input
  Outputs = Difference & Borrow.



Truth table:

A	B	Bin	Difference	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Difference.

A \ BBin	00	01	11	10
0	0	(1)	0	(1)
1	(1)	0	(1)	0

$Difference = A'B'B_{in} + A'BB_{in}' + AB'B_{in}' + ABB_{in}$.

$\Rightarrow B_{in}(A'B' + AB) + B_{in}'(A'B + AB')$

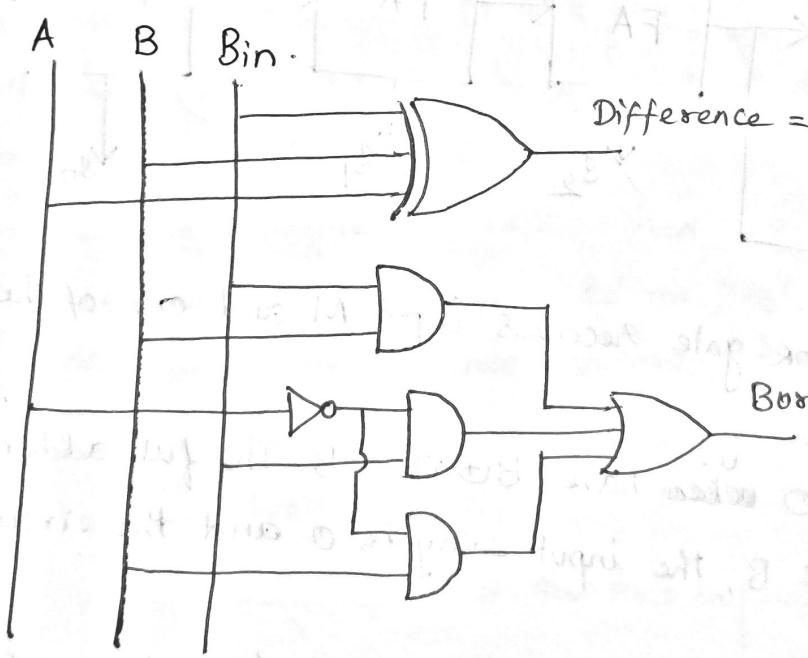
$= B_{in}(\overline{A \oplus B}) + B_{in}'(A \oplus B)$.

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

Borrow.

A \ BBin	00	01	11	10
0	0	(1)	(1)	(1)
1	0	0	(1)	0

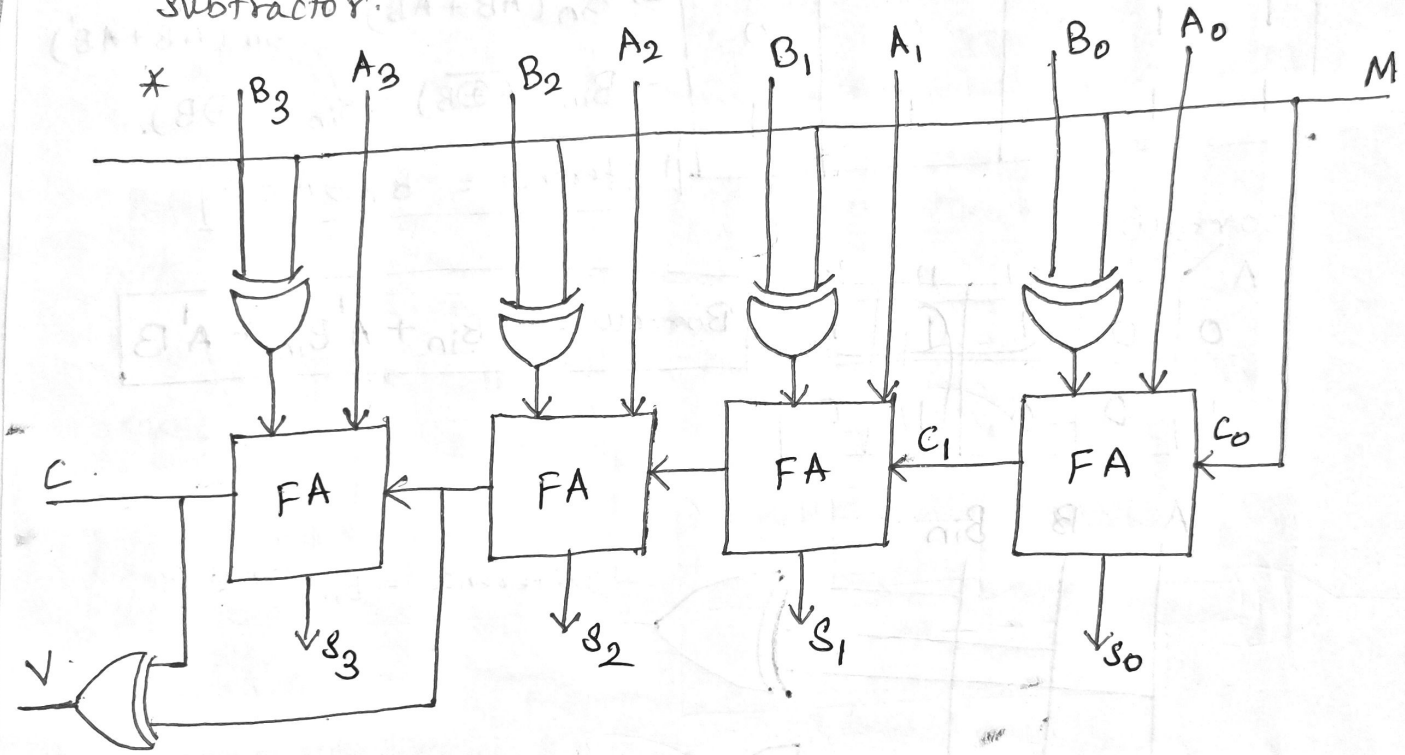
$Borrow = \bar{B}B_{in} + A'B_{in} + A'\bar{B}$



Binary Subtractor:-

* The addition and subtraction operations can be combined into one circuit with one common binary adder by including an exclusive OR gate with each full adder.

* The mode input M controls the operation. when $m=0$ the circuit is an adder. and when $m=1$ the circuit becomes a subtractor.



* Each Ex-OR gate receives input M and one of the inputs of B.

* when $m=0$ we have $B \oplus 0 = B$. The full adders receive the value of B, the input carry is 0 and the circuit performs A plus B.

* when $m=1$ we have $B \oplus 1 = B'$ and $C_0 = 1$. The B. inputs are all complemented and a '1' is added thro' i/p Carry.

* One common hardware circuit handles both types of arithmetic.

Overflow!-

* When two numbers with n digits each are added and the sum is a number occupying n+1 digits, then we say an overflow occurred.
* overflow is a problem in digital computers bcoz the number of bits that hold the number is finite & a result that contains n+1 bits cannot be accommodated by an n-bit word.
* for this reason, many computers detect the occurrence of an overflow.
* The detection of an overflow after the addition of two binary numbers depends on whether the numbers are considered to be signed (or) unsigned.
* when two unsigned numbers are added, an overflow is detected
* In the case of signed numbers - when two signed numbers are added, the sign bit is treated as part of the number & end carry does not indicate an overflow.
* An overflow cannot occur after an addition if one number is positive & other is negative.
* An overflow may occur if the two numbers added are both positive (or) both negative.

* The binary adder - subtractor circuit with outputs C and V.

* If the two binary numbers are considered to be unsigned then the C bit detects a carry after addition (or) a borrow after subtraction.

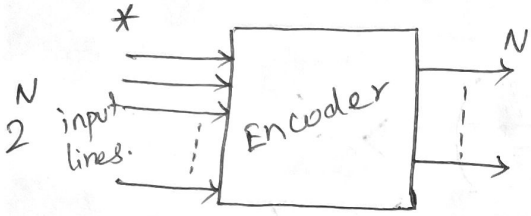
* If the numbers are considered to be signed, then the V bit detects an overflow.

If V = 0 then no overflow occured.

If V = 1 then the result of the operation contains n+1 bits.

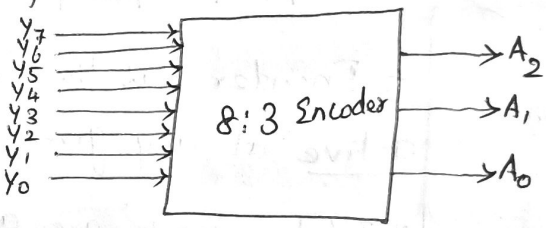
Encoders :-

* An encoder has 2^n input lines and n output lines
* The output lines generate binary code corresponding to the input value.
* An example of the encoder is Octal - binary encoder.



8 : 3 Encoder (Octal to Binary)

* 8:3 Encoder consists of 8 inputs and 3 outputs.
* I/p lines = D_0 to D_7
* O/p lines = x y z
* Truth Table.



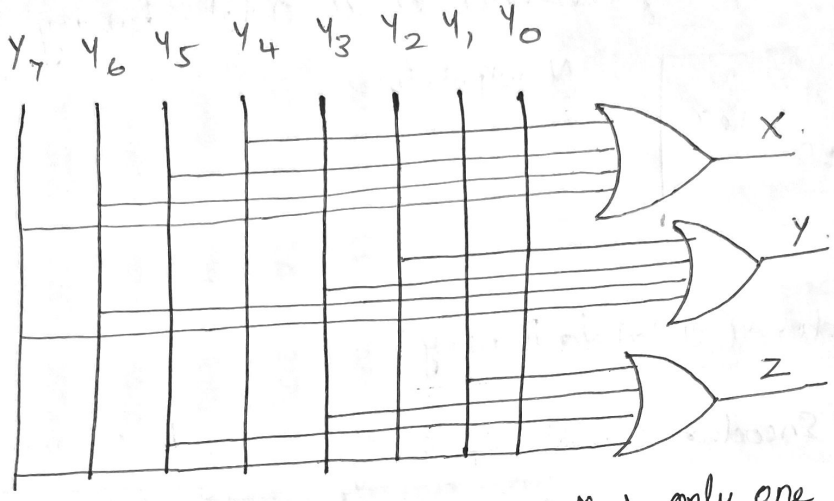
Inputs								outputs.		
Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	x	y	z
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$$X = Y_4 + Y_5 + Y_6 + Y_7$$

$$Y = Y_2 + Y_3 + Y_6 + Y_7$$

$$Z = Y_1 + Y_3 + Y_5 + Y_7$$

* The above function can be implemented using OR gates.



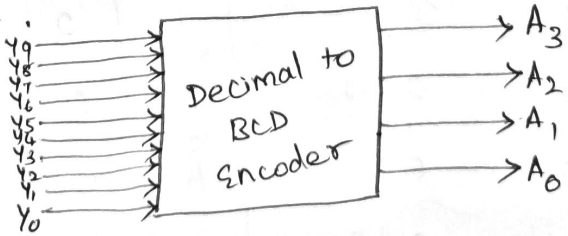
* Encoder has the limitation that only one input can be active at any given time.

* If two inputs are active simultaneously, the output produces a undefined combination.

Decimal to BCD Encoder :-

* Decimal to BCD Encoder consists of 10 Input lines and 4 output lines

* It accepts decoded decimal data as input & encodes it to BCD output.



Truth Table :-

Inputs										outputs			
Y_9	Y_8	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

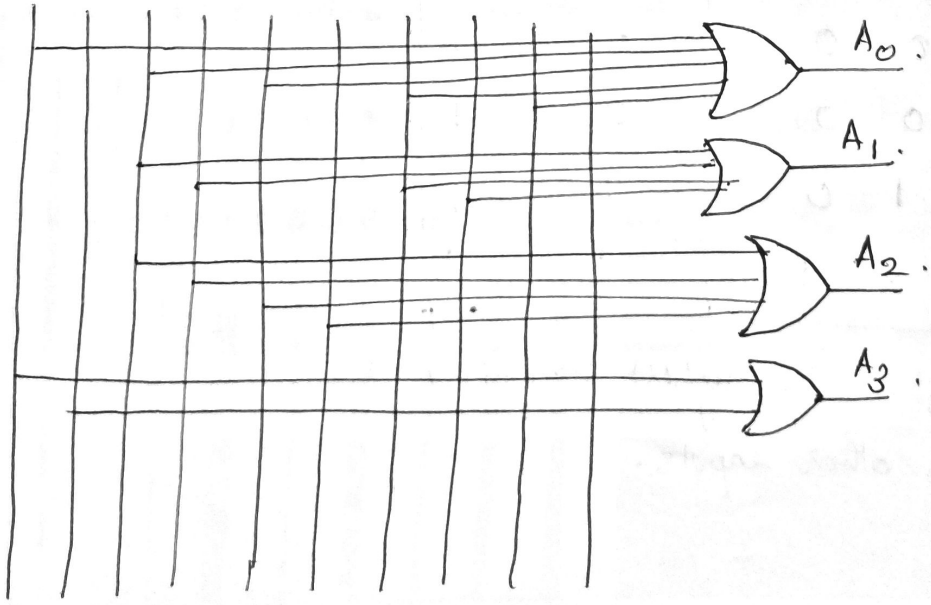
Logical expression: $A_3 = Y_9 + Y_8$

$A_2 = Y_7 + Y_6 + Y_5 + Y_4$

$A_1 = Y_7 + Y_6 + Y_3 + Y_2$

$A_0 = Y_9 + Y_7 + Y_5 + Y_3 + Y_1$.

Y_9 Y_8 Y_7 Y_6 Y_5 Y_4 Y_3 Y_2 Y_1 Y_0



Priority Encoder :-

* Priority Encoder is an Encoder circuit that includes Priority function.
* The operation of the priority Encoder is such that if two or more inputs equal to 1 at the same time, the Input having highest priority will be taken precedence.
* In addition to the two outputs x and y, the circuit has a third output designated by V, this is a Valid bit Indicator that is set to 1 when one or more inputs are equal to 1
* If all inputs are 0, there is no valid input and V is equal to 0.
* The other outputs are not inspected when V equals 0 and are Specified as don't care conditions.

Truth table :-

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

* Input D_3 has the highest priority. Regardless of the values of other inputs.

* D_2 has the next priority & o/p is 10 if $D_2 = 1$ provided $D_3 = 0$. Regardless of the values of the other two lower priority inputs.

* The output of D_1 is generated only if higher priority Inputs are 0.

K. map for X

D_0D_1 \ D_2D_3	00	01	11	10
00	X	1	1	1
01		1	1	1
11		1	1	1
10		1	1	1

D_1 (01, 11, 10) ; D_0 (11, 10) ; D_3' (01, 11, 10)

$$x = D_2 + D_3$$

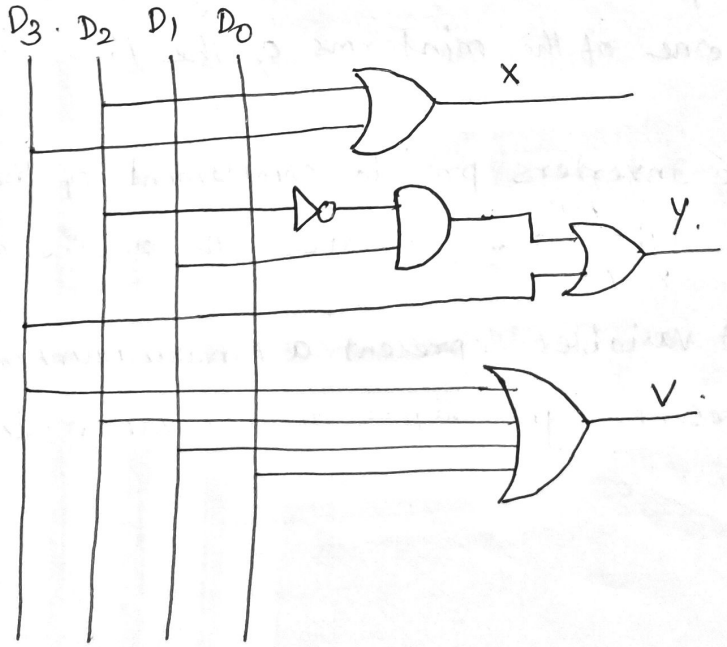
k. map for Y.

D_0D_1 \ D_2D_3	00	01	11	10
00		1	1	
01	1	1	1	
11	1	1	1	
10		1	1	

D_1 (00, 01, 11) ; D_0 (11, 10) ; D_3 (01, 11, 10)

$$Y = D_3 + D_1 \cdot D_2'$$

$$V = D_0 + D_1 + D_2 + D_3$$



DECODER :-

* Decoder is a combinational circuit that converts binary information from n input lines to a maximum 2^n unique output lines.

* If the n-bit coded information has unused combinations. the decoder may have fewer than 2^n outputs.

* The decoder presented is n-to-m line decoders where $m \leq 2^n$, Their purpose is to generate 2^n minterms of n input variables

* Decoder is also used in conjuction with other code converters, such as BCD- to seven segment decoder.

* Three to eight line decoder :-

3-to-8 decoder.

* The 3 inputs are decoded into eight outputs. each representing one of the minterms of the three input variables

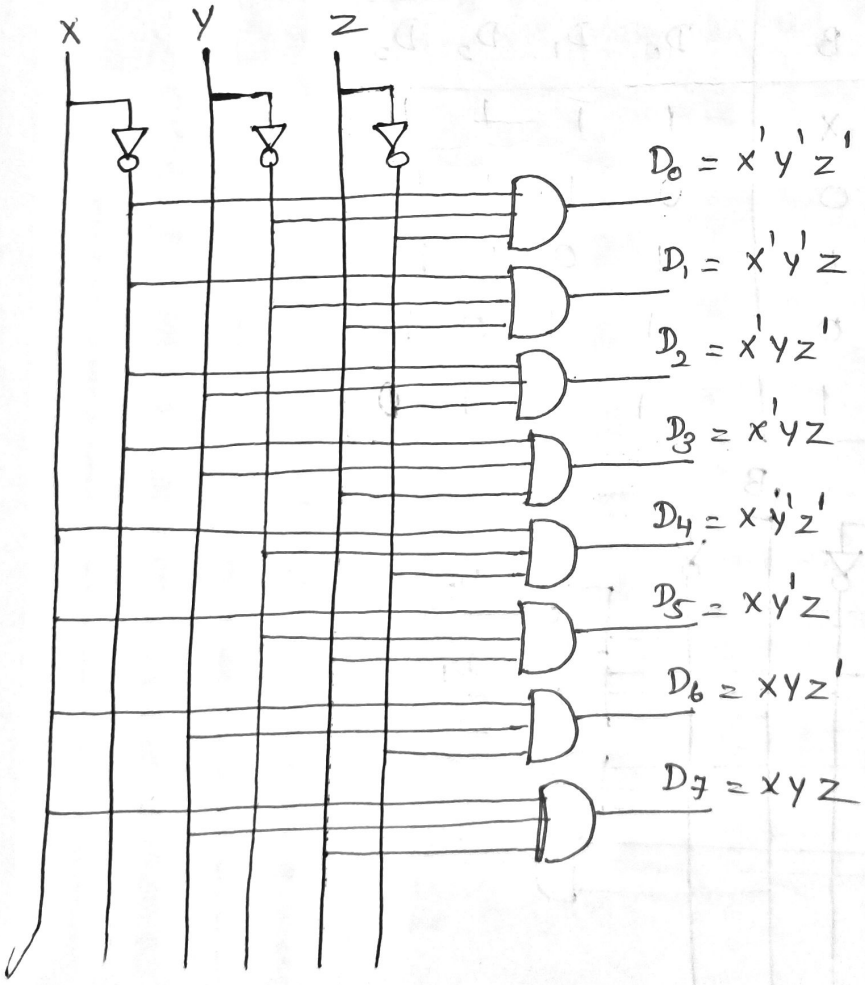
* The three Inverters provide complement of the Inputs.

* each one of the eight AND gates generates one of the minterms.

* The Input variables represent a binary number, & the outputs represent eight digits of a number. in octal number system.

*

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



* Application of decoder is binary to octal conversion.

* for each possible input combination, there are seven outputs equal to 0 and only one equal to 1

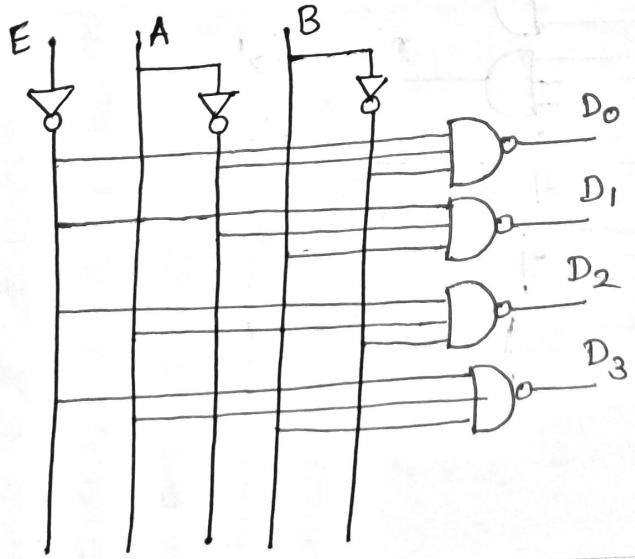
* The output whose value is equal to 1 represents the minterm equivalent of the binary number currently available in the input lines.

Decoder with Enable Input :-

* Decoders include one or more enable inputs to control the circuit operation.

* 2-to-4 decoder with enable input constructed with Nand gates.

E	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



Combinational logic implementation:-

Implementation of a full adder with a decoder:-

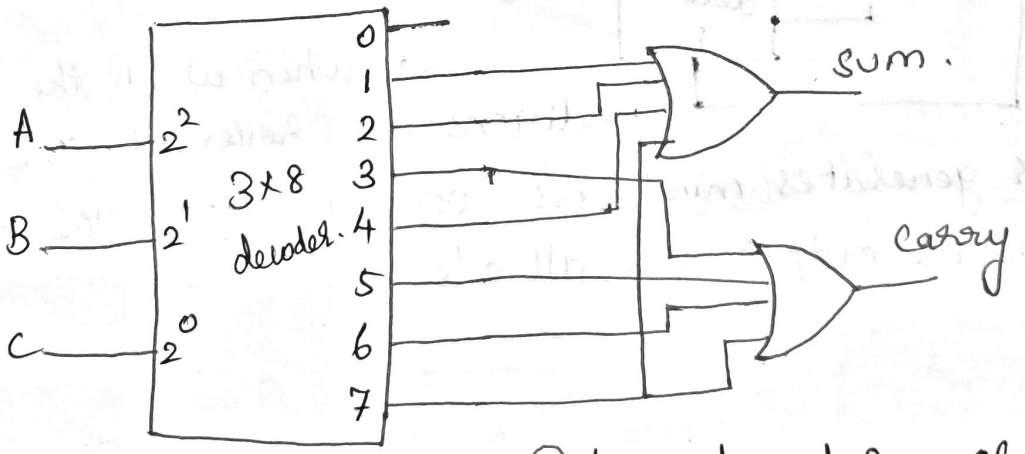
* From the truth table of full adder.

	A	B	C	Sum	Carry
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

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

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

* Step 1: Boolean function for the circuit be expressed as Sum of minterms.
* A decoder is then chosen that generates all the minterms of the i/p variables
* The inputs of each OR gate are selected from the decoders outputs



* The OR gate for output (S) forms logical sum of minterms. 1, 2, 4, and 7
* The OR gate for output (C) forms logical sum of minterms 3, 5, 6 and 7

* Decoder is enabled when E is equal to 0.

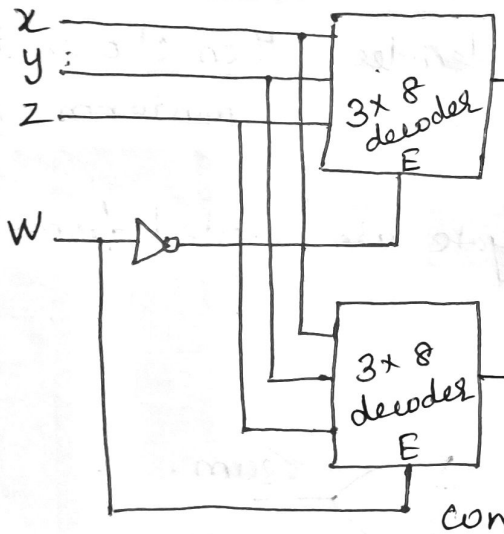
* only one output can be equal to 0 at any given time.

* Circuit is disabled when E is equal to 1 regardless of two inputs.

* A decoder with enable input can function as a demultiplexer.

* Decoders with enable inputs can be connected together to form a larger decoder.

* Two 3 to 8 decoders with enable inputs connected to form 4 to 16 decoder.



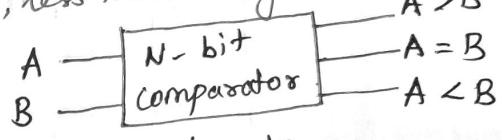
* when w=0 the top decoder is enabled & other is disabled.

* The top decoder generates minterms 000 to 111 and bottom decoder o/p's are all 0's

* when w=1 the enable conditions are reversed. The bottom decoder generates minterms 1000 to 1111 & the top decoder outputs are all 0's

Magnitude Comparator :-

* Magnitude comparator is a combinational circuit that compares two digital or binary numbers in order to find out whether one binary number is equal, less than or greater than the other binary number.



* Design a circuit for which we will have two inputs one for A and other for B and have three output terminals, one for A > B, one for A = B and one for A < B condition.

1- Bit magnitude Comparator :-

* A comparator used to compare two bits is called a single bit comparator.

* It consists of two inputs each for two single bit numbers & three outputs to generate less than, equal to and greater than between two binary numbers.

Truth table :-

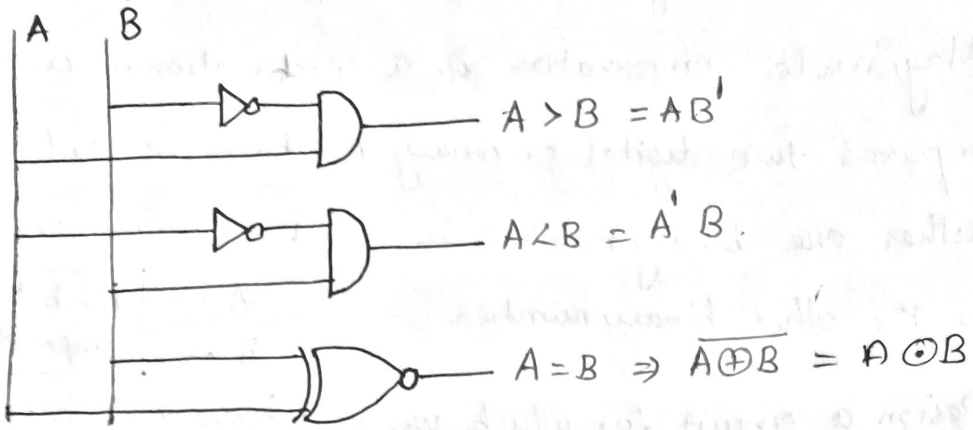
A	B	A < B	A = B	A > B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

$A < B \Rightarrow A' B$.

$A > B \Rightarrow A B'$

$A = B \Rightarrow A'B' + AB$.

Logic Diagram :-

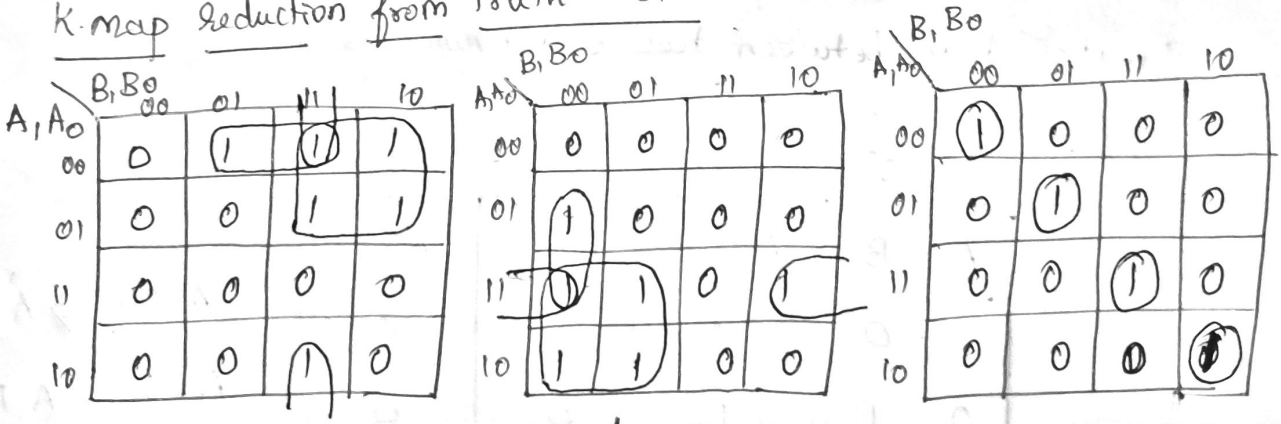


2 - Bit magnitude Comparator :-

* A comparator used to compare two binary numbers each of two bits is called a 2-bit magnitude comparator.

It consists of four inputs & three outputs to generate less than, equal to & greater than between two binary numbers.

K-map Reduction from truth table.



$A < B \Rightarrow A_1' B_1 + A_1' A_0' B_0 + A_0' B_1 B_0$

$A > B \Rightarrow A_1 B_1' + A_0 B_1' B_0' + A_1 A_0 B_0'$

$A = B \Rightarrow A_1' A_0' B_1' B_0' + A_1' A_0 B_1' B_0 + A_1 A_0 B_1 B_0 + A_1 A_0' B_1 B_0'$

Truth table :-

Input				output.		
A_1	A_0	B_0	B_1	$A < B$	$A = B$	$A > B$
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

be used

$$A=B = A_1' A_0' B_1' B_0' + A_1' A_0 B_1' B_0 + A_1 A_0 B_1 B_0 + A_1 A_0' B_1 B_0'$$

$$= A_1' B_1' (A_0' B_0' + A_0 B_0) + A_1 B_1 (A_0 B_0 + A_0' B_0')$$

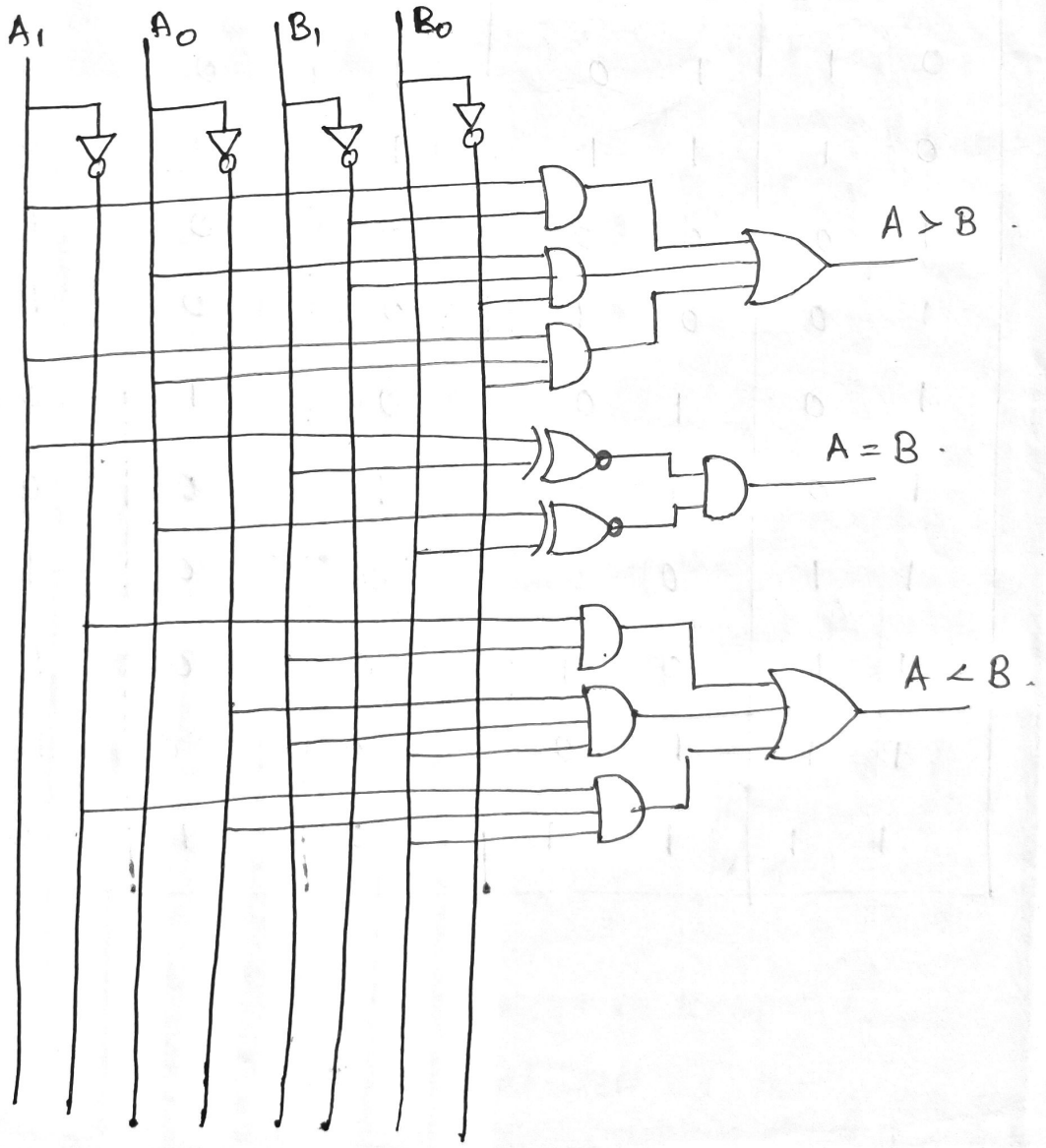
$$= (\overline{A_1 \oplus B_1})(\overline{A_0 \oplus B_0})$$

Therefore:

$$A > B = A_1 B_1' + A_0 B_1' B_0' + A_1 A_0 B_0'$$

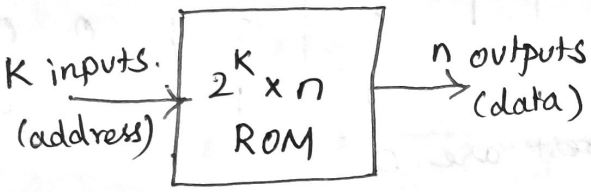
$$A = B = (\overline{A_0 \oplus B_0})(\overline{A_1 \oplus B_1})$$

$$A < B = A_1' B_1 + A_0' B_1 B_0 + A_1' A_0' B_0$$



Read Only Memory :-

* ROM is a memory device in which permanent binary information is stored.
* The binary information must be specified by the designer & is then embedded in the unit to form the required inter connection pattern.
* once the pattern is established, it stays within the unit even when power is turned off & on again.



* ROM consists of K inputs & n outputs.
* Input gives address for memory
* output gives data bits of the stored word that is selected by the address.

* Consider a 4 x 4 ROM which means that it has total of 4 addresses at which information is stored. & each addresses has 4-bit information.
* construct the truth table.

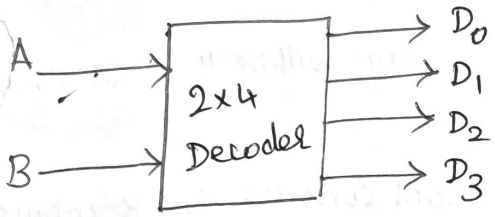
Inputs		outputs.			
X	Y	A	B	C	D
0	0	0	0	1	1
0	1	1	1	0	0
1	0	1	1	1	1
1	ϕ	0	1	1	1

* Truth table shows that at location 00, Content to be stored is 0011 and
location 01 - 1100
location 10 - 1111
location 11 - 0111

* Based on total no. of addresses in ROM & Length of their Content decide the decoder as well as no. of OR gates to be used.

* Generally for a $2^k \times n$ ROM, a $k \times 2^k$ decoder is used. & the total no. of OR gates = total no. of bits stored.

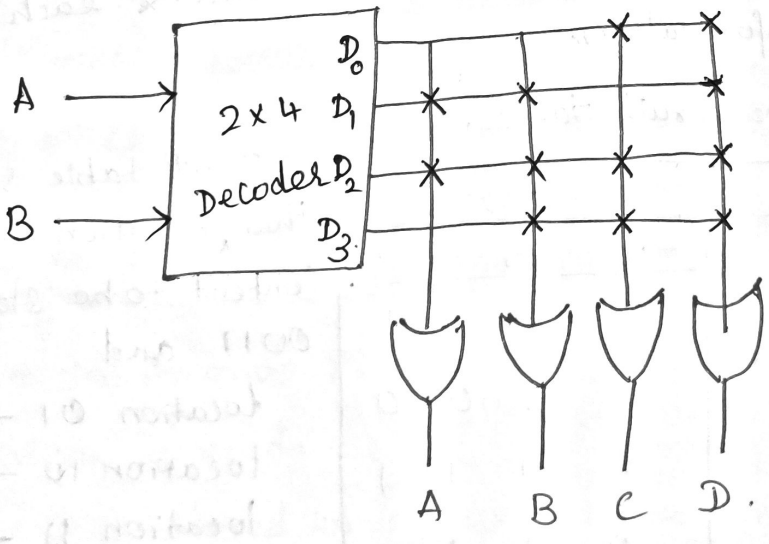
* So In case of 4×4 ROM. the decoder to be used is 2×4 decoder & no. of OR gates = 4.



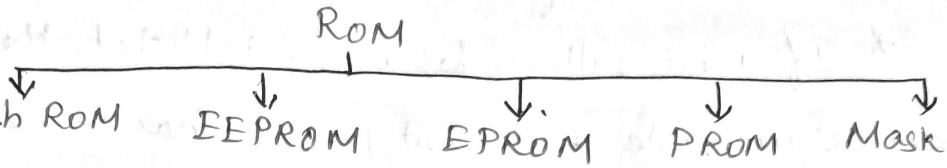
A	B	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

* when both the inputs are 0, then only D_0 is 1 and rest are 0.

* when Input = 01 then $D_1 = 1$
when Input = 10 then $D_2 = 1$
when Input = 11 then $D_3 = 1$



Types of ROM :-



Mask ROM :-

* The specification of ROM is taken by the manufacturer from customer in tabular form
* Then corresponding masks for the paths are produced to get the desired output.
* This ROM is costly as vendor charges special fees for making a particular ROM.
* They are used in Network operating systems, Server operating systems, storing of fonts for laser printers, sound data in electronic musical Instruments.

PROM :-

* It stands for programmable Read only memory.
* Prepared as a blank memory and then it is programmed to store the information
* To program PROM, a PROM programmer or PROM burner is used.
* Data stored in it cannot be modified → so it is called one-time programmable.
* Used in cell phones, video game consoles, RFID tags medical devices,

EPROM :-

* Erasable programmable Read only memory.

* If a bit pattern has been established, the PROM becomes unusable if the bit pattern has to be changed.

* The problem has been overcome by EPROM

* when EPROM is placed under a special ultraviolet light for a length of time, the shortwave radiation makes the EPROM return to its initial state; which then can be programmed accordingly.

EEPROM :- Electrically erasable programmable Read-only memory.

* Similar to EPROM except that EEPROM is returned to its initial state by application of an electrical signal in place of ultraviolet light.

* ease of erasing. It erases (or) writes one byte of data at a time.. used for storing computer system BIOS.

Flash ROM :-

* enhanced version of EEPROM.

* In flash memory blocks of data can be deleted or written at a particular time.

* flash memory is faster than EEPROM.

* many modern PCs have their BIOS stored on a flash memory chip. called as flash BIOS.

Programmable logic Array :-

Combinational PLDs :

* The PROM is a combinational programmable logic device (PLD) an integrated circuit with programmable gates divided into an AND array and an OR Array to provide an AND-OR

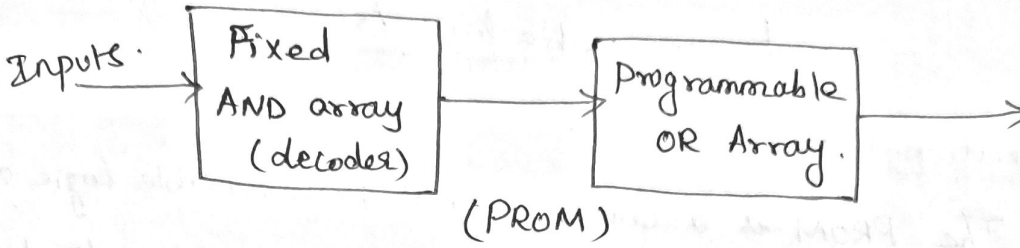
* The three major types of combinational PLD's, differing in the placement of programmable connections in the AND-OR arrays are PROM, PAL, PLA.

* PROM - Programmable Read only memory has a fixed fixed AND array constructed as a decoder and a programmable OR array. The programmable OR gates implement Boolean function in sum-of minterms form.

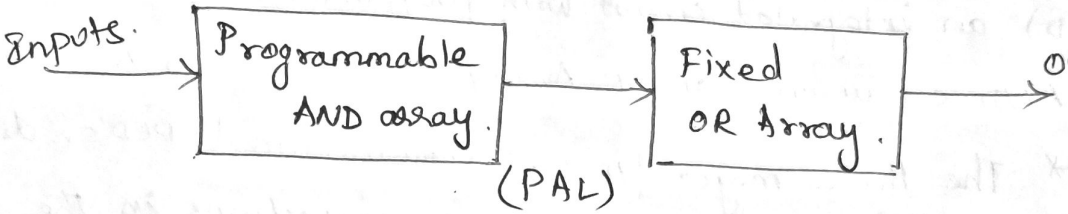
* PAL - Programmable array logic has a programmable AND array and a fixed OR array. The AND gates are programmed to provide the product terms for the Boolean functions. which are logically summed in each OR gate.

* PLA - Programmable logic array.

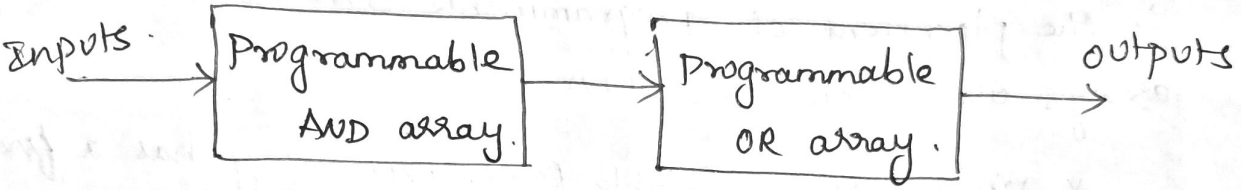
* The most flexible PLD is PLA, in which both. AND and OR arrays can be programmed. The product terms in the AND array may be shared by any OR gate to provide the required sum-of products. Implementation.



(PROM)



(PAL)



(PLA)

Programmable Logic Array :-

* PLA presents the Boolean function in the form of a sum of product (SOP)

* The designing of this PLA can be done using logic gates like AND, OR, and NOT by fabricating on the chip.

Example:-

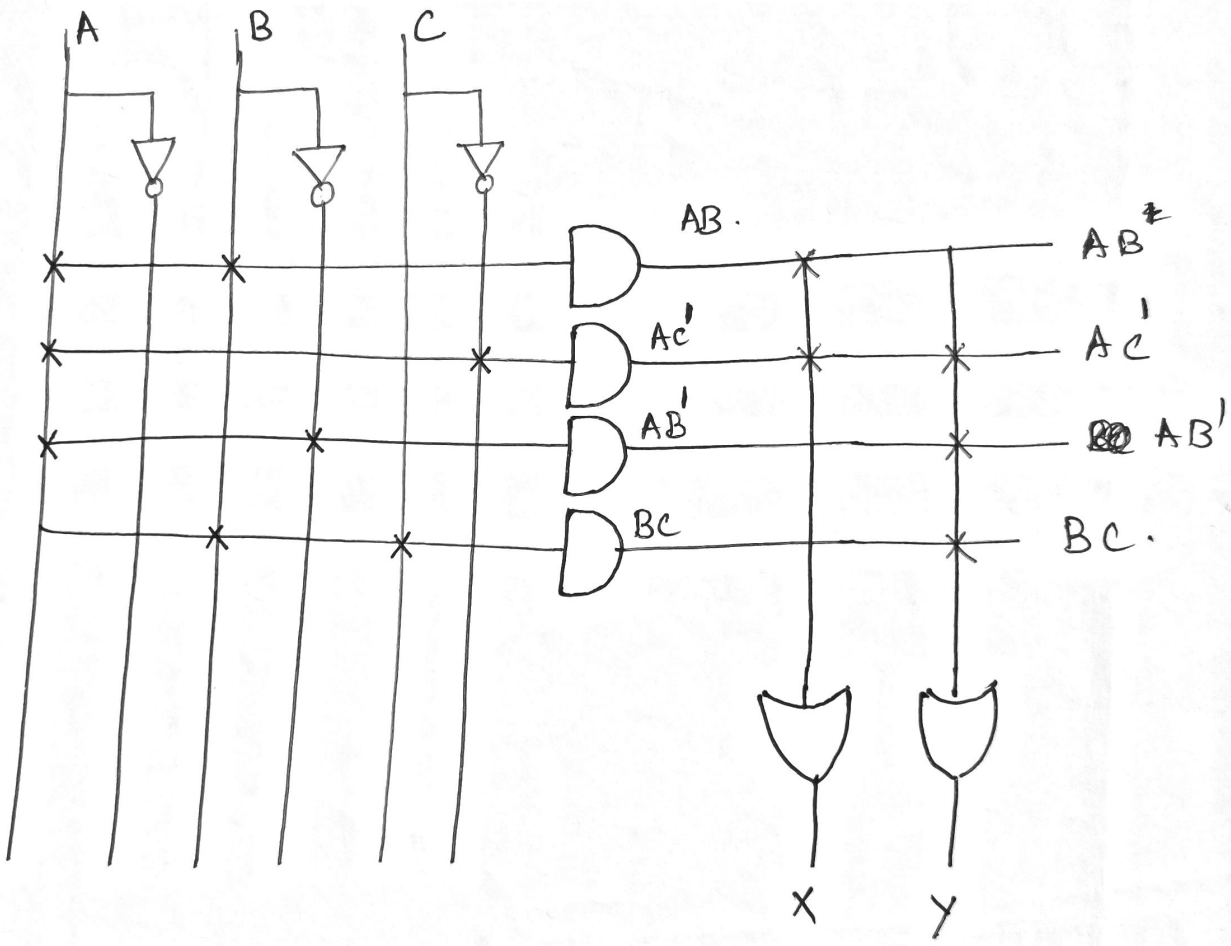
Implement the following Boolean expression using PLA.

$$X = AB + Ac'$$

$$Y = AB' + BC + Ac'$$

PLA Programming Table.

	Product term	Inputs. A B C	outputs. X, Y
AB	1	1 1 –	1 –
Ac'	2	1 – 0	1 1
AB'	3	1 0 –	– 1
Bc	4	– 1 1	– 1

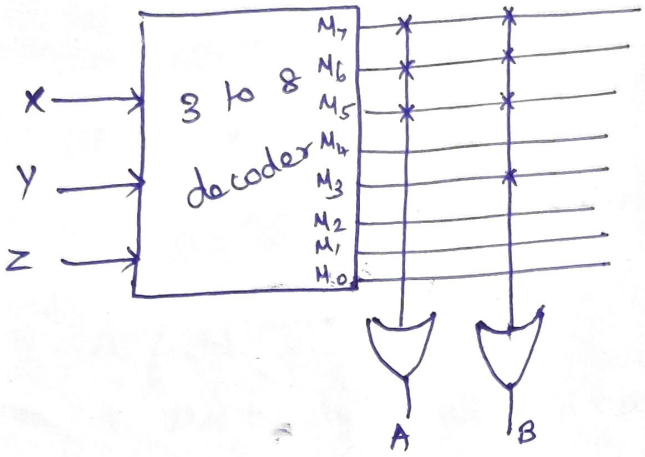


Problem:

Implement the following boolean functions using PROM

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

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

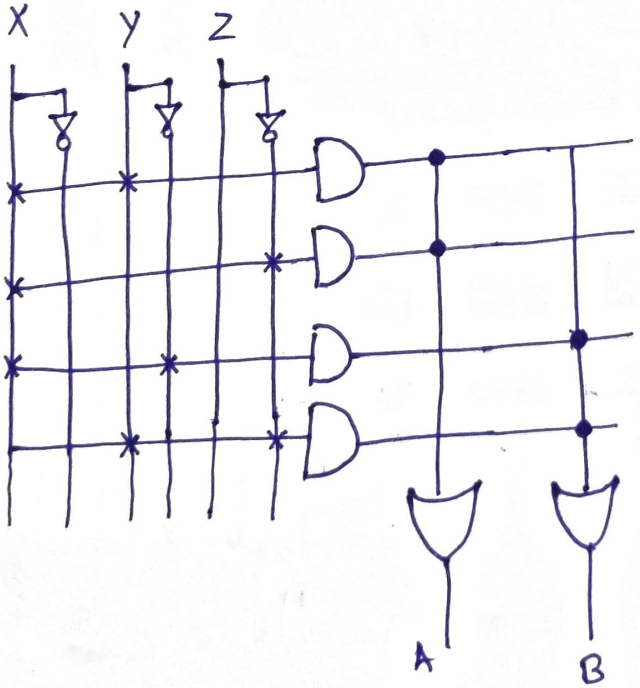


Problem:

Implement the following Boolean functions using PAL

$A = xy + xz'$

$A = xy' + yz'$



Quine - Mccluskey Method :-

* Karnaugh map is an effective way to simplify switching functions which have a small number of variables.

* When the number of variables is large or if several functions must be simplified then Quine-Mccluskey method provides the systematic simplification procedure.

* The procedure consists of two main steps.
    * Determination of prime implicants
    * Prime implicant chart.

Determination of Prime Implicants :-

* Represent each term (min term) by their binary code.

* Define the number of 1's in binary number as the Index of the number.

* Group all the binary numbers of same Index into a group

* The binary numbers are listed in the ascending order of their decimal number equivalent

* Compare the terms in lower Index to the set whose Index is 1 greater.

* Eliminate all redundant variables by applying the Theorem $xy + xy' = x$

* check off all the terms that entered into the combination. The ones that are left are Prime implicants

* Repeat steps of comparing, eliminating until no further reduction is possible.

* We obtain the set of all prime implicants.

Prime Implicant chart :-

* To find the essential Prime implicants.

* Each column carries a decimal number at the top corresponding to the minterms - The columns are signed by such a number in ascending order.

* make a cross under each decimal number contained is the prime implicant represented by that row.

* Find all the columns that contain a single cross and circle them.

* Place an asterisk or strike out the corresponding row of the circled cross.

* The rows marked with an asterisk are the essential prime Implicants.

* Strike out the columns which are belonging to the essential prime implicant rows.

* If Prime implicants are left over see for minimum set of prime implicants to cover the remaining columns.

Problem: 1

Using Quine Mccluskey method find the minimum Sum of product expression for

$$f(a,b,c,d) = \Sigma m(0,1,2,5,6,7,8,9,10,14)$$

Solution.

* Arrange the decimal values in ascending order & write down the binary code & Index

		8 4 2 1	Index
0	-	0 0 0 0	0
1	-	0 0 0 1	1
2	-	0 0 1 0	1
5	-	0 1 0 1	2
6	-	0 1 1 0	2
7	-	0 1 1 1	3
8	-	1 0 0 0	1
9	-	1 0 0 1	2
10	-	1 0 1 0	2
14	-	1 1 1 0	3

* Now group them based on the number of 1's in the Index & assign Index number accordingly - arrange them in ascending order.

Index No	Column - I: Decimal Number	Column - I: Binary Representation a b c d	Column - II: Decimal Number	Column - II: Binary representation a b c d	Column - III: Decimal Number	Column - III: Binary representation a b c d
Index 0	0 ✓	0 0 0 0	0, 1 ✓	0 0 0 –	0, 1, 8, 9 ✓	– 0 0 –
			0, 2 ✓	0 0 – 0	0, 2, 8, 10 ✓	– 0 – 0
Index 1	1 ✓	0 0 0 1	0, 8 ✓	– 0 0 0	~~0, 8, 1, 9~~	~~– 0 0 –~~
	2 ✓	0 0 1 0			~~0, 8, 2, 10~~	~~– 0 – 0~~
	8 ✓	1 0 0 0	1, 5	0 – 0 1		
			1, 9 ✓	– 0 0 1	2, 6, 10, 14 ✓	– – 1 0
	5 ✓	0 1 0 1	2, 6 ✓	0 – 1 0	~~2, 10, 6, 14~~	~~– – 1 0~~
	6 ✓	0 1 1 0	2, 10 ✓	– 0 1 0		
Index 2	9 ✓	1 0 0 1	8, 9 ✓	1 0 0 –	* The striked out Prime implicants are the repeated ones	
	10 ✓	1 0 1 0	8, 10 ✓	1 0 – 0		
Index 3	7 ✓	0 1 1 1	5, 7	0 1 – 1		
	14 ✓	1 1 1 0	6, 7	0 1 1 –		
			6, 14 ✓	– 1 1 0		
			10, 14 ✓	1 – 1 0		

* Therefore the Prime Implicants are

	a b c d	
1, 5.	0 – 0 1	$a' c' d$
5, 7	0 1 – 1	$a' b d$
6, 7	0 1 1 –	$a' b c$
0, 1, 8, 9	– 0 0 –	$b' c'$
0, 2, 8, 10	– 0 – 0	$b' d'$
2, 6, 10, 14	– – 1 0	$c d'$

Prime Implicant chart :-

Decimal Number	Binary representation a, b, c, d	0	1	2	5	6	7	8	9	10	14
0, 1, 8, 9 ←	$b'c'$ *	X	X					X	X		
0, 2, 8, 10	$b'd'$	X		X				X		X	
2, 6, 10, 14 ←	cd' *			X		X				X	X
1, 5	$a'c'd$		X		X						
5, 7	$a'bd$ ←				X		X	← covers both the left Implicants			
6, 7	$a'bc$					X	X				

Single column with X

Single column with X

* Find out the columns with single X marking & Strike out the corresponding Row.

* Strike out the column with x marking corresponding to the striked out Row.

* find out the left over Implicants & Select the minimum set so as to cover all the leftover Prime Implicants.

Essential Prime implicants :-

If a minterm is covered by only one prime Implicant then the Prime Implicant is called Essential Prime Implicant.

The resulting minimum sum of products is

$$f(a, b, c, d) = b'c' + cd' + a'bd$$

Problem : 2

Using Quine mccluskey Method find the minimum sum of product expression for

$$f(x_1, x_2, x_3, x_4, x_5) = \Sigma(0,2,4,5,6,7,8,10,14,17,18,21,29,31) + \Sigma_d(11,20,22)$$

Solution :-

	x_1	x_2	x_3	x_4	x_5
0	0	0	0	0	0
2	0	0	0	1	0
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
10	0	1	0	1	0
14	0	1	1	1	0
17	1	0	0	1	0
18	1	0	0	1	0
21	1	0	1	0	1
29	1	1	1	0	1
31	1	1	1	1	1
11	0	1	0	1	1
20	1	0	1	0	0
22	1	0	1	1	0

Decimal Number	Representation of Each term $x_1 x_2 x_3 x_4 x_5$
0 ✓	0 0 0 0 0
2 ✓	0 0 0 1 0
4 ✓	0 0 1 0 0
8 ✓	0 1 0 0 0
5 ✓	0 0 1 0 1
6 ✓	0 0 1 1 0
10 ✓	0 1 0 1 0
17 ✓	1 0 0 0 1
18 ✓	1 0 0 1 0
20 ✓	1 0 1 0 0
7 ✓	0 0 1 1 1
11 ✓	0 1 0 1 1
14 ✓	0 1 1 1 0
21 ✓	1 0 1 0 1
22 ✓	1 0 1 1 0
29 ✓	1 1 1 0 1
31 ✓	1 1 1 1 1

Decimal Numbers.	Representation of Each term $x_1 x_2 x_3 x_4 x_5$
0, 2 ✓	0 0 0 – 0
0, 4 ✓	0 0 – 0 0
0, 8 ✓	0 – 0 0 0
2, 6 ✓	0 0 – 1 0
2, 10 ✓	0 – 0 1 0
2, 18 ✓	– 0 0 1 0
4, 5 ✓	0 0 1 0 –
4, 6 ✓	0 0 1 – 0
4, 20 ✓	– 0 1 0 0
8, 10 ✓	0 1 0 – 0
5, 7 ✓	0 0 1 – 1
5, 21 ✓	– 0 1 0 1
6, 7 ✓	0 0 1 1 –
6, 14 ✓	0 – 1 1 0
6, 22 ✓	– 0 1 1 0
10, 14 ✓	0 1 – 1 0
10, 11	0 1 0 1 –
17, 21	1 0 – 0 1
18, 22 ✓	1 0 – 1 0
20, 21 ✓	1 0 1 0 –
20, 22 ✓	1 0 1 – 0
21, 29	1 – 1 0 1
29, 31	1 1 1 – 1

Decimal Number	Representation of Each term $x_1 x_2 x_3 x_4 x_5$
0, 2, 4, 6	0 0 – – 0
0, 2, 8, 10	0 – 0 – 0
2, 6, 10, 14	0 – – 1 0
2, 6, 18, 22	– 0 – 1 0
4, 5, 6, 7	0 0 1 – –
4, 5, 20, 21	– 0 1 0 –
4, 20, 6, 22	– 0 1 – 0

Prime implicant Table.

Decimal Number	0	2	4	5	6	7	8	10	14	17	18	21	29	31
0, 2, 4, 6	X	X	X		X									
*0, 2, 8, 10	X	X					(X)	X						
*2, 6, 10, 14		X			X			X	(X)					
*2, 6, 18, 22		X			X						(X)			
*4, 5, 6, 7			X	X	X	(X)								
4, 5, 20, 21			X	X								X		
4, 20, 6, 22			X		X									
10, 11								X						
*17, 21										(X)		X		
21, 29												X	X	
*29, 31													X	(X)

* The essential prime implicants are

	$x_1\ x_2\ x_3\ x_4\ x_5$
0, 2, 8, 10	0 − 0 − 0
2, 6, 10, 14	0 − − 1 0
2, 6, 18, 22	− 0 − 1 0
4, 5, 6, 7	0 0 1 − −
17, 21	1 0 − 0 1
29, 31	1 1 1 − 1

* don't care terms min terms = 11, 20, 22 are not taken into account for Prime implicant

$$f(x_1 x_2 x_3 x_4 x_5) = x_1' x_3' x_5' + x_1' x_4 x_5' + x_2' x_4 x_5' + x_1' x_2' x_3 + x_1 x_2' x_4' x_5 + x_1 x_2 x_3 x_5$$

Problem : 3

$f(x_1, x_2 x_3 x_4 x_5) = \Sigma (0, 1, 2, 8, 9, 15, 17, 21, 24, 25, 27, 31)$

Solution:

		x_1	x_2	x_3	x_4	x_5
0	-	0	0	0	0	0
1	-	0	0	0	0	1
2	-	0	0	0	1	0
8	-	0	0	0	0	0
9	-	0	1	0	0	1
15	-	0	1	1	1	1
17	-	1	0	0	0	1
21	-	1	0	1	0	1
24	-	1	1	0	0	0
25	-	1	1	0	0	1
27	-	1	1	0	1	1
31	-	1	1	1	1	1

	Decimal Numbers	Binary Representation $x_1 x_2 x_3 x_4 x_5$	Decimal Numbers	Binary Representation $x_1 x_2 x_3 x_4 x_5$	Decimal Numbers	Binary Representation
Index 0	0 ✓	0 0 0 0 0	0, 1 ✓	0 0 0 0 –	0, 1, 8, 9	0 – 0 0 –
Index 1	1 ✓	0 0 0 0 1	0, 2	0 0 0 – 0	1, 9, 17, 25	– – 0 0 1
	2 ✓	0 0 0 1 0	0, 8 ✓	0 – 0 0 0	8, 24, 9, 25	– 1 0 0 –
	8 ✓	0 1 0 0 0	1, 9 ✓	0 – 0 0 1	0, 8, 1, 9	0 – 0 0 –
Index 2	9 ✓	0 1 0 0 1	1, 17	– 0 0 0 1		
	17 ✓	1 0 0 0 1	8, 9 ✓	0 1 0 0 –		
	24 ✓	1 1 0 0 0	8, 24 ✓	– 1 0 0 0		
Index 3	21 ✓	1 0 1 0 1	9, 25 ✓	– 1 0 0 1		
	25 ✓	1 1 0 0 1	17, 21	1 0 – 0 1		
Index 4	15 ✓	0 1 1 1 1	17, 25 ✓	1 – 0 0 1		
	27 ✓	1 1 0 1 1	24, 25	1 1 0 0 –		
Index 5	31 ✓	1 1 1 1 1	15, 31	– 1 1 1 1		
			27, 31	1 1 – 1 1		

Prime Implicant Table :-

Decimal	0	1	2	8	9	15	17	21	24	25	27	31
0, 1, 8, 9	X	X		X	X							
1, 9, 17, 25		X			X		X			X		
8, 24, 9, 25				X	X				X	X		
* 0, 2	X		(X)									
1, 17		X					X					
* 17, 21							X	(X)				
24, 25									X	X		
* 15, 31						(X)						X
* 27, 31											(X)	X

* The prime implicants ~~are~~ cannot be reduced further. So remove the essential prime implicants from the table & draw a table with the remaining implicants.

	Decimal	0	1	8	9	17	24	25	
* a	0, 1, 8, 9	(X)	X	X	X				
b	1, 9, 17, 25		X		X	X		X	
c	8, 24, 9, 25			X	X		X	X	
d	1, 17		X			X			
e	24, 25						X	X	

Column dominance

$9 \supset 8$

$25 \supset 24$

$1 \supset 0$

∴ Column 8, 24, & 0 can be eliminated

Row dominance

$b \supset d$

$c \supset e$

∴ Rows d & e can be eliminated

Decimal	1	9	17	25
0, 1, 8, 9	X	X		
* 1, 9, 17, 25	X	X	X	X
8, 24, 9, 25		X		X

Row dominance

$b \supset c$

* The prime implicants are

	x_1	x_2	x_3	x_4	x_5
0, 1, 8, 9	-	0	-	0	0	-
1, 9, 17, 25	-	-	0	0	1
0, 2	0	0	0	-	0
17, 21	1	0	-	0	1
15, 31	-	1	1	1	1
27, 31	1	1	-	1	1

$$f(x_1 x_2 x_3 x_4 x_5) = x_1' x_3' x_4' + x_3' x_4' x_5 + x_1' x_2' x_3' x_5' + x_1 x_2' x_4' x_5 + x_2 x_3 x_4 x_5 + x_1 x_2 x_4 x_5$$

Assignment

FPGA - Basics :-

Introduction :-

* Field programmable Gate Array - is a device that consists a matrix of reconfigurable gate array logic circuitry.

* FPGA use dedicated hardware for processing logic & do not have an operating system.

* FPGA based systems can literally rewire their internal circuitry to allow reconfiguration after the control system is deployed to the field.

* FPGA are truely parallel in nature, so different processing operations do not have to compete for the same resources.

* multiple control loops can run on a single FPGA device at different rates.

* FPGA devices deliver the performance & rediability of dedicated hardware circuitry.

Need for FPGA :-

* SSI, MSI, and LSI components required interconnections to help, connect the IC in order to generate global control signals & Data signals from one subsystem to another sub system.

* TO solve the problem custom IC's were developed which replaced large amount of Interconnect → Improving Performance & Reducing system complexity & manufacturing cost.

* Bcoz of increased design time, the custom IC's were not on time to market sensitive.

* FPGAs were introduced as an alternative to custom IC's for implementing entire system on chip & to provide flexibility of reprogramability to the user.

* Another Advantage of FPGA is with the help of computer Aided design tools circuit could be implemented in a short amount of time.

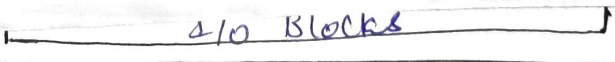
FPGA structural classification :-

* Basic structure of an FPGA includes logic elements, Programmable interconnects and Memory.
* on the basis of internal arrangement of blocks FPGA can be divided into three classes.

  * Symmetrical arrays
  * Row based architecture
  * Hierarchical PLD's

Symmetrical arrays:

* It consists of configurable logic blocks (CLB) arranged in rows and columns of a matrix and interconnect laid out between them.
* Symmetrical matrix is surrounded by I/o blocks which connect it to outside world.
* Each CLB consists of n-input lookuptable and a pair of programmable flipflops.
* I/o blocks also control functions such as tristate control, output transition speed.



* Interconnects provide routing path.

* Direct interconnects b/w adjacent logic elements have smaller delay compared to general purpose interconnect.

Row Based Architecture:-

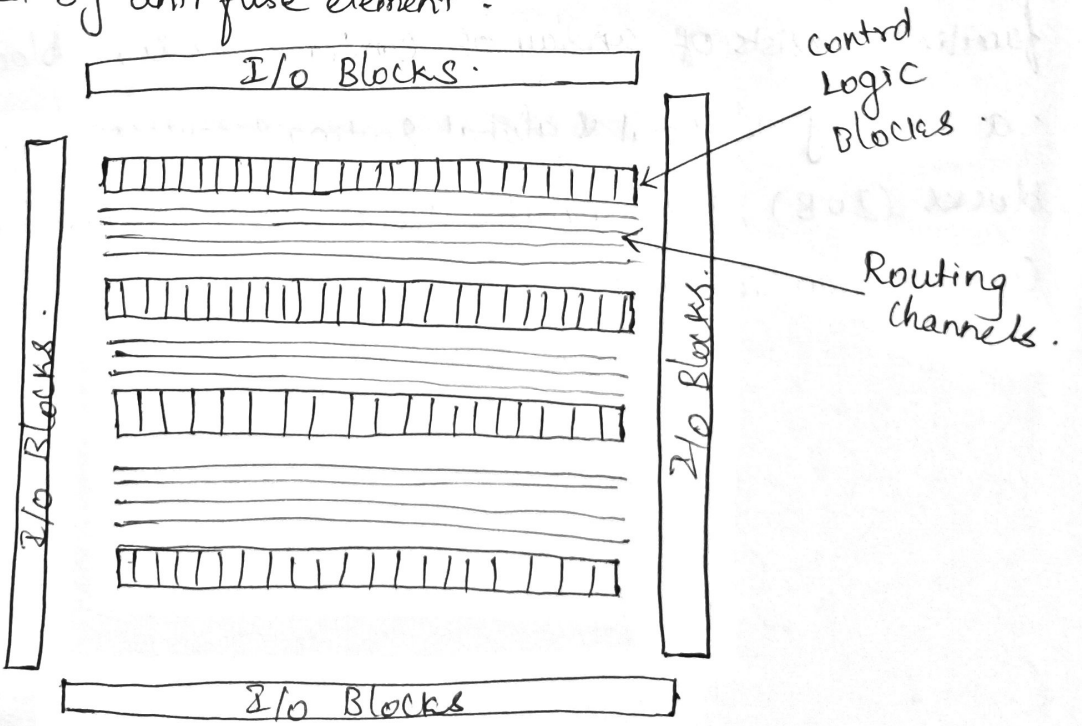
* It consists of Alternating rows of logic modules & programmable interconnect tracks.

* IOB (Input output Block) is located in the periphery of the rows.

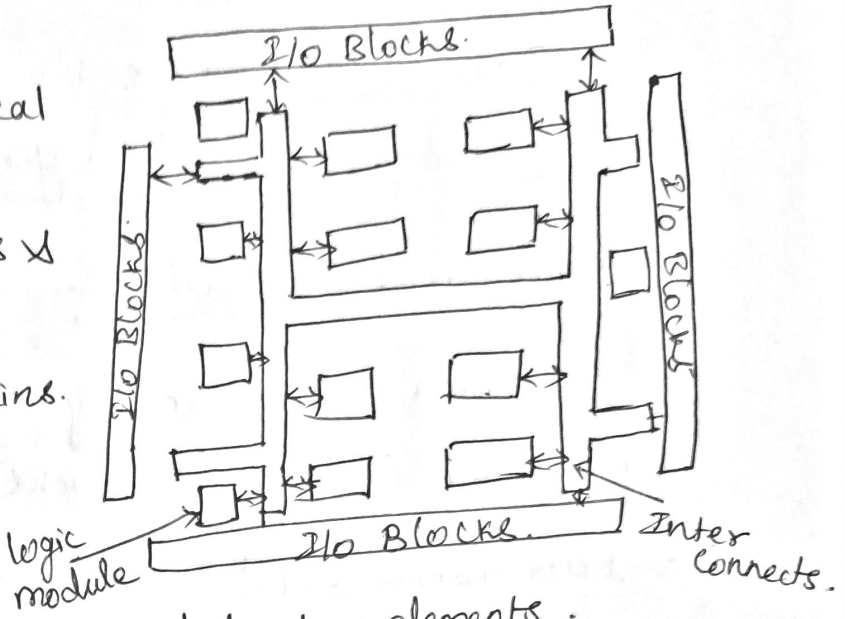
* one row may be connected to adjacent rows via Vertical Interconnect. Logic modules can be implemented in various combinations.

* Combinational modules – only combinational elements.
  Sequential modules – both combinational elements along with Flipflops.

* Routing tracks are divided into smaller segments connected by anti fuse element.



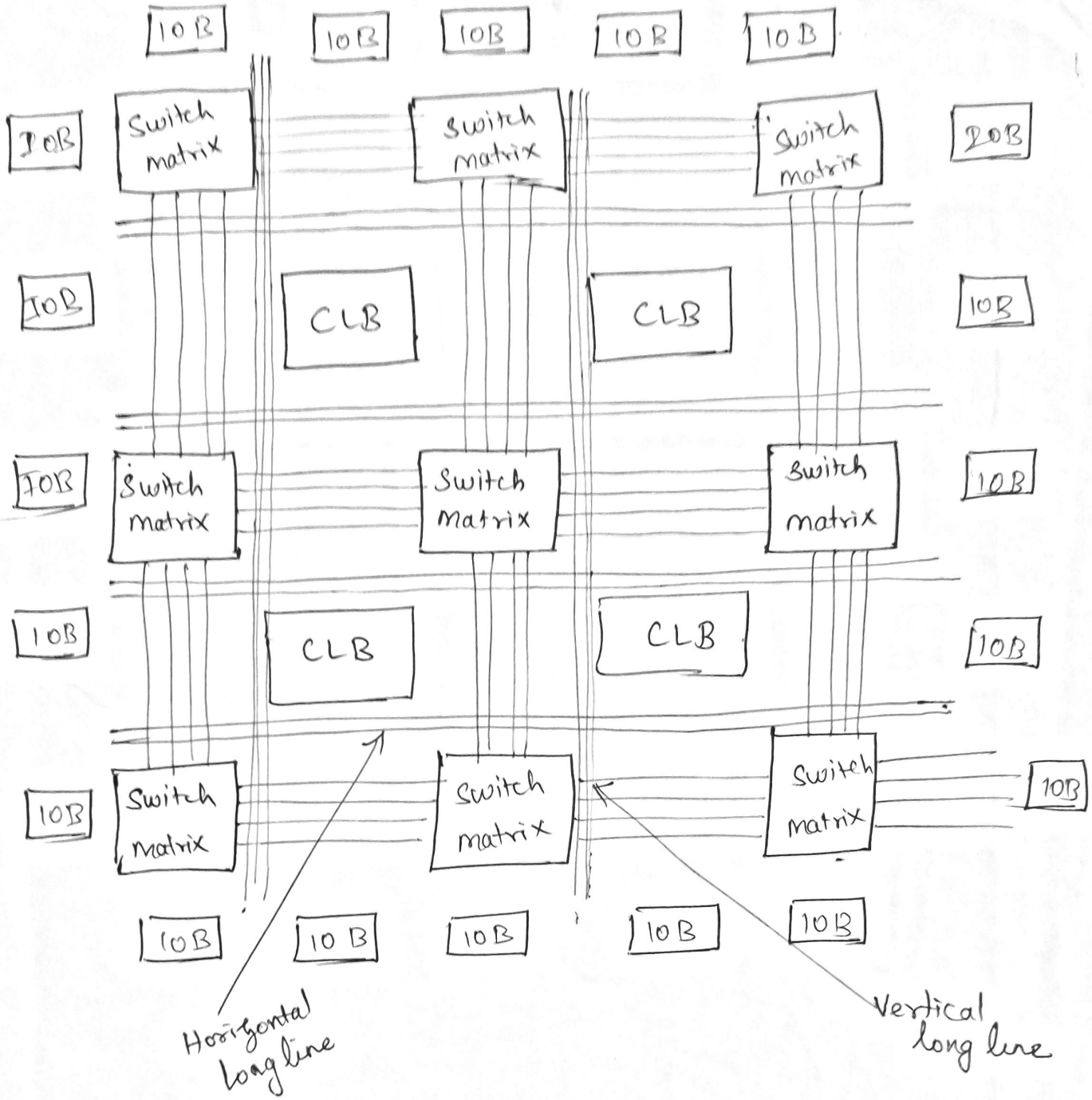
Hierarchical PLD's.



* Designed in hierarchical manner with top level containing only logic blocks & interconnects.
* Each logic block contains number of logic modules.
* logic module has Combinational as well as sequential function elements.
* functional elements is controlled by programmed Memory.
* Communication b/w logic blocks is achieved by programmable interconnects arrays.
* Input output blocks surround this scheme of logic blocks & enterconnects.

Basic xilinx Architecture :-

* The basic architecture of spartan & earlier device families consists of array of configurable logic blocks (CLB's), a variety of local & global routing resources, Input output blocks (IOB), Programmable I/O buffers & SRAM based Configuration memory.


IOB
Switch matrix
CLB
Horizontal long line
Vertical long line