S.No.	Date	Title	Page No.	Teacher's Sign/ Remarks

Number Systems

↓

Radix/Base [No. of symbols available (digits)]

eg: ✓Decimal - 10 (0,9)

✓Binary - 2 (0,1)

$[25]_{10} = [11001]_2$ $\left(1\times2^4 + 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0\right)$

2 | 25 1
2 | 12 0
2 | 6 0
2 | 3 1
1

$[25.25]_{10} = [11001.01]_2$

0.25 × 2
0.5 × 2
1.0

✓Octal - 8 (0-7)

$[123]_{10} = (173)_8$

$(173)_8 = (123)_{10}$

$(45.17)_8 = [37.234]_{10}$

✓Hexadecimal - 16 [0-9, A-F]

$(125)_{10} \rightarrow (7B)_{16}$

125
78
16 | 500

$(ABC)_{16} \rightarrow [2748]_{10}$

$(18.2A)_{16} \rightarrow [24.129]_{16}$

$(1100)_2 \rightarrow [12]_8$

$(1101010110)_2 \rightarrow (1526)_8$

$(456)_8 \rightarrow (100101110)_2$

$(45.62)_8 \rightarrow (100101.110001)_2$

$(18.2A)_{16} \rightarrow (00011000.00101010)_2$

$11\,01.11\,00\,00\,11\,0 \rightarrow$ 1B86 H

$456_8 \rightarrow 100\,101\,110 \rightarrow 12D$ OCTAL → BIN → HEX

$ABC_{16} \rightarrow 1010\,1011\,1100 \rightarrow 5274$

$456.05_8 \rightarrow 100101110.000101 \rightarrow 12E.14_{16}$

OPERATION ON NUMBER SYSTEM

ADDITION	SUBSTRACTION	MULTIPLICATION
Addend	Substrahend	Multiplicand
Augend	Minuend	Multiplier
sum	Difference	Product

DIVISION
- Dividend
- Divisor

Quotient

R - Radix

R's complement $R's = R^n - N$

(R-1)'s complement.

ie 2 types of complement for any number system

45.

$10^2 - 45 = 100 - 45 = 55$

10's complement of 45 = 55

If there is n number of digits in integer part and m in ~~decimal~~ fractional part

2's comp of $(100)_2 = 2^3 - 100$

$= 1000 - 100 = 100$ (Ans)

$(45)_8 \longrightarrow 8^2 - 45 = 64 - 45$

$= 100 - 45 = (33)_8$

$(45)_{16} \rightarrow 16^2 - 45$

$256 - 45$

$100 - 45 = BB$

8|8 0
1

16|16
16|256
16

100 - 45 = 33

100 - 45 = BB

$(R-1)$'s complement $\rightarrow R^n - N - R^{-m}$

$(45)_{10}$ – 9's comp $\rightarrow 10^2 - 45 - 10^{-0}$

$\rightarrow 100 - 45 - 1 = 54$

$(45.45)_{10} - 100 - 45 - 16^{-2} = 55.54$

$(100)_2 = 2^3 - 100 - 2^{-0}$

$= 1000 - 100 - 1 = (011)_2$

$(45)_{16}$ = BB BA

16's comp 15th comp

1ABC2.24A6

D 45 55 – 54 Subtract every digit from 9-1

B 100 – 100 – 011

D 45 – 33 – 32

H 45 BB BA

Subst from 15

$(1A.BC2.24A6) = E543D.DB5A$

$= E543b.DB5A$

FOR BINARY 100 011 100 – 1's comp

– SUBSTRACTION.

If there is carry, cut that out and it will be right answer and if not take the complement and give negative.

eg: 1000 − 0100

X 100

Take 2's complement,

0100 → 1011 + 1 = 1100

⇒ 1000 + 1100 = 10100 → 4

X [Ignore carry]

Substraction With (r-1)'s complement

The substraction of 2 positive nos M and N Both of the same radix r is

1. Add the minuend with the (r-1)'s complement of the substrahend
2. (a) If an end carry occur add 1 to the LSB (end around carry)
   (b) If there is no carry occurs take the (r-1)'s complement of the output and put a negative sign at the front.

eg: 1000 − 100 = 100

1000 + 0100 (1's comp of 1011) = 1100

1000 + 1100 = *0100 ⇒ 0100

8-4

Substraction using r's complement

1. Add minend with r's complement of substrahend
2. (a) For an end carry discard it
   (b) if there is no carry take the r's complement of o/p and place -ve at front

2nds comp	1's comp (there is negative zero)
1100	1100
+ 1100	+ 1100
1100	1100
+ 0011	+ 0011
+ 1	1111
0100	− 0000
→ 1100	(negative zero.)
− 0100	
* 0000	
One arithmetic zero	Two Arithmetic zeroes

Binary Codes

Binary coded decimal ↓ diff from bin. codes.

n bits ⟶ Maxm of 2^n quantities

2^n quantities ⟶ Mumm of n bits.

Decimal Codes

	(BCD) 8 4 2 1	Excess 3	8 4 -2 -1	2 4 2 1	Biquinary code 5 0 4 3 2 1 0
0	0 0 0 0	0 0 1 1	0 0 0 0	0 0 0 0	0 1 0 0 0 0 1
1	0 0 0 1	0 1 0 0	0 1 1 1	0 0 0 1	0 0 0 0 0 1 1
2	0 0 1 0	0 1 0 1	0 1 1 0	0 0 1 0	0 0 0 0 1 0 1
3	0 0 1 1	0 1 1 0	0 1 0 1	0 0 1 1	0 0 0 1 0 0 1
4	0 1 0 0	0 1 1 1	0 1 0 0	0 1 0 0	0 0 1 0 0 0 1
5	0 1 0 1	1 0 0 0	1 0 1 1	1 0 1 1	1 0 0 0 0 0 1
6	0 1 1 0	1 0 0 1	1 0 1 0	1 1 0 0	1 0 0 0 0 1 0
7	0 1 1 1	1 0 1 0	1 0 0 1	1 1 0 1	1 0 0 0 1 0 0
8	1 0 0 0	1 0 1 1	1 0 0 0	1 1 1 0	1 0 0 1 0 0 0
9	1 0 0 1	1 1 0 0	1 1 1 1	1 1 1 1	1 0 1 0 0 0 0

ERROR CHECKING:

Never possible: codes like (1010, 1011, 1100 etc.) in BCD

Biquinary code - Self error detecting code.

There are weighted codes except excess 3.

	12
Binary conversion	1100
BCD	0001 0010

Biquinary code

0 1 0 0 0 0 1

0 1 0 0 0 1 0

0 1 0 0 1 0 0

0 1 0 1 0 0 0

0 1 1 0 0 0 0

1 0 0 0 0 0 1

1 0 0 0 0 1 0

1 0 0 0 1 0 0

1 0 0 1 0 0 0

1 0 1 0 0 0 0

Nos are represented in computers either in binary or in decimal using a binary code

9's complement of a decimal

395. Sub from 9

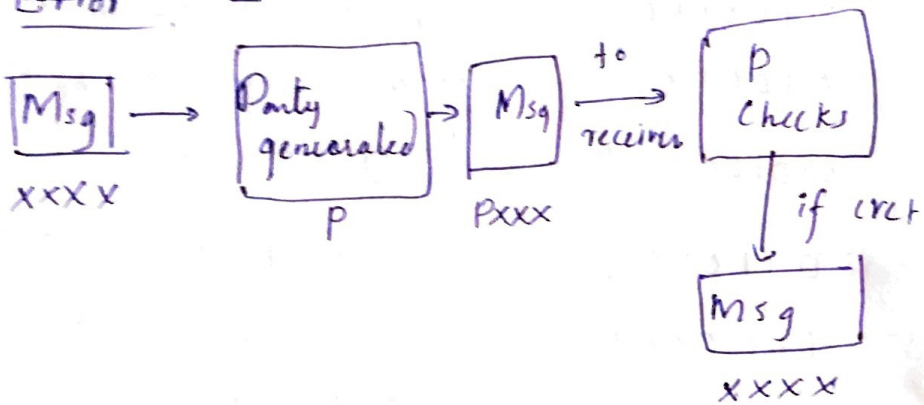
604

→ For the first 1, 1 will be present at 2nd location and for the last at 1st place

One 1 present in 2nd place, other 1 in any other place other than first place, (0-3)

→ One 1 present in 1st place, another 1 in any other place other than 2nd place. (5-9)

~~Error~~ Error Detection Codes



Parity < Odd / Even

~~Even~~/Odd Parity	Message	Correct/Not
1	1010	✓
1	1011	✗
1	1100	✓

Frequency of Error

P	P(odd)	P	P(even)
1	0000	0	0000
0	0001	1	0001
	. . .		
1	1001	0	1001

To make parity odd add a 1 if it is odd then 0

parity even →

Reflected Code / Gray Code (Not predefined)

0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100

One no and another differ by a single bit

Depends on starting value

8 1100

9 1101

10 1111

11 1110

12 1010

13 1011

14 1001

15 1000

ALPHANUMERIC CODES

6 bit internal code

8 bit EBCDIC Code
(Extended Binary Coded Decimal Interchange Code)

7 bit ASCII Code
(American Stand Code for Inf. Interchange)

BINARY LOGIC

Deals with variables that take on two distinct values and with operations that get the meaning True/False

→ Consisting of binary variables and logical operations.

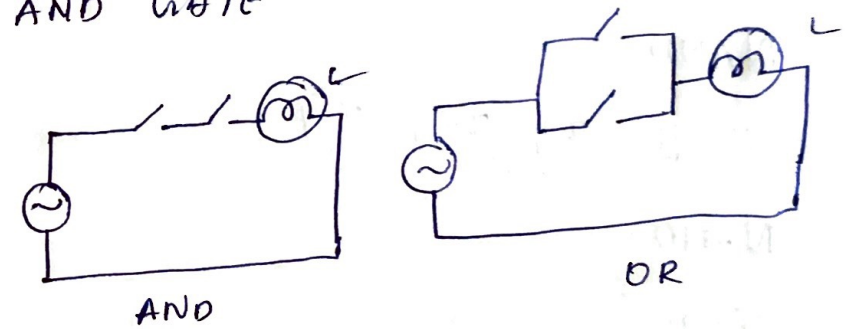
LOGIC GATES:

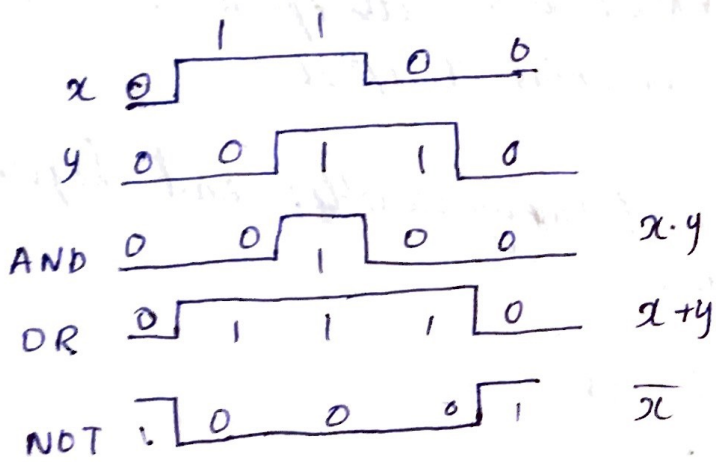
AND, OR } Min 2 inputs

NOT → single i/p – INVERTER

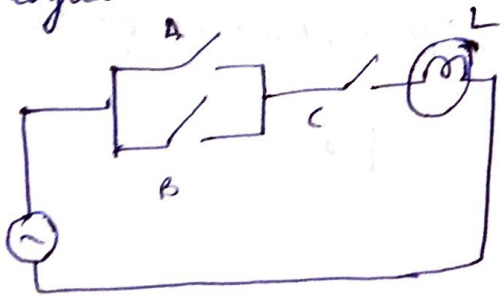
Truth table: All combination of i/p → o/p

→ AND GATE





→ Express the following switching circuit in logical notation.



$L = (A+B).C$

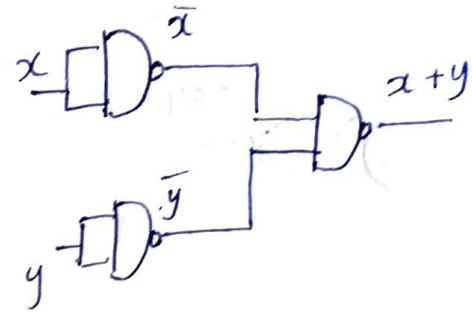
NAND

x	y	x̄	x+y	x·y

NAND.

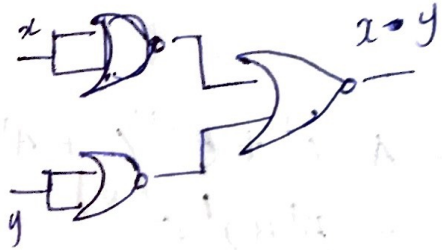
→ AND

OR.



NOR

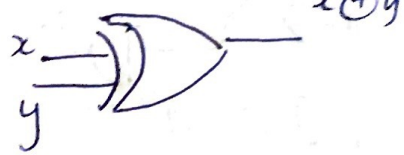
→ AND



NOT

EXCLUSIVE OR GATE

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

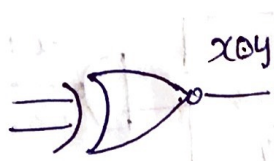


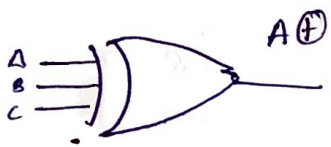
NOT

OR

Exclusive NOR gate

x	y	$x \odot y$
0	0	1
1	0	0
0	1	0
1	1	1





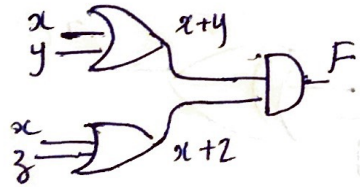
$A \oplus B = AB' + A'B$

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

$A \odot B = A'B' + AB$

Implement:

$F = [x+y][x+z]$



BOOLEAN ALGEBRA

- may be defined with a set of elements a set of operation and a no of axioms/ postulates

  set, $\in$, $\notin$, $\{\}$

- A binary operation defined in a set S of elements is a rule that assign to each pair of elements from S an unique element from S

1. Closure $a * b = c$ $\quad a, b, c \in S$

   set of natural numbers + addition.

   subtraction is not closed (-ve nos)

2. ASSOCIATIVE

   $(x * y) * z = x * (y * z)$ for all $x, y, z \in S$

3. Commutative

   $x * y = y * x$ for all $x, y \in S$

4. Identity element

   $e * x = x * e = x$ for every $x \in S$

5. Inverse → A set S having identity element w.r.t a binary operator ⊕ is said to have an inverse whenever, for every element $x \in S$, there exists an element such that; $x \oplus y = e$

e.g. Set of integers wrt addition, the inverse of a is,

$$a + (-a) = I = 0$$

6. Distributive law,

$$x * (y \cdot z) = (x * y) \cdot (x * z)$$

Multiplication and addition

- Algebraic structure

↓

- Field - is a set of elements together with two binary operations each having the properties 1 to 5 and both operations combined to give property 6.

Field of real nos (set of real nos with two binary operations + and ·)

Binary operator + defines addition.
The additive identity is 0.
The additive inverse defines subtraction
Binary operator defines multiplication
The multiplication identity is 1
The multiplication inverse defines division.
The only distributive law is · over +

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

- Boolean algebra

is an algebraic structure defined on a set of elements B together with two binary operators + and ·, provided the following Huntington postulates are satisfied

1. a) closure wrt the operator +
   b) Closure wrt the operator ·
2. a) An identity element wrt +,

designated by 0

$x+0 = 0+x = x$

(b) An identity element wrt $\cdot$, designated by 1,

$x \cdot 1 = 1 \cdot x = x$

(c)

3. a) Commutative wrt $+$: $x+y = y+x$

b) Commutative wrt $\cdot$: $x \cdot y = y \cdot x$

4. a) $\cdot$ is distributive over $+$:

$x(y+z) = (x \cdot y) + (x \cdot z)$

b) $+$ is distributive over $\cdot$:

$x + (y \cdot z) = (x+y) \cdot (x+z)$

5. For every element $x \in B$, there exists an element $x' \in B$ (complement of x) such that (a) $x + x' = 1$ and

(b) $x \cdot x' = 0$

6. There exists atleast 2 elements $x, y \in B$ such that $x \neq y$

Comparison:

1. H_P doesn't have associative law
2. The distributive property of $+$ over $\cdot$, is valid
3. B.A doesn't have an addition / multiplicative inverse. Therefore there is no subtraction / division
4. Operation complement is not available in ordinary algebra
5. ~~boolean algebra~~ Ordinary algebra deals with real nos
6. B.A depend with only 2 nos 0 & 1

B.A should satisfy:

1. The elements of the set B
2. The rules of operation for the two binary operations
3. That set of elements B, together with

the two operation satisfies the 6 huntington postulates

1. Closure
2. Identity
3. Commutative
4. Distributive
5. Complement
6. 2 distinct elements 0 & 1 and $0 \neq 1$

Basic Theorems & Properties Of Boolean Algebra

1. Duality Principle

One ~~part~~ can be obtained

The dual of an expression can be obtained by interchanging the operator & identity elements [or Huntington postulate]

Basic Theorem

(a) $x+0 = x$ (b) $x \cdot 1 = x$

(I) (a) $x + x' = 1$ (b) $x \cdot x' = 0$

(II) (a) $x + x = x$ (b) $x \cdot x = x$

(III) (a) $x + 1 = 1$ (b) $x \cdot 0 = 0$

(IV) Involution Theorem on $(x')' = x$

(V) Comm (a) $x+y = y+x$ (b) $x \cdot y = y \cdot x$

(VI) Assoc (a) $x+(y+z) = (x+y)+z$ (b) $x \cdot (yz) = (xy) \cdot z$

(VII) Distributive theorem

(a) $x(y+z) = xy + xz$

(b) $x + (yz) = (x+y) \cdot (x+z)$

(IX) a) $(x+y)' = x' \cdot y'$ [De Morgans Theorem]

b) $(x \cdot y)' = x' + y'$

(X) (a) $x + xy = x$ [Absorption Theorem]

(b) $x \cdot (x+y) = x$

- Prove $x + x = x$

$\Rightarrow (x+x) \cdot 1$

$\Rightarrow (x+x)(x+\bar{x})$

$\Rightarrow x + x\cdot\bar{x}$

$\Rightarrow x \qquad [x\cdot\bar{x} = 0]$

2. $x \cdot x = x$

$\Rightarrow (x \cdot x) + 1$

$\Rightarrow (x+1)\cdot x + (x\cdot x) + (x + \bar{x})$

$\Rightarrow$

PROOF :

- ⇒ By basic theorem
- ⇒ By truth table
- ⇒ By Venn diagram

Boolean function

$F = x + y$

A binary variable can have a value of 0/1

A boolean function is an expression formed with Binary variables, the two binary operators OR and AND, the unary operator NOT, parenthesis and equal sign.

eg: 'xyz' $F_1 = xyz'$

$F_2 = x + y'z$

$F_3 = x'y'z + x'yz + xy'$

$F_4 = xy' + x'z$

x	y	z	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

$x'y'z + x'yz + xy' = x'z[y' + y] + xy'$

$= x'z + xy'$

Minimized function

→ fewer no. of gates

→ fewer no. of i/ps

Algebraic Manipulation.

Literal → a primed / unprimed variable [complement]

When a boolean function is implemented with logic gates, each literal in the function designates input to gate and each term is implemented with a gate.

⇒ Simplify the following boolean functions to a minm no. of literals.

1. $x + x'y \rightarrow (x+x')\cdot(x+y)$

DUAL ⇑⇓

$\Rightarrow 1 \cdot (x+y) \Rightarrow x+y$

2. $x(x'+y) \Rightarrow xx' + xy \Rightarrow xy$

3. $x'y'z \; x'yz + xy \Rightarrow x'z(y'+y) + xy$

$\Rightarrow x'z + xy$

4. $xy + x'z + yz \Rightarrow xy + z(x'+y)$

$\Rightarrow xy + (x+z)(y+x')$

$\Rightarrow xy + x'z + yz(x+x')$

$\Rightarrow xy + x'z + xyz + yzx'$

$\Rightarrow xy(1+z) + x'z(1+y)$

$\Rightarrow xy + x'z \Rightarrow (x+y)\cdot(x'+z)$

5. $(x+y)(x'+z)(y+z) \Rightarrow (x+yz)(x+y)\cdot(x'+yz)$

$\Rightarrow xy + x'z$

Complement Of A Function

Complement of F is F' and is obtained by interchange of 0's by 1's and 1's by 0's in the value of F

$F = x'y + xy$

$F' = x'y' + xy'$

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

$= (x'y)' \cdot (xy)'$ By taking dual

$= (x+y') \cdot (x'+y') \Leftarrow xy' + x'y'$

$= xx' + xy' + y'x' + y'y'$

$= xy' + x'y'$ ✱

DEMORGANS THEOREM

$(A+B)' = A' \cdot B'$

$(A+B+C)' = A'.B'.C'$.

GENERALIZED FORM OF DEMORGAN THEOREM

- Complement of a fn is obtained by interchanging AND and OR operators and complementing each literal

eg. $F_1 = x'y'z' + x'yz$

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

$= x'[(y+z')(y'+z)] \cdot x + [y+z)(y'+z']$

Take dual and complement each literal

$\Rightarrow$ ~~$[x+y'+z'] \cdot (x'+y+z')$~~

$F_1 = x'.yz' + x'y'z$

$= (x'+y+z') \cdot (x'+y'+z)$

$F_2 = x(y'z' + yz)$

$= x + (y'+z') \cdot (y+z)$

Simplify the foll bool. form to a minm no. of literals

[a] $xy + xy'$

$x(y+y') = x$

[b] $(x+y) \cdot (x+y')$

$= x$

[c] $xyz + x'y + xyz'$

$xy[z+z'] + x'y$

$= xy + x'y$

$= y[x+x']$

$= y$

[d] $xz + x'yz$

$z[x+x'y]$

$= z[xx' + xy]$

$= xyz$

$= z[(x+x') \cdot (x+y)]$

$= [x+y]z$

[e] ~~A+~~

[e] $(A+B)' \cdot (A'+B')'$

$= (A'.B') \cdot (A.B)$

$= 0$

(f) $y[wz' + wz] + xy$

$y[wz' + wz + x]$

$= y[w+x]$

- Find the complement of the Boolean function and reduce them to a minm no. of literals.

a) $(BC' + A'D)(AB' + CD')$

$AB'BC' + BC' = AB'BC' + BC'CD' + A'DAB' + A'DCD'$

$= D' = 1$

b) $B'D + A'BC' + ACD + A'BC$

$= D(AC + B') = ACD + B'D$

$= D' + (A'C' \cdot B) = A'C'D' + BD'$

c) $B'D + A'BC'$

$= B'D'$

$= B'D + ACD + A'BC(C + C')$

$= B'D + ACD + A'BC + A'BC'$

$= (B + D')(A + B')(A' + C' + D')$

$= (AB + AD' + B'D')(A' + C' + D')$

d) $= ABC' + ABD' + AB'C' + B'D'A'$

$+ B'D'C'$

$=$

c) $-[(AB)'A][(A.B)'B]$

$(A.B)' = 1$

d) $AB' + C'D'$

$(A' + B)(C' + D)$

Obtain the truthtable of the following.

$\rightarrow F = xy + xy' + y'z$

$= x + y'z$

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

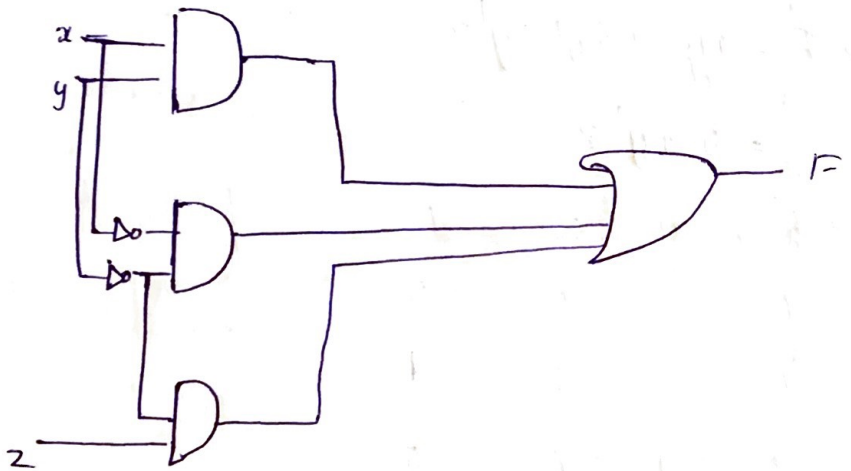
- $F = xy + x'y' + y'z$ $\quad (x'+y')(x+y)[y+z']$

a] Implement with AND OR & NOT gates

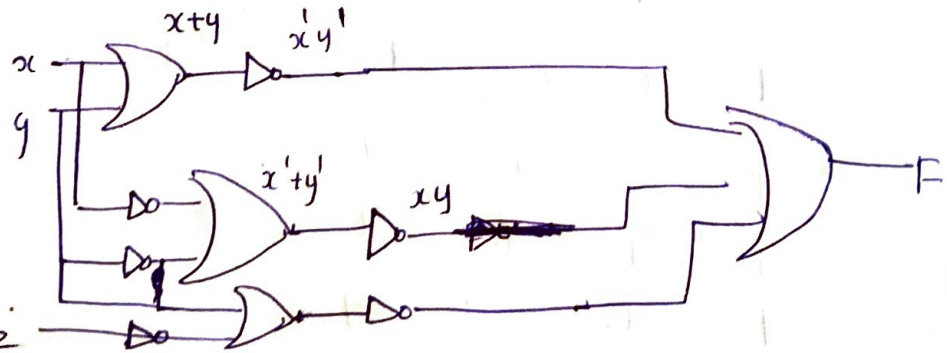
b] Implement it with only OR & NOT gats

c] Implement it with only AND & NOT gates

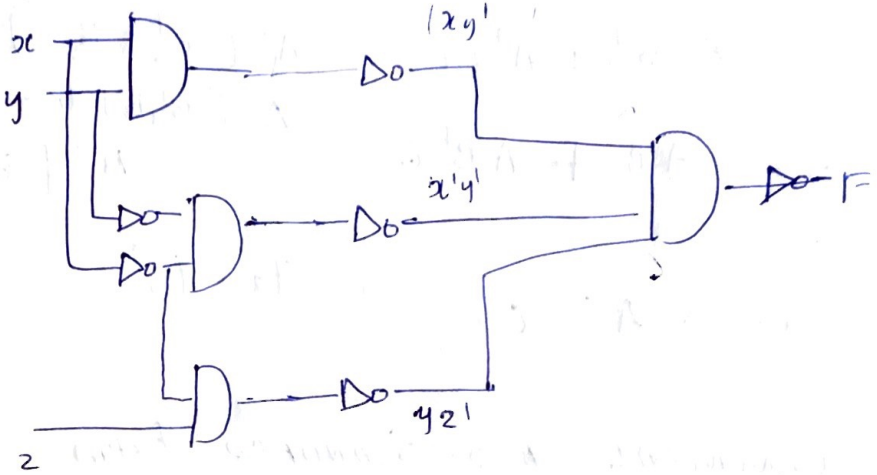
a]



b]



c]



→ Simplify the fns T_1 & T_2 to min no. of literal

A	B	C	T_1	T_2
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

$T_1 = A'B'C' + A'B'C + A'BC'$

$A'B'(C+C') + A'BC'$

$= A'B' + A'BC'$ $= A'[B' + BC']$

$= A'[(B'+B)(B'+C')]$

$= A'[B'+C']$

~~$T_2 = AB + AB'C$~~

$T_2 = T_1'$

$T_2 = A + BC$

CANONICAL AND STANDARD FORMS

CANONICAL
- Min term → standard product
- Max term → standard sum

In a canonical form all the variables should be there.

Standard form

eg $A + BC$

[sum of products)

Non standard form

$A'(B+C')$

(Cannot be saved like that)

			Min term		Max term	
x	y	z	Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x+y+z$	M_0
0	0	1	$x'y'z$	m_1	$x+y+z'$	M_1
0	1	0	$x'yz'$	m_2	$x+y'+z$	M_2
0	1	1	$x'yz$	m_3	$x+y'+z'$	M_3
1	0	0	$xy'z'$	m_4	$x'+y+z$	M_4
1	0	1	$xy'z$	m_5	$x'+y+z'$	M_5
1	1	0	xyz'	m_6	$x'+y'+z$	M_6
1	1	1	xyz	m_7	$x'+y'+z'$	M_7

$(m_0)' = M_0$

x	y	z	F_1	F_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	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

$F_1 = x'y'z + xy'z' + xyz$

$\Rightarrow m_1 + m_4 + m_7$

$F_1(x,y,z) \Rightarrow \sum(1,4,7)$

$F_2 = m_3 + m_5 + m_6 + m_7$

$F_2(x,y,z) = \sum(3,5,6,7)$

$(F_1)' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$

$(F_1')' = F_1$

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

$(x'+y'+z)$

$= M_0 \cdot M_2 \cdot M_3 \cdot M_5 M_6$

$= \Pi(0,2,3,5,6)$

$F_2' = \sum(3,5,6,7)$

$F_2 = \Pi(0,1,2,4)$

→ Boolean fun expressed as a sum of minterm or product of maxterm are said to be in canonical form

SUM OF MINTERMS :

Express the boolean fun $F = A + B'C$ in a sum of minterm form

$F = ABC + ABC' + AB'C + AB'C' + A'B'C$

$F = A = \sum(1,4,5,6,7)$

$A = A[B+B']$

$\Rightarrow AB + AB'$

$\Rightarrow AB(C+C') + AB'(C+C')$

$\Rightarrow ABC + ABC' + AB'C + AB'C'$

$B'C \Rightarrow (A+A')B'C$

$\Rightarrow AB'C + A'B'C$

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

- Express the boolean fn $F = xy + x'z$ in a product of maxterm form

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$\Pi [0, 2, 4, 5)$

Maxterm - 0

Minterm - 1

Conversion Between Canonical Form

Complement of a function expressed as the sum of minterm equals the sum of minterm missing from original term

Eg: $F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$

$F'(A, B, C) = \Sigma(0, 2, 3) = m_0 + m_2 + m_3$

$F(A, B, C) = (m_0 + m_2 + m_3)'$

$= m_0' \cdot m_2' \cdot m_3'$

$= M_0 \cdot M_2 \cdot M_3$

$= \Pi(0, 2, 3)$

$F(x, y, z) = \Pi[0, 2, 4, 5)$

$\Downarrow$

$F(x, y, z) = \Sigma(1, 3, 6, 7)$

STANDARD FORMS — sum of products, product of sums

$F_1 = y' + xy + x'y'z'$

$F_2 = x(y' + z)(x' + y + z' + w)$

NON-STANDARD FORM :-

$F_3 = (AB + CD)(A'B' + C'D')$

$\Downarrow$

$F_3 = ABC'D' + A'B'CD$ (standard form)

- Express following in sum of min terms and product of maxterm

a) $F(A, B, C, D) = D(A' + B) + B'D$

$= A'D + BD + B'D$

$F = \sum(1,3,5,7,9,11,13,15)$

$F = \pi(0,2,4,6,8,10,12,14)$.

6) $F(A,B,C,D) = (A+B'+C)(A+B')(A+C'+D')$
$(A'+B+C+B')(B+C'+D')$

$= (A+B')($

$= AA+AB'+AB'$

$F = \sum 0,1,2,8,10,12,13,14,15)$
$\pi(3,4,5,6,7,9,11)$

- $F(x,y,z) = 1$

$F(x,y,z) = \sum(0,1,2,3,4,5,6,7)$

No max term

- $F(x,y,z) = (xy+z)(y+xz)$ $\quad xy = xy(z+z')$

$= xyy + xyyz + yz + xz$

$= xy + xyz + yz + xz$

$\Sigma(3,5,6,7)\ \pi(0,1,2,4)$

SIMPLIFICATION OF BOOLEAN FUNCTION

- MAP METHOD - VEITCH - KARNAUGH

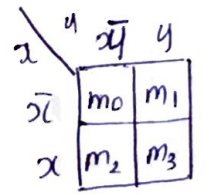
KARNAUGH MAP / K-MAP:

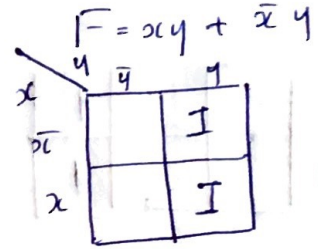
Adjacent box differ by 1 bit.

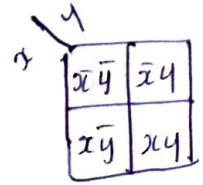
Total no. of boxes = 2^n (n - variable)

Make a combination power of 2 no. of boxes and produce the output

2 variable Kmap:

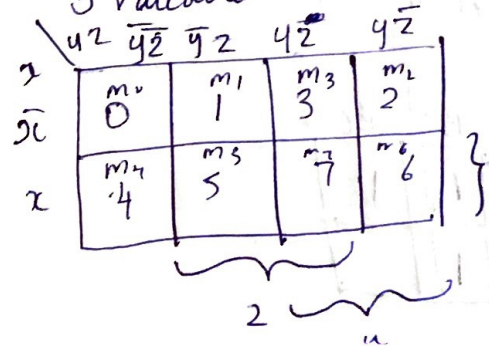


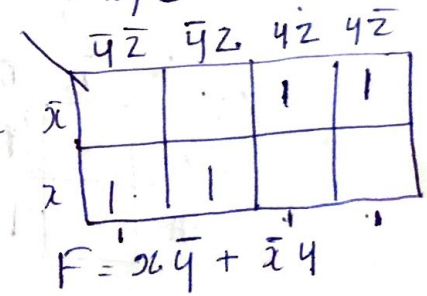




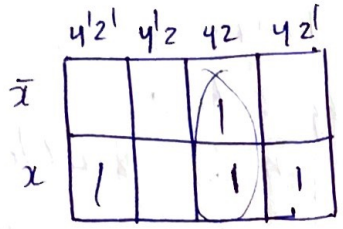
$F = y$

3 variable K-MAP



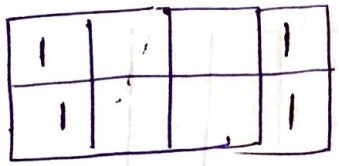


$F = x'yz + xy'z' + xyz + xyz'$



$F = yz + x\bar{z}$

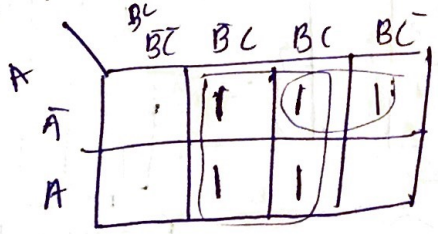
$F = x'y'z' + x'yz' + xy'z' + xyz'$



$F = y'z' + yz'$

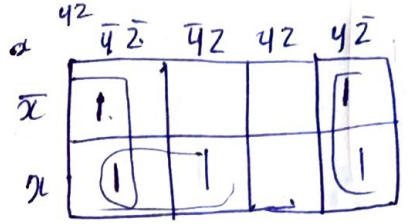
z'

$\Rightarrow F = A'C + A'B + AB'C + BC.$



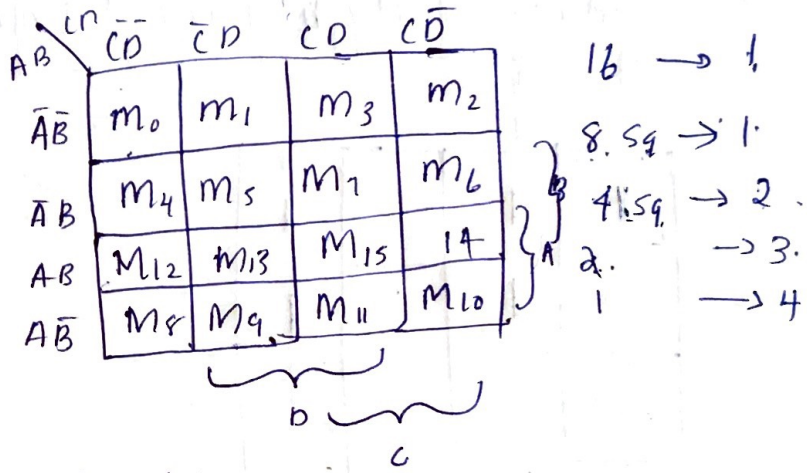
$F = C + \bar{A}B$

$\Rightarrow F(x,y,z) = \Sigma(0,2,4,5,6)$



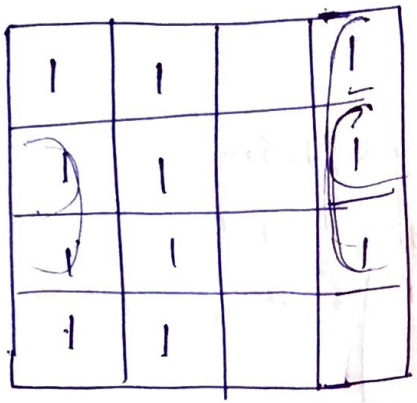
$F = \bar{z} + x\bar{y}$

$\Rightarrow$ 4 - Variable K-map



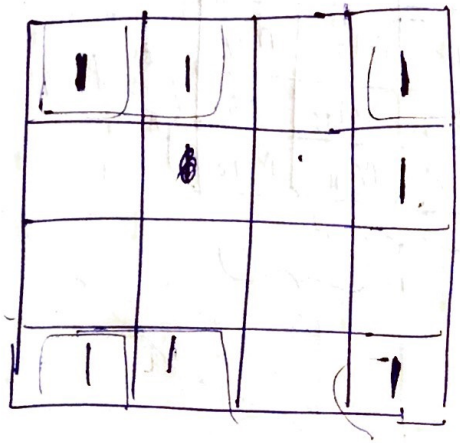
eg: Simplify.

$F(w,x,y,z) = \Sigma(0,1,2,4,5,6,8,9,12,13,14)$



$F = \bar{y} + \bar{x}\bar{y} + wxz + w'z$

2) $F = A'B'C' + B'CD' + A'BCD' + AB'C'$



$F = B'D' + B'C' + A'CD'$

FIVE VARIABLES K-MAP

AB \ CDE	000	001	011	010	110	111	101	100
00	0	1	2	3	6	7	5	4
01	8	9	11	10	14	15	13	12
11	24	25	27	26	30	31	29	28
10	16	17	19	18	22	23	21	20

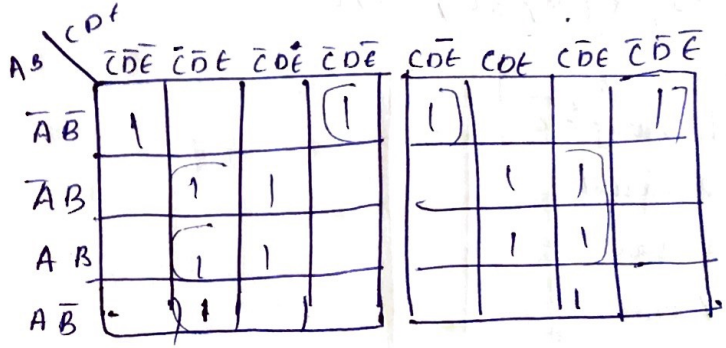
MSB

00 01 11 10 10 11 01 00

Add MSB 0 Add MSB 1

000 001 011 010 110 111 101 100

Simplify $(A,B,C,D,E) = \{0,2,4,8,9,11,13,15,17,21,25,27,29\}$



$F = BE + A\bar{D}E + \bar{A}\bar{B}\bar{E}$

SIX VARIABLE K - MAP

ABC \ DEF	000	001	011	010	110	111	101	100
000	0	1	3	2	6	7	5	4
001	8	9	11	10	14	15	13	12
011	24	25	27	26	30	31	29	28
010	16	17	19	18	22	23	21	20
110	48	49	51	50	54	55	53	52
111	56	57	59	58	62	63	61	60
101	40	41	43	42	46	47	45	44
100	32	33	35	34	38	39	37	36

31 ⟶ 15, 23, 24, 30, 27, 63

64 ⟶ 1

32 ⟶ 1 variable

16 ⟶ 2 variab

8 ⟶ 3 var

4 ⟶ 4 va

2 ⟶ 5 var

1 ⟶ 6 var

Product of sum simplification

eg: Simplify the following boolean function

100 (a) SOP & (b) POS

$F(A,B,C,D) = \Sigma(0,1,2,5,8,9,10)$

(a) F =

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

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

b) $F = (C'+D')(C'+A'+B')(B'+D)$

$F' = AB + CD + B\bar{D}$

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1

(a) F =

(b) F' =

1 1 1 0

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

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

$= \bar{x}z + xz\bar{z}$

(a) $F = x'z + zz'$

(b) $F' = xz + x'z'$

$f = (x'+z')(x+z)$

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

(0)

(3) B = $F = (A'+B'+C)(B+D)$

$F' = ABC' + B'D'$

Mark 0's and mark 1 in all remain boxes.

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

$F = \bar{A}B + BC + \bar{B}D$

$F = \bar{B}\bar{D} + AB\bar{C}$

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

→ Simplify

a) $F(x,y,z) = \Sigma(2,3,6,7)$

b) $F(A,B,C,D) = \Sigma(7,13,14,15)$

c) $F(A,B,C,D) = \Sigma(4,6,7,15)$

1 Obtain the simplified expression in POS

(i) $F(x,y,z) = \Pi(0,1,4,5)$

(ii) $F(A,B,C,D) = \Pi(0,1,2,3;4)$

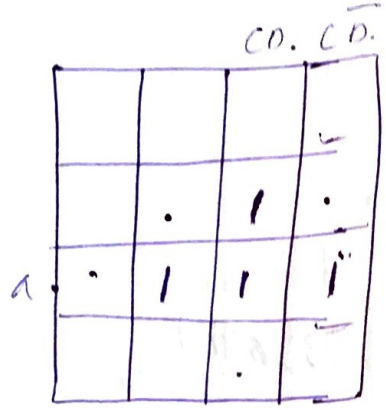
(iii) $F(w,x,y,z) = \Pi(1,3,5,7,15,13)$

a)

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

$F(x,y,z) = y$

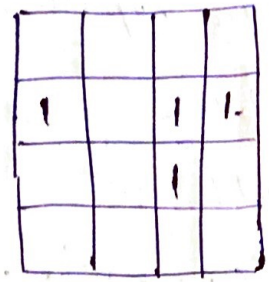
b)



$F = A\bar{C} + ABD + \bar{B}CD$

$= AB(\bar{C}+D) + ACD$

c)

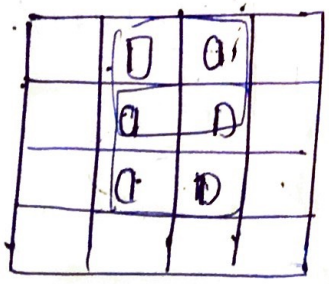


$F = BCD + \bar{A}B\bar{D}$

(i) $F = 4$

(ii) $F = (B+\bar{C}\,|\,(A+B)(A+C+b)$

(iii)



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

$= BD + \bar{A}BD$

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

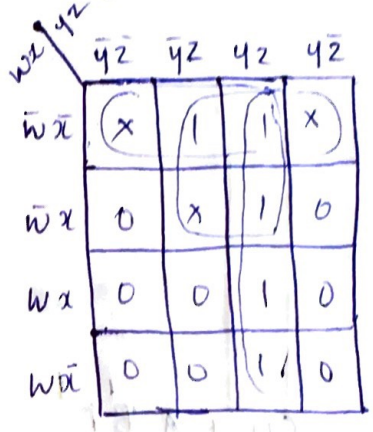
DON'T CARE CONDITION

~~BCD → 4 bit~~

- Simplify the boolean form

$F(w,x,y,z) = \Sigma(1,3,7,11,15)$

$d(w,x,y,z) = \Sigma(0,2,5)$.



Combining 1's → $F(w,x,y,z) = yz + \bar{w}z$

OR $F = yz + \bar{w}\bar{x}$

Combining 0's → $\bar{z} + w\bar{y}$

$F = z(\bar{w}+y)$

Don't care condition is used if it goes for a large combination.

2) $F = (y' + x'z')$

$d = yz + xy$

3) $F = B'C'D' + BCD' + ABCD'$

$d = B'CD' + A'BC'D$

a)

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

$F = 1$

b)

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

$F = C\bar{D} + A'B'D'$

Simplify (a) in SOP and (b) in POS

a) $F = A'B'D' + A'CD + A'BC$

$d = A'BC'D + ACD + AB'D'$

b) $F = W'(x'y + x'y' + xyz) + x'z'(y + w)$

$d = W'x(y'z + yz') + Wyz$

(c) $F = ACE + A'CD'E' + A'C'DE$

$d = DE' + A'B'E + AD'E'$

a)

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

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

$F' = A + C\bar{D} + B\bar{C}$

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

b)

wx \ yz	$y\bar{z}$	$\bar{y}z$	yz	$y\bar{z}$
$\bar{w}\bar{x}$	1	1	1	1
$\bar{w}x$	0	X	1	X
wx	0	0	X	0
$w\bar{x}$	1	0	X	1

$F = \bar{w}\bar{z} + \bar{x}\bar{z}$

$F = (w' + z')(x' + z)$

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

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

$f = (w' + z')(x' + z)$

c)

AC \ DE	$D\bar{E}$	$\bar{D}\bar{E}$	DE	$D\bar{E}$
$\bar{A}\bar{C}$	0	X	1	X
$\bar{A}C$	1	X	0	X
AC	X	1	1	X
$A\bar{C}$	X	0	0	X

$F = C\bar{D} + AC + A'C'E$

$F' = A\bar{C} + C'E' + A'C\bar{E}$

$= (A' + C)(C + E)(A + C' + E')$

The following boolean expression $BE + B'D_{E'}$ is a simplified ~~expression of~~ version of expression

$A'BE + BC\bar{D}E + BC'D'E + A'B'\bar{D}E' + B'C'D_{E'}$

Are there any dont care conditions? If so what are they?

$F = \bar{A}BE + A\bar{B}C\bar{D}\bar{E}$

$+ ABC\bar{D}E + ABCD\bar{E} + A\bar{B}CD\bar{E}$

THE TABULATION METHOD / QUINE, MCCLUSKY METHOD

→ Prime implicants

1. Finding the prime implicants
2. Determination of essential prime implicants

Simplify $F = \Sigma(0,1,2,8,10,11,14,15)$

1. Determination of prime implicants.

(a) (b)

W	x	y	z
0	0	0	0
1	0	0	1
2	0	0	1	0
8	1	0	0	0
10	1	0	1	0
11	1	0	1	1
14	1	1	1	0
15	1	1	1	1

(Based on no. of 1's)

	W	x	y	z	
(0,1)	0	0	0	–	
(0,2)	0	0	–	0	
(0,8)	–	0	0	0	
(2,10)	–	0	+	0	
(8,10)	1	0	–	0	
(10,11)	1	0	1	–	
(10,14)	1	–	1	0	
(11,15)	1	–	1	1	
(14,15)	1	1	1	–	

Powers of 2

(c)

W	x	y	z	
–	0	–	0	(0,2,8,10)
–	0	–	0	(0,8,2,10)
1	–	1	–	(10,14,11,15)
1	–	1	–	(10,14,11,15)

$F = w'x'y' + x'z' + wxy$

(a)

0
1
2
8
10
11
14
15

(b)

0,1 (1)

0,2 (2) ✓

0,8 (8) ✓

(2,10) (8) ✓

(8,10) (2) ✓

10,11 (1) *

10,14 (4) ✓

11,15 (4) ✓

14,15 (1) *

(c)

0,2,8,10 (2,8)

0,8,2,10 (2,8)

10,11,14,15 (1,4)

10,14,11,15 (1,4)

0 0 0	(0,1)
0 0 0 1	

$w'x'y' + x'z' + wy$

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

(0,2,8,10)

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

(10,11,14,15)

SIMPLIFY $F(wxyz) = \Sigma(1,4,6,7,8,9,10,11,15)$

	w	x	y	z
1	0	0	0	1
4				
8				
6				

(b)

6 * (1,9) (8)

9 * (4,6) (2)

(8,9) (1) ✓

(8,10) (2) ✓

6,7 (1)

9,11 (2) ✓

10,11 (1) ✓

7,15 (8)

11,15 (4)

(c)

8,9,10,11 (1,2)

(8,10,9,11) (1,2)

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

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

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

1	0 0 0
	1 0 1 1

Prime implicants

$x'y'z$, $w'xz'$, $w'xy$

xyz, wyz, wx'

PRIME IMPLICANT TABLE

	1	4	6	7	8	9	10	11	15
x'y'z(1,9)	*								
✓ w'xz'(4,6)		*	*						
✗ w'xy(6,7)				*					
✓ xyz(7,15)				*					*
✗ wyz(11,15)									*
✓ wx'(8,9,10,11)					*	*	*	*	

F = x'y'z + w'xz' + wx' + xyz

→ Simplify using tabulation

a) $F(A,B,C,D,E,F,G) = \sum(10, 24, 52, 60)$

(a)	(b)	
0010100	001-100 (8)	0-1-100
0011100	0-1 0100 (32)	0-1-100
0110100	0-11100 (32)	
0111100	011-100 (8)	

a) $F(A,B,C,D,E,F,G) = \sum(20, 28, 38, 39, 52, 60, 102, 103, 127)$

⇒ ABCDEFG + A'CEF'G' + BC'DEF

DONT CARE CONDITION

↳ Included in the determination of prime implicants

↳ Not included in setting up the prime implicant table

COMBINATIONAL CIRCUITS

DESIGN PROCEDURE

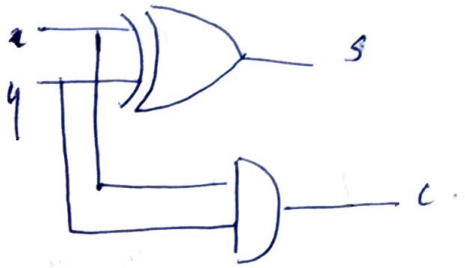
1. Problem statement
2. No of i/p and no. of o/p's to i/p & o/p
3. Give some symbols and names to i/p & o/p
4. Find relationships b/w i/p and o/p by using a truth table
5. Derive expression for o/p variable wrt i/p variable
6. Simplify expression
7. Draw logic design.

ADDERS:

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

$S = x'y + xy'$

$C = xy$



FULL ADDER

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

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

$C = x'yz + xy'z + xyz' + xyz$

$S = x \oplus y \oplus z$

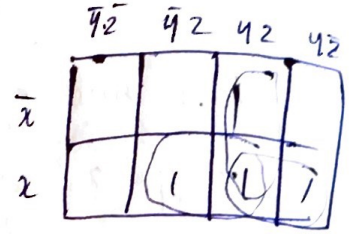
$= (x'y + y'x) \oplus z = (x'y + xy')'z + (x'y + xy')z'$

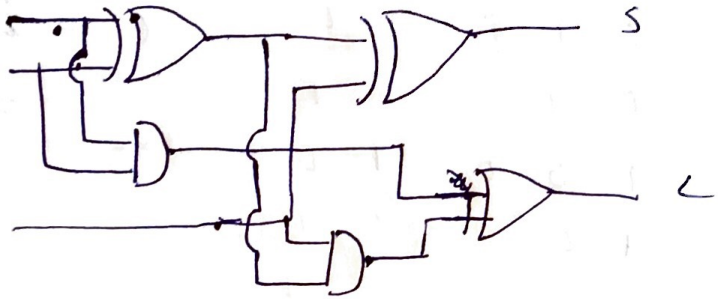
$= x'y'z + x'yz + xy'z' + xyz'$

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

$= xyz + x'y'z + x'y z' + xy' z'$.

$C = yz + xz + xy$





2 HA + 1 OR

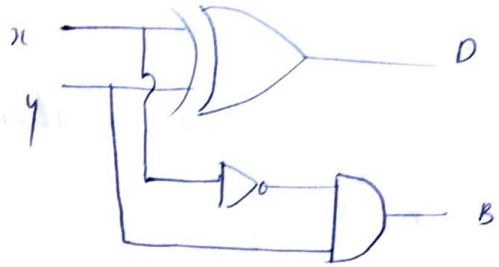
SUBTRACTOR:

→ HS

x	y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$D = x'y + xy'$

$B = x'y$

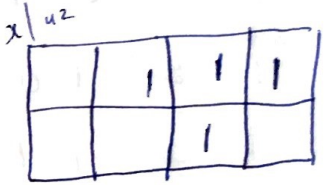


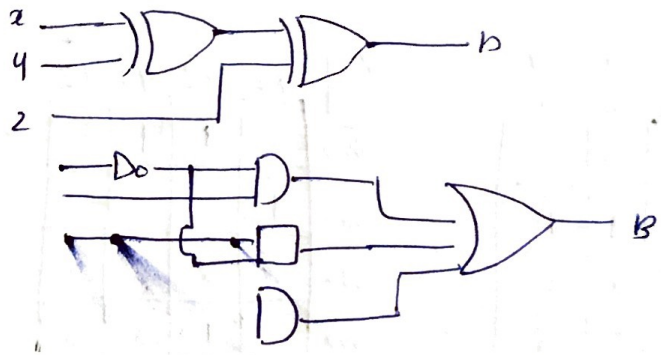
FULL SUBSTRACTOR

x	y	z	D	B
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

$D =$

$B = x'z + x'y + yz$





Code Converters

- Design a BCD to excess 3 code converter.

BCD				Ex-3			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				

$\bar{A}B\bar{C}D$

W

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

Y

WX \ YZ	$\bar{Y}\bar{Z}$	$\bar{Y}Z$	YZ	$Y\bar{Z}$
$\bar{W}\bar{X}$		1	1	1
$\bar{W}X$	1			
WX	X	X	X	X
$W\bar{X}$		1	X	X

$W = A + BD + BC$

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

Y

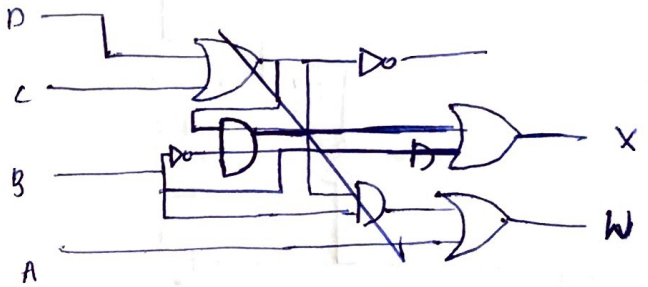
AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1		1	
$\bar{A}B$	1		1	
AB	X	X	X	X
$A\bar{B}$	1		X	X

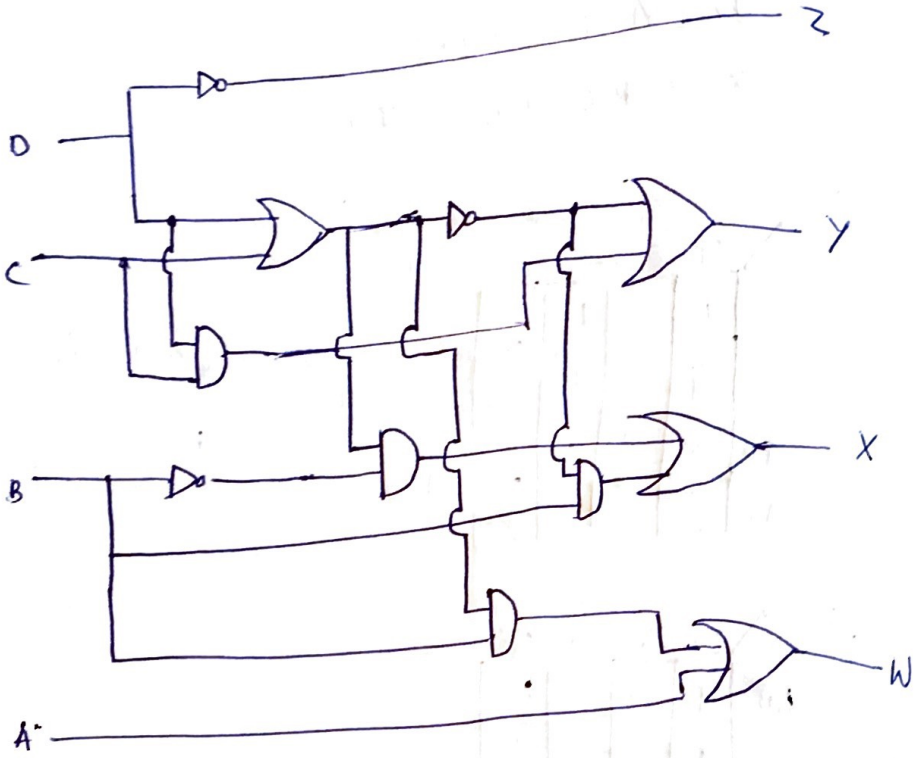
$Y = CD + \bar{C}\bar{D}$

$Z = \bar{D}$

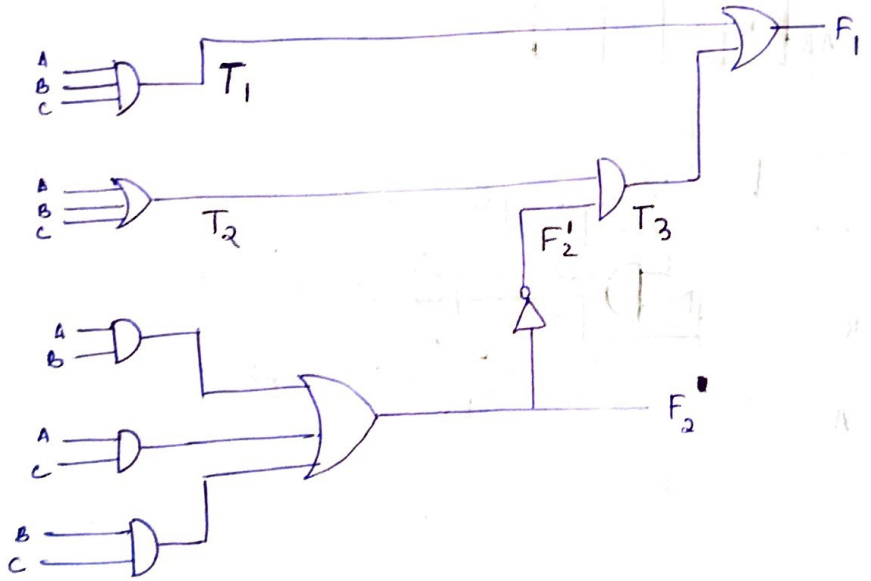
2

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





Analysis procedure



$F_2 = AB + AC + BC$ $T_1 = ABC$ $T_2 = A+B+C$

$T_3 = T_2 \cdot F_2'$

$T_3 = (A+B+C)(AB + AC + BC)'$

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

$F_1 = T_1 + T_3$ $= (AB + \bar{A}B + \bar{A}C + \bar{B}C)(\bar{A}\bar{B} + \bar{A}\bar{C} + \bar{B}\bar{C} + \bar{C})$

$\Rightarrow ABC + T_3$

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

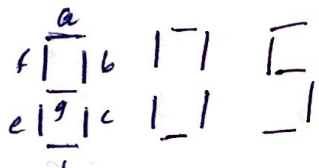
$F_1 = T_1 + T_3$

$= ABC + T_3$

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

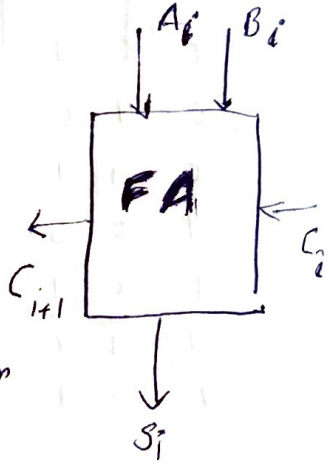
A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A bcd to seven segment decoder is a combinational circuit that accepts a decimal ~~shift~~ digit in BCD and generates the appropriate outputs for selection of segments in a display indicator used for displaying the decimal digits. Design a BCD to seven segment decoder circuit.



W	X	Y	Z	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0

BINARY PARALLEL ADDER



I/P carry 1 0

Augend 1 0 1 1

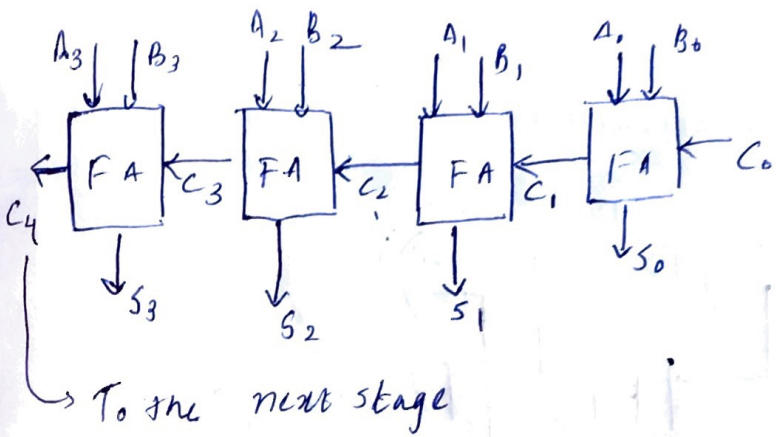
Addend 0 0 1 1

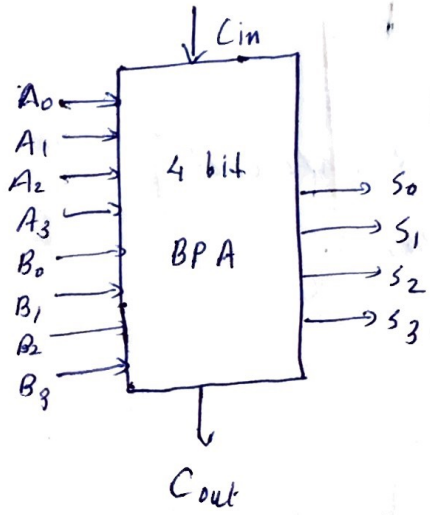
Sum 1 1 1 0 → final sum

O/P Carry 0 0 1 1

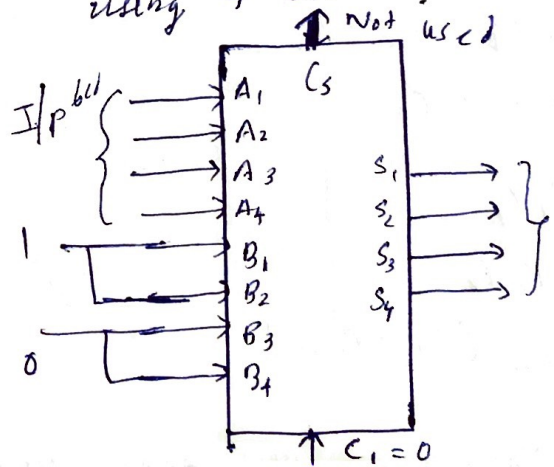
↓

final O/P carry





Eg. Design a BCD to excess 3 code convertor using 4 bit · BPA ?



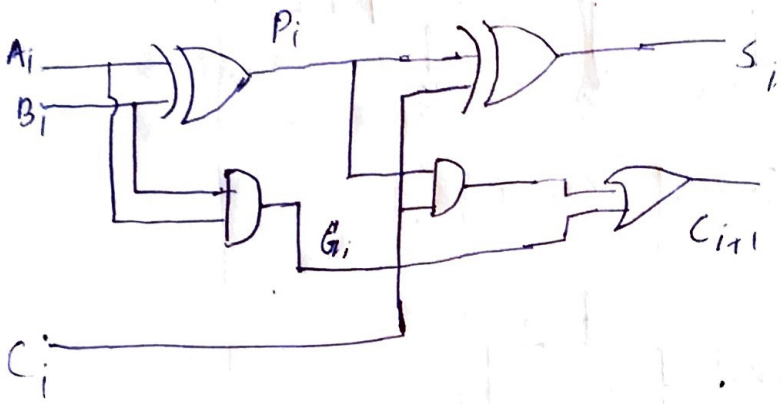
~~O/P Ex-3~~

2 gate level

$2 \times 4 = 8$

128 gate level

CARRY PROPOGATION



Output carry - 2 gate level.

Carry propogation $\triangleq$

Carry generation

Look ahead carry

$P_i = A_i \oplus B_i$

$G_i = A_i \cdot B_i$

O/P sum $= P_i \oplus C_i$

O/P carry $= G_i + P_i C_i$

$C_2 = G_1 + P_1 C_1$

$C_3 = G_2 + P_2 C_2$

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

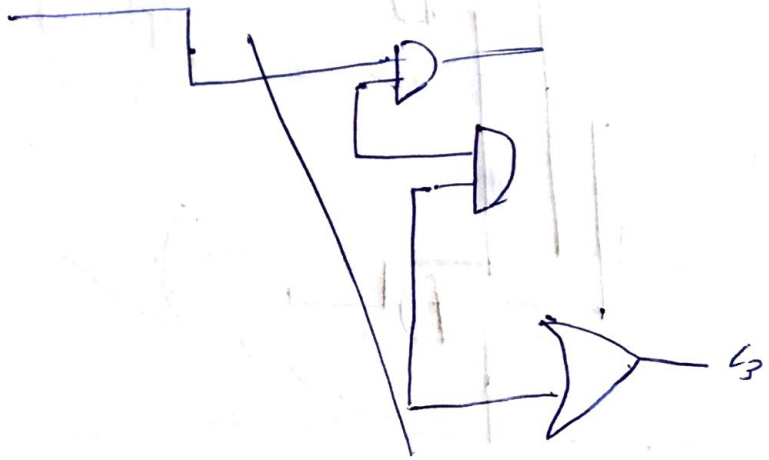
$= G_2 + P_2 G_1 + P_2 P_1 C_1$

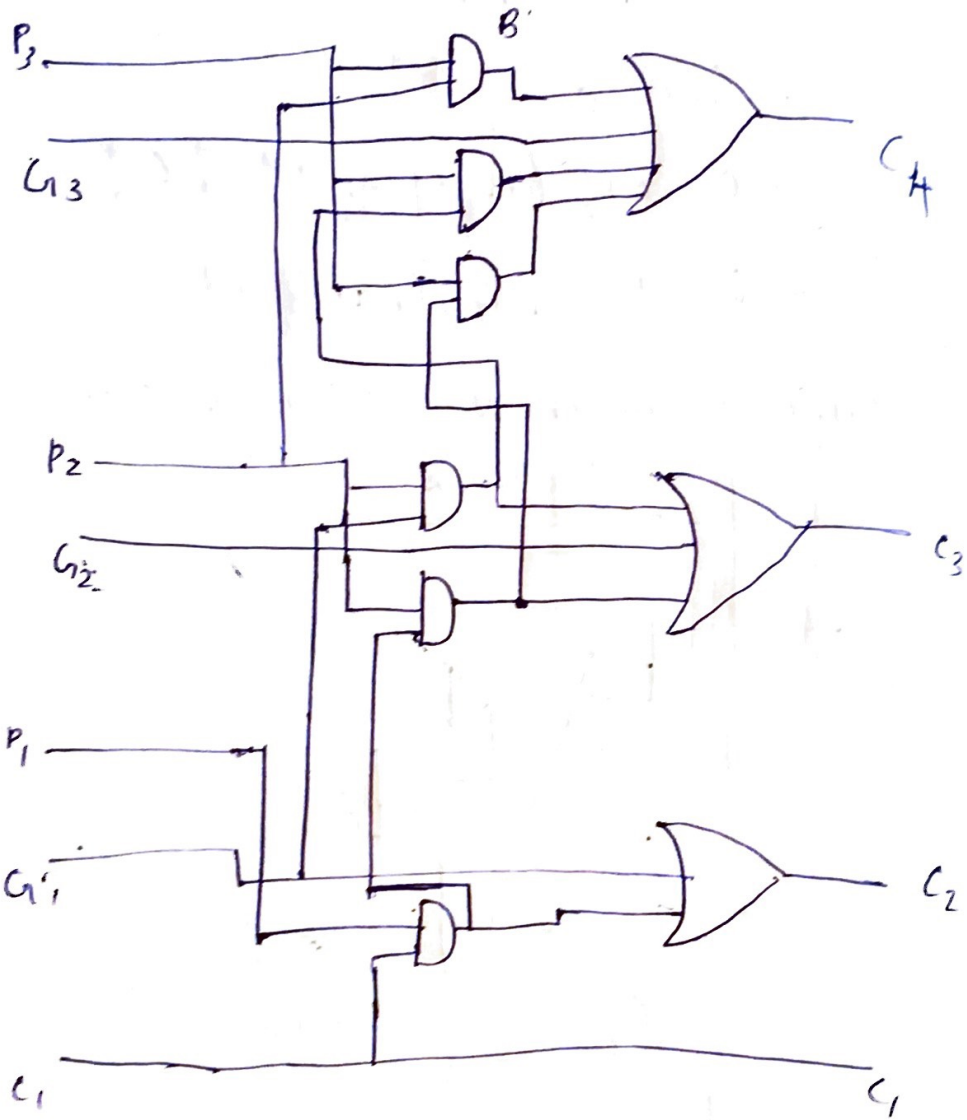
$C_4 = G_3 + P_3 C_3$

$= G_3 + P_3 (G_2 + P_2 G_1 + P_2 P_1 C_1)$

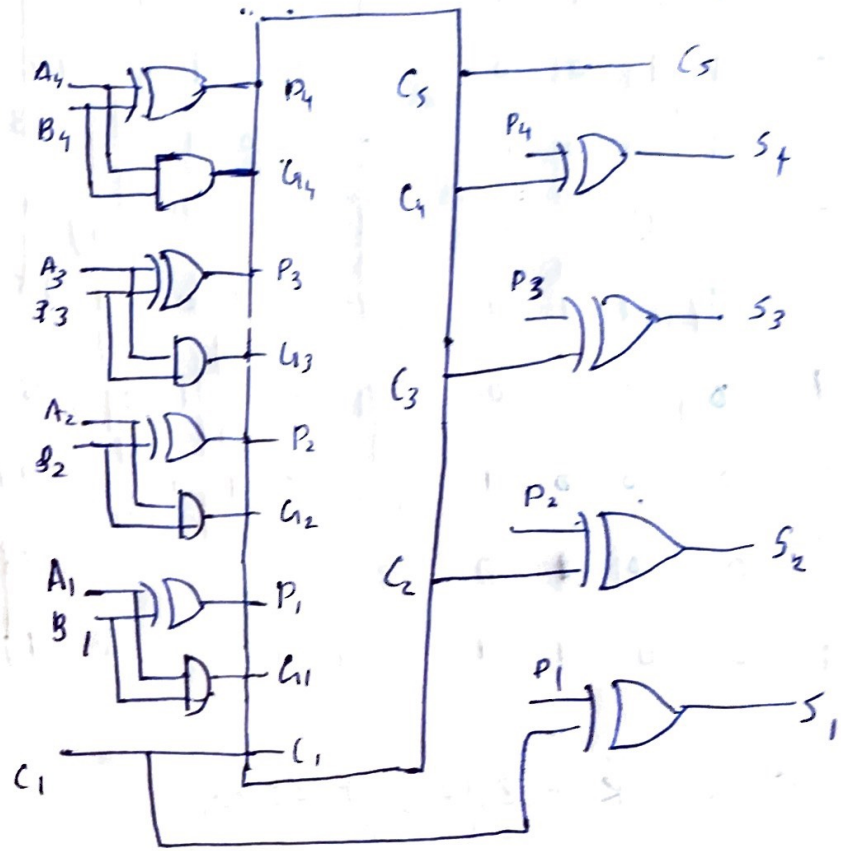
$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1$

LOGIC DIAGRAM OF LOOK AHEAD CARRY GENERATOR





4-bit binary parallel adder with look ahead carry generation

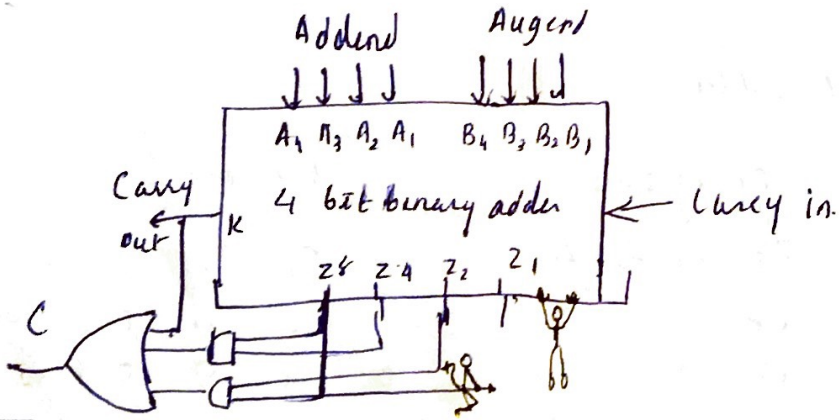


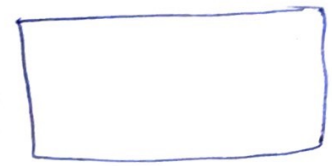
Decimal adder

Binary sum					BCD sum					
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1

0	1	0	0	1		0	1	0	0	1	9	
0	1	0	1	0		1	0	0	0	0	10	
0	1	0	1	1		1	0	0	0	1	11	
0	1	1	0	0		1	0	0	1	0	12	
0	1	1	0	1		1	0	0	1	1	13	
0	1	1	1	0		1	0	1	0	0	14	
0	1	1	1	1		1	0	1	0	1	15	
1	0	0	0	0		1	0	1	1	0	16	
1	0	0	0	1		1	0	1	1	1	17	
1	0	0	1	0		1	1	0	0	0	18	
1	0	0	1	1		1	1	0	0	1	19	

$$C = K + Z_8 Z_2 + Z_8 Z_4$$





MAGNITUDE COMPARATOR

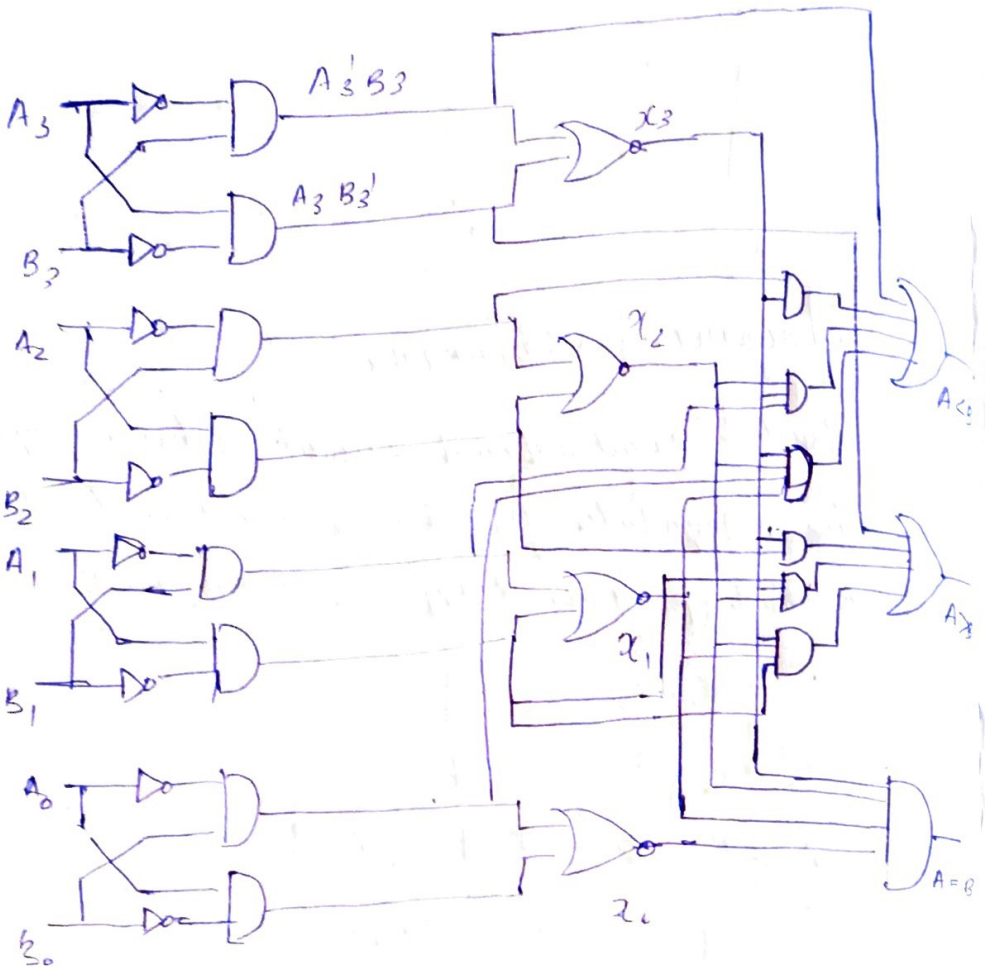
- digital combinational circuit which compare the magnitudes of 2 binary numbers A and B and produces 3 o/p's → A > B, A = B, A < B

$$x_i = A_i B_i + A_i' B_i'$$

$$A = B \Rightarrow x_3\, x_2\, x_1\, x_0$$

$$A > B \Rightarrow A_3 B_3' + x_3 A_2 B_2' + x_3 x_2 A_2 B_2' + x_3 x_2 x_1 A_1 B_1'$$

$$A < B \Rightarrow A_3' B_3 + x_3 A_2' B_2 + x_3 x_2 A_1' B_1 + x_3 x_2 x_1 A_0' B_0$$



In circuit it will be converted first to binary

DECODER

~~Correct~~ 2×4 decoder ⇒ 2 i/p 4 o/p

n i/p $\leq 2^n$ o/p

If decimal decoder used ⇒ 4 i/p only 10 o/p

4 thru

Convert information from end lines to a maximum of 2^n outputs

- A combinational circuit which converts input from n-lines to a maximum possible 2^n output line. If the i/p has unused values then no. of o/p will be $< 2^n$

  n to m line decoder $m \leq 2^n$

eg: 3 to 8 decoder.

Design a BCD to decimal decoder (4 to 10 deco

BCD	o/p
wxyz	$D_0 - D_9$

wx \ yz	$\bar{y}\bar{z}$	$\bar{y}z$	$y z$	$y\bar{z}$
$\bar{w}\bar{x}$	D_0	D_1	D_3	D_2
$\bar{w}x$	D_4	D_5	D_7	D_6
wx	~~D_{12}~~	~~D_{13}~~	~~D_{15}~~	~~D_{14}~~
$w\bar{x}$	D_8	D_9	~~D_{11}~~	~~D_{10}~~

$D_0 = \bar{w}\bar{x}\bar{y}\bar{z}$

$D_1 = \bar{w}\bar{x}\bar{y}z$

$D_2 = \bar{x}\bar{y}z$

$D_3 = \bar{x}yz$

$D_4 = x\bar{y}\bar{z}$

$D_5 = x\bar{y}z$

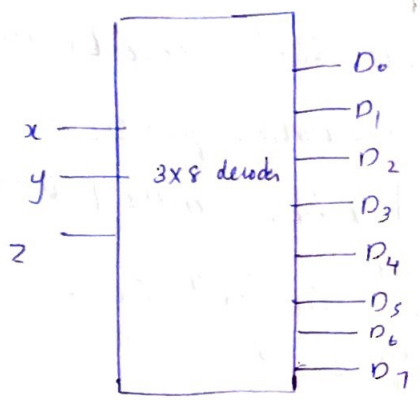
$D_6 = xy\bar{z}$

$D_7 = xyz$

$D_8 = \bar{x}\bar{y}\bar{z} = w\bar{z}$

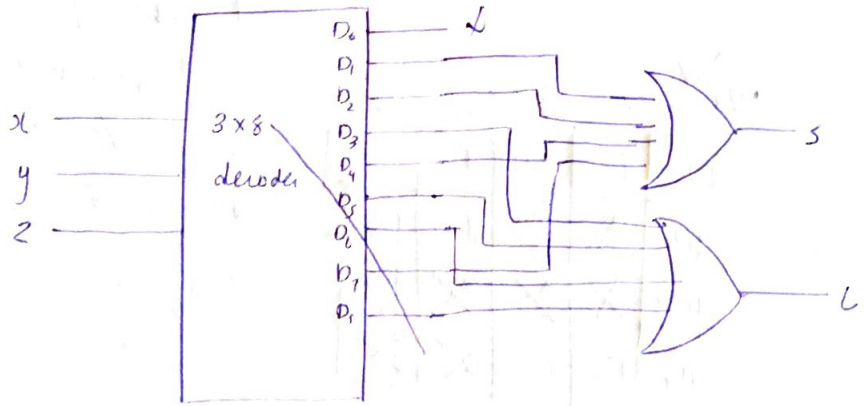
$D_9 = x\bar{y}z = wz$

3 to 8 decoder.



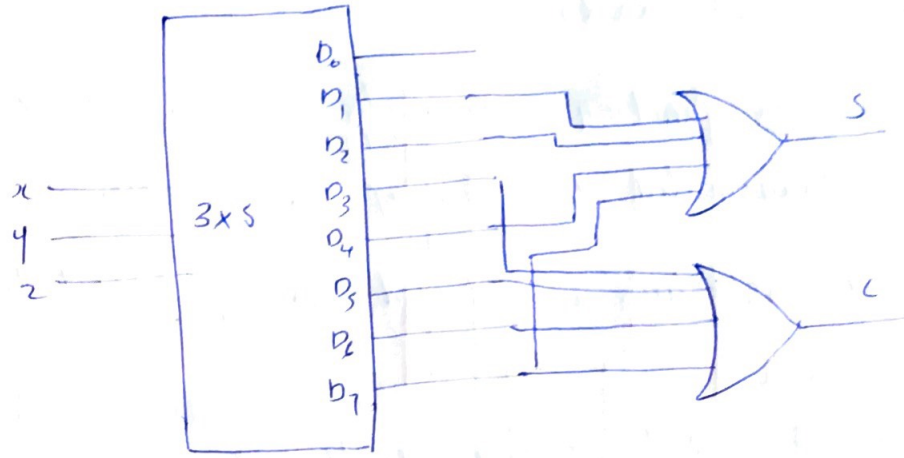
⇒ Design a full adder using a 3x8 decoder and 2 OR gates

COMBINATIONAL LOGIC IMPLEMENTATION



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

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

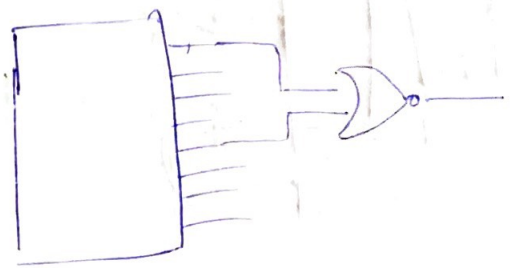


A decoder provides the 2^n minterms of n i/p variable. Since any boolean function can be expressed in sum of minterm canonical form, we can use a decoder to generate the minterms and an external OR gate to form the sum

ie Any combinational circuit with n i/ps and m o/p can be implemented with a n-to-2^n decoder and m OR gates

$S = \Sigma(1,2,3,5,6,7)$

eg:



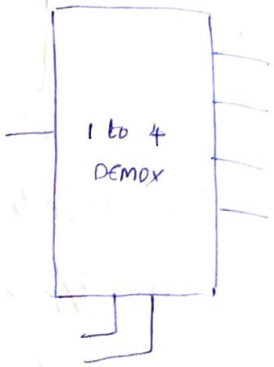
0, 4:

NOR gate

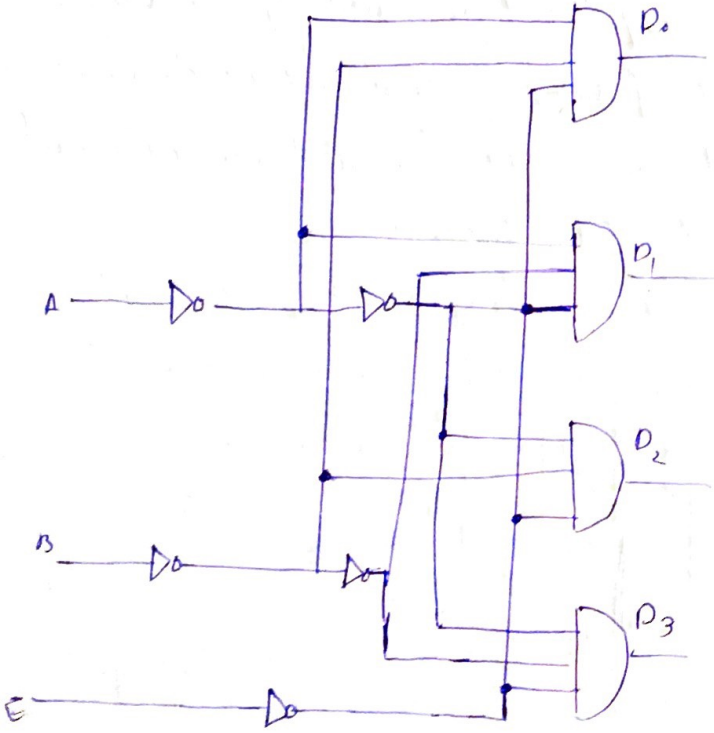
MUX → Many → One
DEMUX ⇒ One → Many

DEMULTIPLEXERS

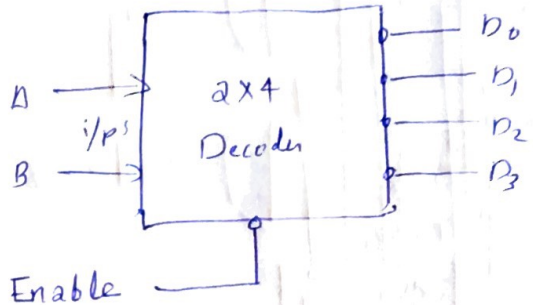
Digital function where i/p from a single line is transmitted to one of the possible 2^n o/p lines. The o/p line is selected by 'n' select lines.



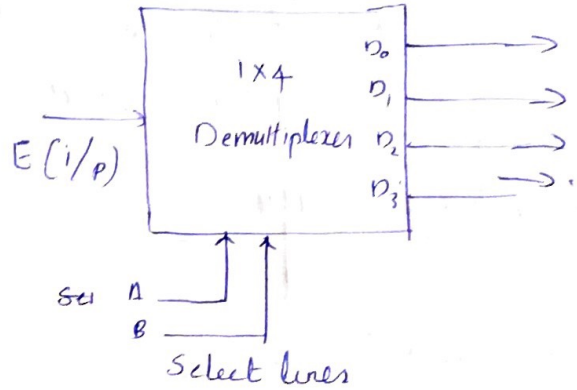
ex. A to 2 to 4 line decode with enable



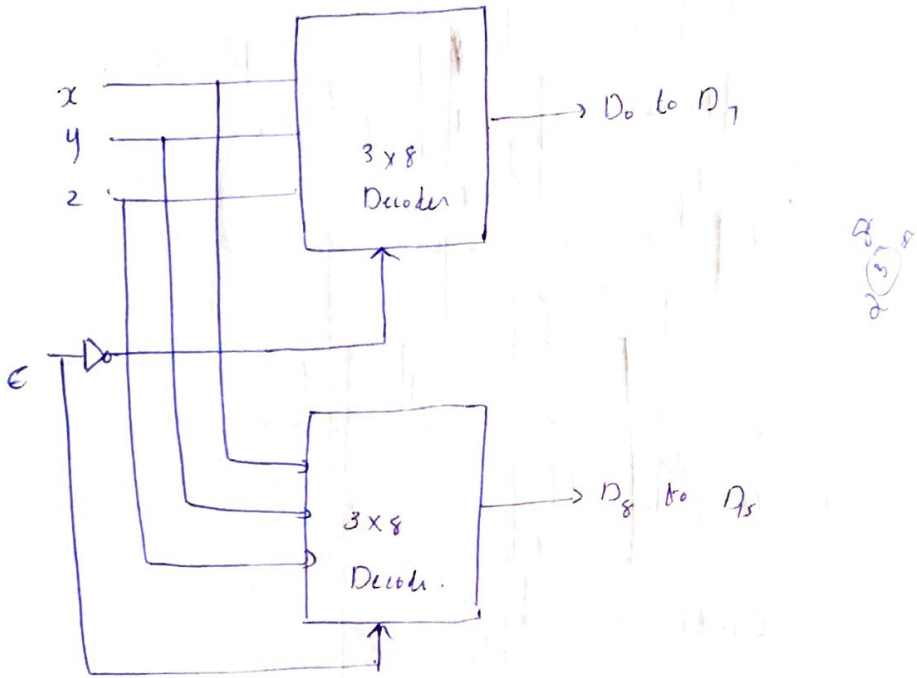
E	A	B	D_0	D_1	D_2	D_3
1	X	Y	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0



Decoder with enable.



Design a 4x16 decoder with two 3x8 decoders



ENCODER - Reverse operation of a decoder

no. of i/ps → $\leq 2^n$

no. of o/ps → n

ex: Octal to binary encoder.

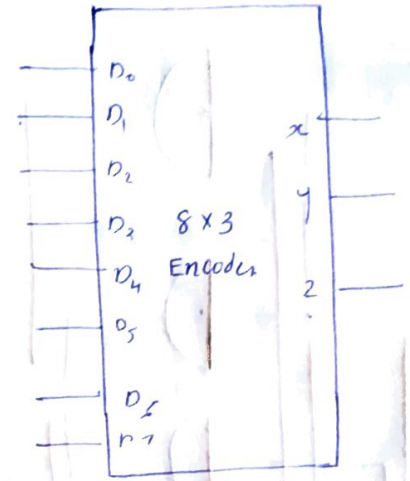
$D_0 - D_7$

$x = D_4 + D_5 + D_6 + D_7$

$y = D_2 + D_3 + D_6 + D_7$

$z = D_1 + D_3 + D_5 + D_7$

000	D_0
001	1
010	2
011	3
100	4
101	5
110	6
111	D_7



Encoder → IC → Priority encoder.

If 2 signals are semultaneously 1, then Larger o/p is selected

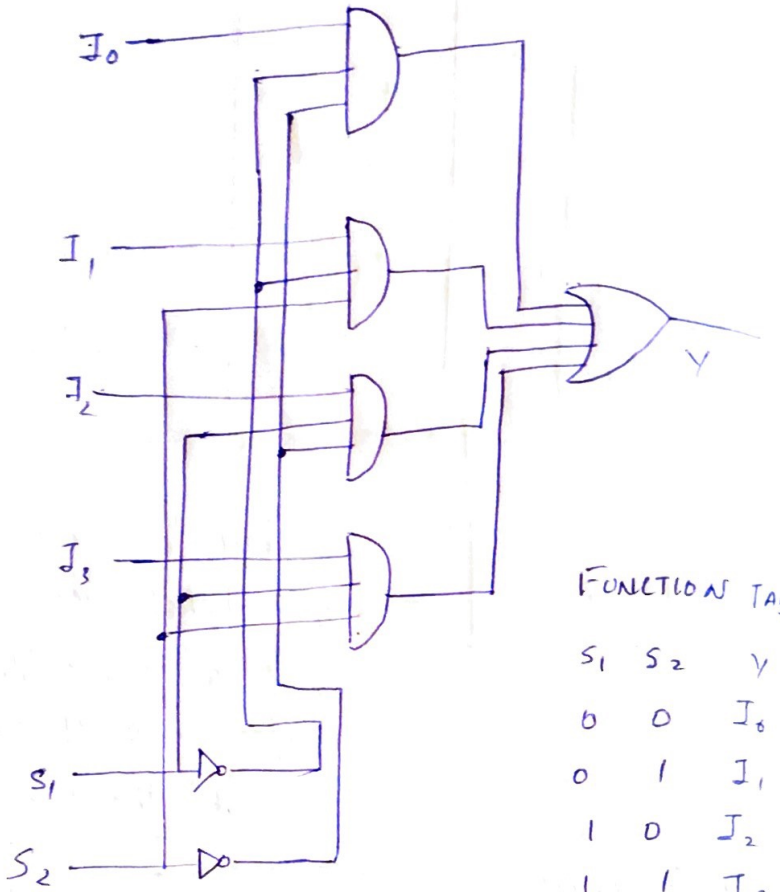
eg: If D_2 and $D_5 = 1$, we select the o/p 101

DEMULTIPLEXER

⇒ Data selection

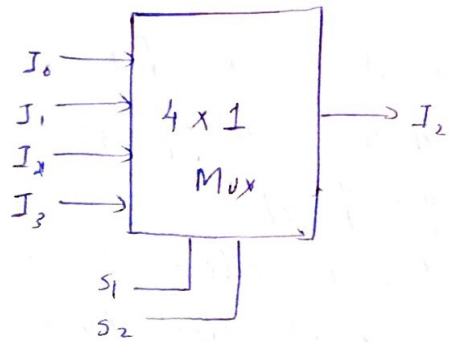
- a combinational circuit that selects binary information from one of many i/p lines & directs it into a single o/p line.

The selection of a particular i/p line is controlled by a set of select lines



FUNCTION TABLE

S_1	S_2	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



MUX → with enable

↓

Strobe (something that activates) multiplexer

BOOLEAN FUNCTION IMPLEMENTATION

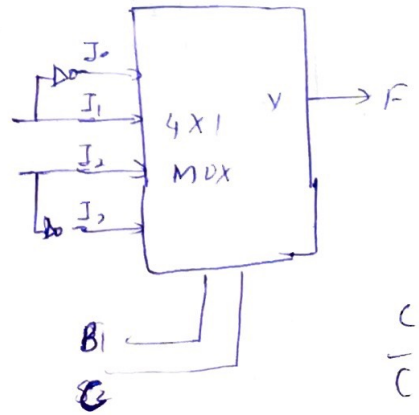
- Any boolean function having n variables can be implemented in a 2^n to 1 MUX

e.g. $F(A,B,C) = \Sigma(1,3,5,6)$

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

(A) IMPLEMENTATION TABLE

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



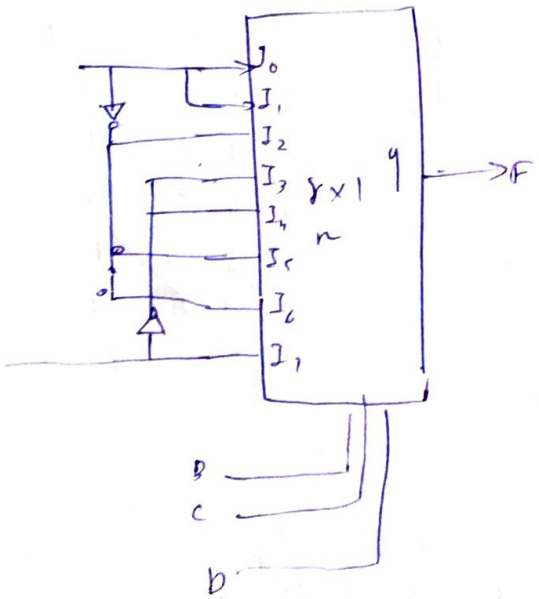
	I_0	I_1	I_2	I_3
	0	2	4	6
C	1	3	5	7
$\bar{C}$				
	C	C	C	$\bar{C}$

- Implement the following function with a MUX

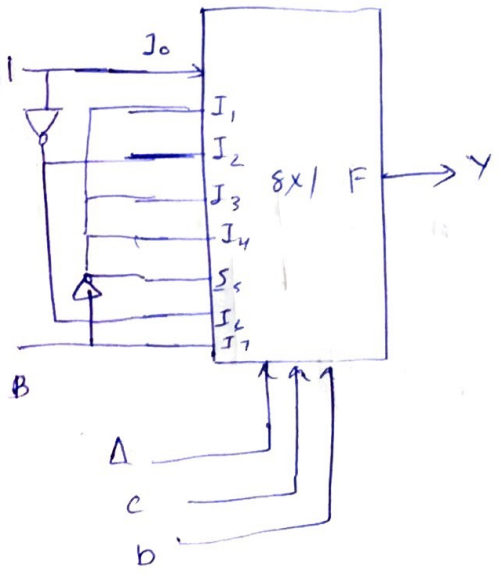
$F(A,B,C,D) = \Sigma(0,1,3,4,8,9,15)$

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{A}$	(0)	(1)	2	(3)	(4)	5	6	7
A	(8)	(9)	10	11	12	13	14	(15)
	1	1	0	$\bar{A}$	$\bar{A}$	0	0	A

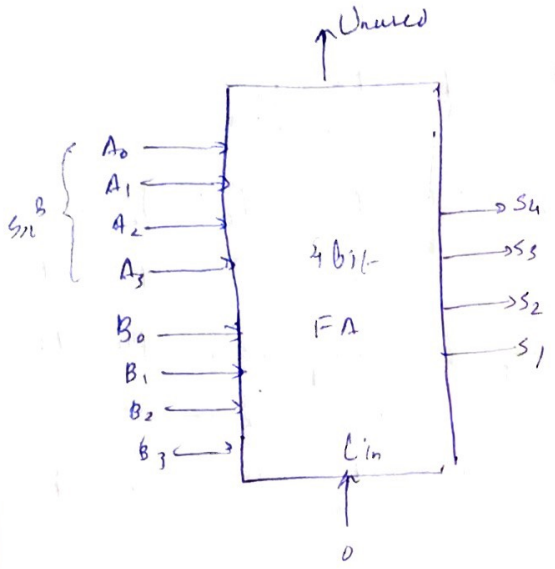
A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



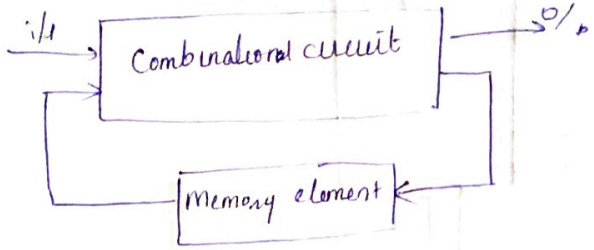
	J_0	J_1	J_2	J_3	J_4	J_5	J_6	J_7
$\bar{B}$	(0)	(1)	2	(3)	(4)	5	6	7
B	8	9	10	11	12	13	(14)	(15)
	1	$\bar{B}$	0	$\bar{B}$	$\bar{B}$	$\bar{B}$	0	B



4) Design an excess 3 to BCD converter using a 4 bit full adder MSI circuit.



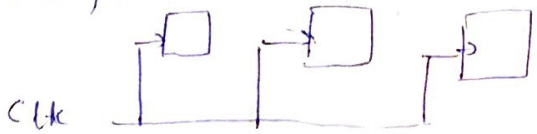
SEQUENTIAL LOGIC



Synchronous :-

→ Clock pulse

At fixed intervals



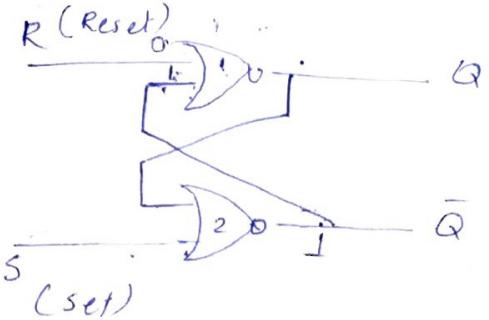
Asynchronous

Each component triggered in a non-uniform manner

Memory element → Flip flop.

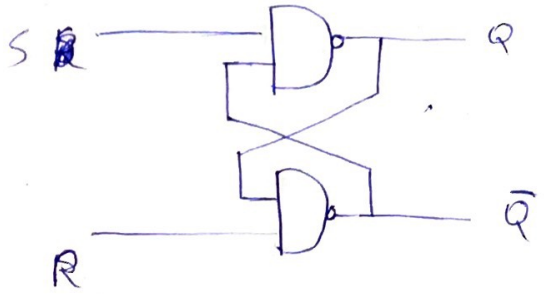
Basic flip flop.

NAND and NOR,



Reset = 0
Set = 1

S	R	Q	$\bar{Q}$
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1
1	1	0	0



S	R	Q	$\bar{Q}$
1	0	0	1
0	1	1	0
0	0	1	1