1. BOOLEAN ALGEBRA AND LOGIC GATES

Number system

1. Binary to Decimal Conversion

$$(1010.011)_2 = 2^3 + 2^1 + 2^{-2} + 2^{-3}$$

$$= (10.375)_{10}$$

2. Binary to octal

$$(101100011010111.111100000110)_2$$

divide into three digits

$$\frac{10}{2}\ \frac{110}{6}\ \frac{001}{1}\ \frac{101}{5}\ \frac{011}{3} \cdot \frac{111}{7}\ \frac{100}{4}\ \frac{000}{0}\ \frac{110}{6}$$

$$= (26153.7406)_8$$

3. Binary to Hexadecimal

divide the binary number into four digits

$$\frac{10}{2}\ \frac{1100}{C}\ \frac{0110}{6}\ \frac{1011}{B} \cdot \frac{1111}{F}\ \frac{0010}{2}$$

$$= (2C6B.F2)_{16}$$

1. Decimal to Binary Conversion

i. $(41)_{10}$

$$
\begin{array}{r|l}
2 & 41 \\
2 & 20 - 1 \\
2 & 10 - 0 \\
2 & 5 - 0 \\
2 & 2 - 1 \\
 & 1 - 0
\end{array}
$$

$(101001)_2$

ii. $(0.6875)_{10}$

$0.6875 \times 2 = (1).3750$

$0.3750 \times 2 = (0).7500$

$0.7500 \times 2 = (1).5000$

$0.500 \times 2 = (1).0000$

$(0.1011)_2$

2. Decimal to octal Conversion

$(153.513)_{10}$

$$
\begin{array}{r|l}
8 & 153 \\
8 & 19 - 1 \\
 & 2 - 3
\end{array}
$$

$(231)_8$

$0.513 \times 8 = (4).104$

$0.104 \times 8 = (0).832$

$0.832 \times 8 = (6).656$

$0.656 \times 8 = (5).248$

$0.248 \times 8 = (1).984$

$(0.40651)_8$

ANS $(231.40651)_8$

3. Decimal to Hexadecimal Conversion

$(2482)_{10}$

16 | 2482
16 | 155 – 2
9 – 11

$(9\,11\,2)_{16}$

ANS = $(9C2)_{16}$

1. octal to binary Conversion

$(673.124)_8$

$= \underset{6}{110}\ \underset{7}{111}\ \underset{3}{011}\ .\ \underset{1}{001}\ \underset{2}{010}\ \underset{4}{100}$

$(110111011.001010100)_2$

2. octal to Decimal Conversion

$(231.406)_8$

$2\times8^2 + 3\times8^1 + 1\times8^0 . 4\times8^{-1} + 0\times8^{-2} + 6\times8^{-3}$

$128 + 24 + 1 . 0.5 + 0 + 0.01171$

$(153.51171)_{10}$

3. octal to Hexadecimal Conversion

$(673.124)_8 = (\quad)_{16}$

$\underset{6}{110}\ \underset{7}{111}\ \underset{3}{011} \cdot \underset{1}{001}\ \underset{2}{010}\ \underset{4}{100}$

110111011.001010100

$\underset{1}{0001}\ \underset{11}{1011}\ \underset{11}{1011} \cdot \underset{2}{0010}\ \underset{10}{1010}$

$(1BB.2A)_{16}$

1. Hexadecimal to Binary Conversion

$(306.D)_{16} = (\quad)_2$

$\underset{0011}{3}\ \underset{0000}{0}\ \underset{0110}{6} \cdot \underset{1101}{D}$

$(001100000110.1101)_2$

2. Hexadecimal to Decimal Conversion

$(306.D)_{16} = (\quad)_{10}$

$3\times16^{+2} + 0\times16^{+1} + 6\times16^{0} \cdot 13\times16^{-1}$

$768 + 0 + 6 \cdot 0.8125$

$(774.8125)_{10}$

3. Hexadecimal to octal Conversion

$(306.D)_{16} = (\quad)_8$

3	0	6	.	D
0011	000	0110		1101

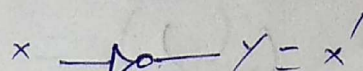
110	000	110	.	110	100
6	0	6		6	4

$(606.64)_8$

Decimal (base 10)	Binary (base 2)	octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

LOGIC GATES

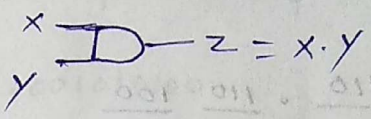
1. NOT Gate



TRUTH TABLE

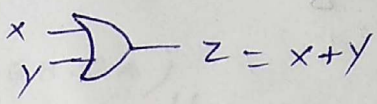
x	y
0	1
1	0

2. AND Gate



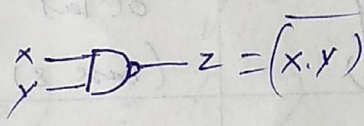
x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

3. OR gate



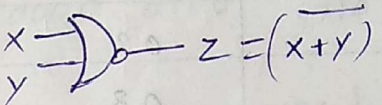
x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

4. NAND Gate



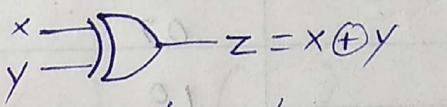
x	y	z
0	0	1
0	1	1
1	0	1
1	1	0

5. NOR gate



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

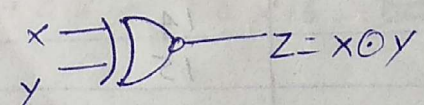
6. XOR gate



$z = x'y + yx'$

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

7. XNOR gate



$z = x'y' + xy$

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

Complements :-

Complements are used in digital Computers for simplifying the subtraction operations and for logical manipulations.

Two types

1) r's Complement

2) (r-1)'s Complement

10's and 9's Complement for decimal numbers

2's and 1's Complement for binary numbers.

r's Complement of a numbers N is :

$r^n - N$ n - no of digits

10's Complement of $(52520)_{10}$

$10^5 - 52520 = 47480$

10's Complement of $(0.3267)_{10}$

$10^0 - 0.3267 = 0.6733$

2's Complement of $(101100)_2$ is

$(2^6)_{10} - (101100)_2 = (100000 - 101100)_2$

$= 010100$

2's Complement of $(0.0110)_2$ is $(1 - 0.0110)_2$

$= 0.1010.$

(r-1)'s Complement of a Given positive number N is

$$r^n - r^{-m} - N$$

r - radix n - no of digits in integer part

m - no of digits in fractional part

Example

The 9's Complement of $(25.639)_{10}$

$$10^2 - 10^{-3} - 25.639$$

$$= 99.999 - 25.639$$

$$= 74.360$$

The 1's Complement of $(101100)_2$ is

$$(2^6 - 1)_{10} - (101100) = (111111 - 101100)_2$$

$$= 010011$$

The 1's Complement of $(0.0110)_2$ is

$$(1 - 2^{-4})_{10} - (0.0110)_2$$

$$= (0.1111 - 0.0110)_2$$

$$= 0.1001$$

Note:

9's Complement Can be obtained by subtraction of 9 to every digit

1's Complement Can be obtained by Changing 0's to 1's and 1's to 0's.

Subtraction with r's Complements

The subtraction of two positive numbers (M − N) both of base r, may be done as follows.

1. Add the minuend M to the r's complement of the subtrahend N.
2. Inspect the result obtained in step 1.
   a) If an end carry occurs, discard it
   b) If an end carry does not occur, take the r's complement of the result and place a negative sign

Example:

1. Subtract (1010100 − 1000100)

```
  1010100
+ 0111100   <- 2's complement
-------
(1)0010000
```

end carry discard

Ans: 0010000

2. Subtract (1000100 − 1010100)

```
  1000100
+ 0101100   <- 2's complement
-------
  1110000
```

Here no carry so takes 2's complement of the result and place negative sign in front.

− (2's complement of 1110000) = Ans − 10000

Subtraction With (r-1)'s Complement.

The subtraction of (M - N) With base r is

1. Add the minuend M to the (r-1)'s Complement of subtrahend N.

2. Inspect the result obtained in step 1
   - a) If an end Carry occurs add 1 to the Least significant digit
   - b) If an end Carry does not occur take the (r-1)'s Complement of the resut and place a negative sign in front.

Example:

1. Subtract (1010100 - 1000100) using (r-1)'s Complement

```
  + 1010100
    0111011   <- 1's Complement
   ---------
end Carry  (1)0001111
             \---->  1 +
   ---------
    0010000
   ---------
```

2. Subtract (1000100 - 1010100)

```
  + 1000100
    0101011   <- 1's Complement
   ---------
    1101111
   ---------
```

no end Carry so Take 1's Complement of The result and place negative sign.

ANS = - (0010000)

Comparison between 1's and 2's Complement

1) While taking Complement 1's Complement is easier Compared to 2's Complement because in 1's complement 1's are Converted to 0's and 0's to 1's.

2) During subtraction 2's Complement is advantageous it requires only one arithmetic addition whereas subtraction using 1's Complement requires two arithmetic addition if end Carry occurs.

BINARY CODES:

Digital Data is represented, stored and transmitted as group of binary bits are Called as binary Code.

Types

1. Weighted Code:
   e.g 8421, BCD, 2421
2. Non weighted Codes:
   e.g Excess-3 Code, Gray Code
3. Alphanumeric Codes
   e.g ASCII, EBCDIC
4. Error Detection Codes
   e.g parity, Hamming Codes.

Binary Coded Decimal (BCD) Code

In this Code each decimal digit is represented by a 4-bit binary number.

Decimal	(BCD) 8421	Excess-3	2421
0 —	0000	0011	0000
1 —	0001	0100	0001
2 —	0010	0101	0010
3 —	0011	0110	0011
4 —	0100	0111	0100
5 —	0101	1000	1011
6 —	0110	1001	1100
7 —	0111	1010	1101
8 —	1000	1011	1110
9 —	1001	1100	1111

e.g BCD Code for 15 is 0001 0101.

BCD additions

```
  4 - 0100
+ 5 - 0101
------------
Ans 9 - 1001
```

```
  4   0100
+ 8   1000
------------
 12   1100 - binary
    + 0110
------------
12 - 00010010 - BCD
```

If the result is more than 9 then add 6 to the result to get BCD value.

Gray Code :-

It is also a Reflected Code as only one bit changes from one number to another.

		Gray Code
0000	0 – 0000	– 0000
0001	1 – 0001	– 0001
0010	2 – 0010	– 0011
0011	3 – 0011	– 0010
0100	4 – 0100	– 0110
0101	5 – 0101	– 0111
0110	6 – 0110	– 0101
0111	7 – 0111	– 0100
1000	8 – 1000	– 1100
1001	9 – 1001	– 1101
1010	10 – 1010	– 1111
1011	11 – 1011	– 1110
1100	12 – 1100	– 1010
1101	13 – 1101	– 1011
1110	14 – 1110	– 1001
1111	15 – 1111	– 1000

Binary Logic :

Binary Logic deals with variables that take on two discrete values and with operations that assume logical meaning. The value can be (true or false, yes or no, 1 or 0)

Boolean Algebra :-

Duality principle :- It states that every algebraic expression deducible from postulates of Boolean algebra remain valid if the operators and identity elements are interchanged.

In simple interchange OR and AND operators and replace 1's by 0's and 0's by 1's.

Example

$x'yz' + x'y'z \xleftrightarrow{\text{Duality}} (x'+y+z')(x'+y'+z)$

Postulates and Theorem of Boolean algebra

Postulate 2	a) $x+0=x$	b) $x.1=x$
Postulate 5	a) $x+x'=1$	b) $x.x'=0$
Theorem 1	a) $x+x=x$	b) $x.x=x$
Theorem 2	a) $x+1=1$	b) $x.0=0$
Theorem 3 (Involation)	$(x')'=x$	
Postulate 3 (Commutative)	a) $x+y=y+x$	b) $xy=yx$
Theorem 4 Associative	a) $x+(y+z)=(x+y)+z$	b) $x(yz)=(xy)z$
Postulate 4 (Distributive)	a) $x(y+z)=xy+xz$	b) $x+yz=(x+y)(x+z)$
Theorem 5 Demorgan	a) $(x+y)'=x'y'$	b) $(xy)'=x'+y'$
Theorem 6 Absorption	a) $x+xy=x$	b) $x(x+y)=x$

operator precedence:

1) parantheses 2) NOT 3) AND 4) OR

Simplify Boolean function to minimum number of literals

1. $x + x'y$

By using Distributive

$= (x + x')(x + y)$ $\boxed{x + x' = 1}$

$= x + y$

2. $xy + x'z + yz$

$= xy + x'z + yz(x + x')$ $\boxed{x + x' = 1}$

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

$= xy(1 + z) + x'z(1 + y)$ $\boxed{1 + z = 1,\ 1 + y = 1}$

$= xy + x'z$

3) $x(x' + y)$

$= xx' + xy$ $\boxed{xx' = 0}$

$= xy$

Complement of a Function

Example:

Find the Complement of the function

1) $F_1 = x'yz' + x'y'z$

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

$F_1' = (x+y'+z)(x+y+z')$

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

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

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

~~Canonical~~

Canonical and standard form:

A binary Variable may appear either in its normal form (x) or in its Complement form x'.

A literal is a primed or unprimed Variable.

e.g $F = x'yz + xy$

Here x', y, z, x, y each one is a literal

Minterms and Maxterms :-

For Three binary Variables

			Minterms		Maxterms	
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

A minterm is a product (AND) of all Variables in the function, in direct or Complemented form.

A manterm is a sum (OR) of all the Variables in the function, in primed or unprimed form.

The Variable is primed if the bit is 1(x) and unprimed if the Corresponding bit is 0.(x')

Example:

x	y	z		Function f_1
0	0	0	—	0
0	0	1	—	1
0	1	0	—	0
0	1	1	—	0
1	0	0	—	1
1	0	1	—	0
1	1	0	—	0
1	1	1	—	1

From the truth table the Boolean function can be expressed as.

$$f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$$

$$f_1 = \sum(1,4,7)$$

Above Boolean function is expressed in sum of minterms.

To Express the above Boolean expression in product of Maxterms

$$f_1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$$

$$f_1 = (x'y'z' + x'yz' + x'yz + xy'z + xyz')'$$

$$f_1 = (x+y+z)(x+y'+z)(x+y'+z')(x'+y+z')(x'+y'+z)$$

$$f_1 = M_0 \cdot M_2 \cdot M_3 M_5 M_6$$

$$f_1 = \pi(0,2,3,5,6)$$

Boolean function expressed as a sum of minterms or product of manterms are said to be in canonical form.

Example:

1. Express the Boolean function $F = A + B'C$ in a sum of minterms.

$$F = A + B'C$$

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

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

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

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

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

$$F = m_1 + m_4 + m_5 + m_6 + m_7$$

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

2. Express the Boolean function $F = xy + x'z$ in a product of manterm form.

$$F = xy + x'z$$

using distributive property

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

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

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

$$(x+z)(y+z)$$

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

Each OR term missing one variable.

$F = (x'+y+zz')(x+z+yy')(y+z+xx')$

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

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

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

$F = M_0 M_2 M_4 M_5$

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

In Canonical form Each minterm or maxterm must Contain all the variables either in Complemented or uncomplemented form. (or) All the term have same no of literals

e.g $F(A,B,C) = ABC + ABC' + AB'C + AB'C' + AB'C$.

In standard form each term in the function may Contain one, two or any number of literals

Two types 1. sum of products
2. product of sums.

e.g $F(x,y,z) = y' + xy + x'y'z'$ — SOP

~~$F(A,B,C,D) = A'B'C'D$~~.

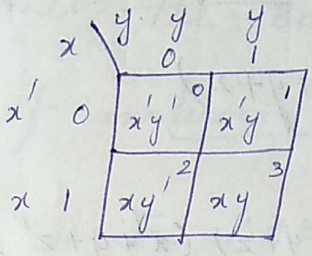
$F(x,y,z,w) = x(y'+z)(x'+y+z'+w)$ — POS

Simplification of Boolean Functions

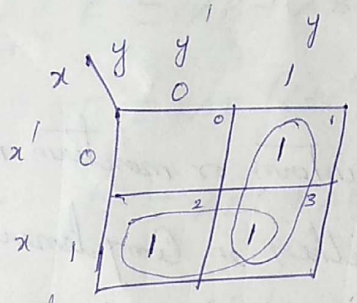
The MAP method

Also called "Karnaugh map"

Two Variable map

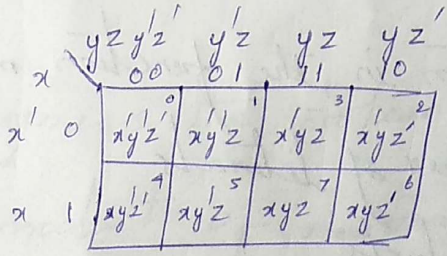


e.g Simplify the Boolean function $F = \Sigma(1,2,3)$



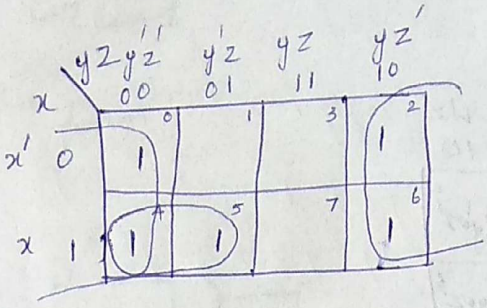
simplified function $F = x + y$

Three Variable Map



e.g Simplify $F(x,y,z) = \Sigma(0,2,4,5,6)$

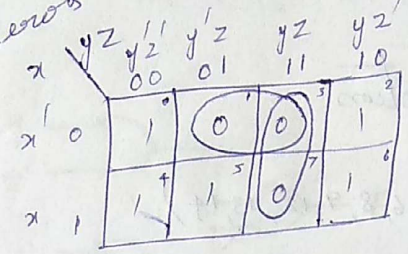
It is three variable Boolean function and minterms are given.



$$F = z' + xy'$$

It is in the form of SOP (sum of products)

To get POS form Combine zeros



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

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

$$F = (x + z')(y' + z') \quad \text{POS form}$$

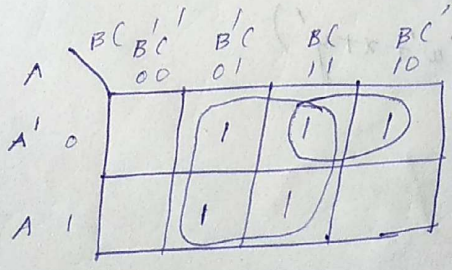
Note:

1) To get SOP form Combine 1's in K-map

2) To get POS form Combine 0's in the K-map and then take Complement

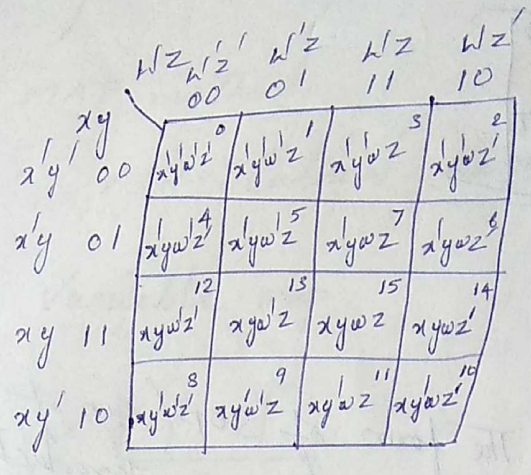
2. Simplify the Boolean function

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



$$F = C + A'B$$

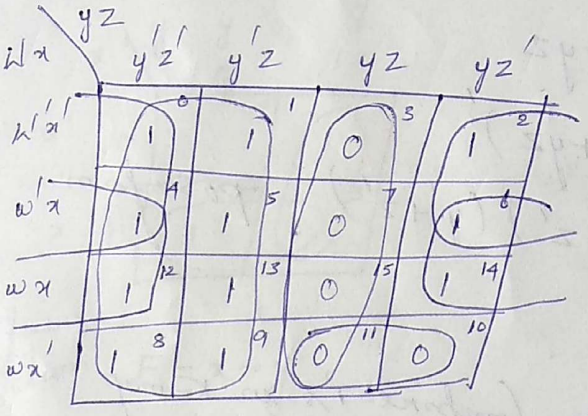
Four Variable Map



Example

Simplify the Boolean function

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



$$F = y' + w'z' + xz'$$

To get pos form

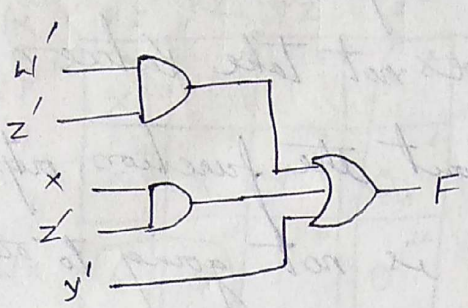
$$F' = yz + wx'y$$

$$F = (yz + wx'y)'$$

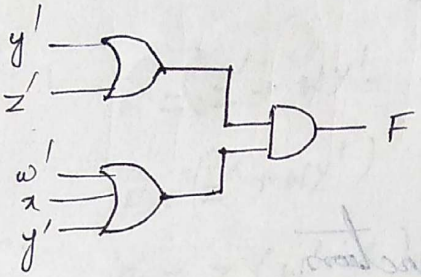
$$F = (y'+z')(w'+x+y')$$

Logic Circuit

SOP form:



POS form:



Five Variable map

AB \ CDE	000	001	011	010	110	111	101	100
00	0	1	3	2	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

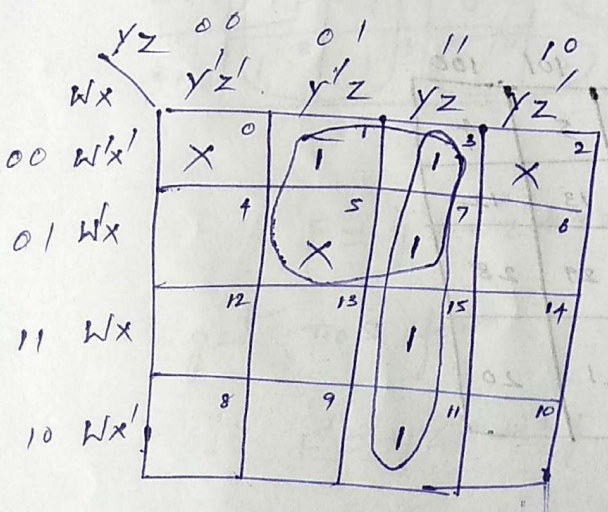
Don't-Care Conditions:

In some applications in which certain combination of input variables never occur. therefore it does not take value 0 or 1. so we don't care about the function output for the variable which is not going to occur. so it is marked with 'x' and it can be used for minimization.

Example:-

Simplify the Boolean function.

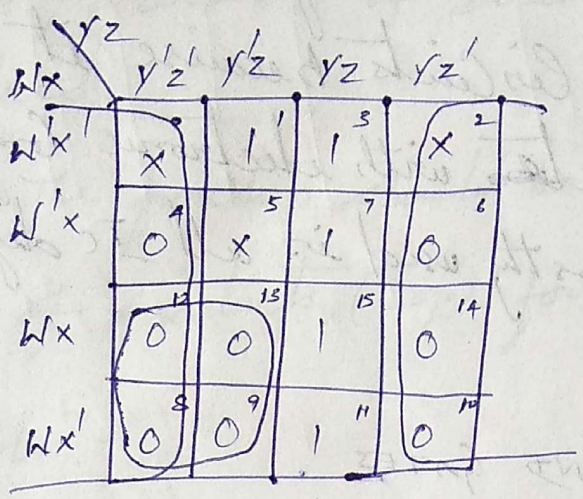
$$F(w,x,y,z) = m\Sigma(1,3,7,11,15) + d\Sigma(0,2,5)$$



$$F = yz + w'z$$

It is SOP form.

pos form :-



$$F' = z' + wy'$$

$$F = (z' + wy')'$$

pos form. $$F = z(w' + y)$$

Disadvantages of K-map :-

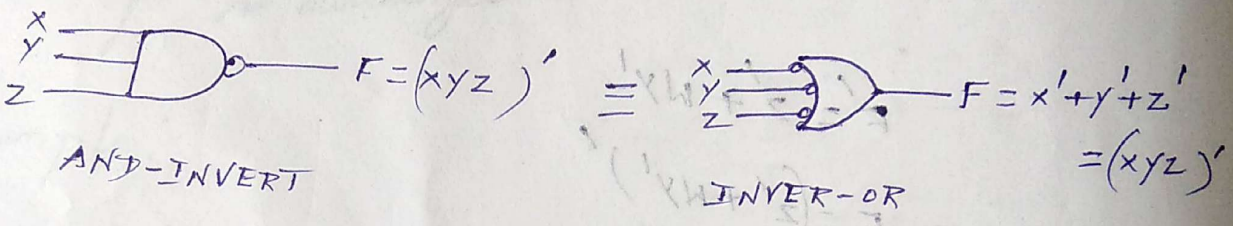
1) Complexity of K-map simplification process increases with increase in the number of variables.

2) The minimum expression obtained might not be unique.

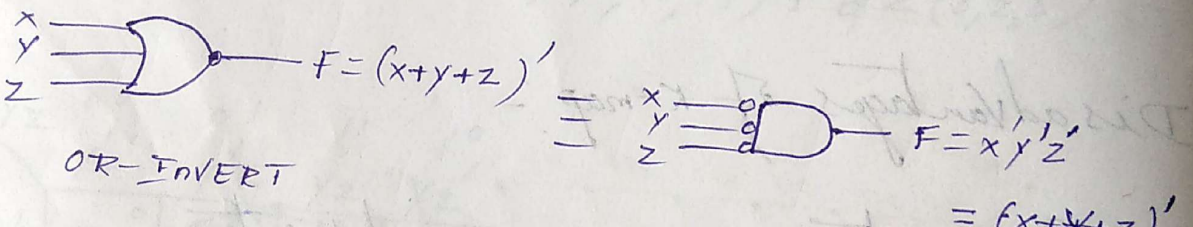
NAND and NOR Implementation :-

NAND and NOR gates are mostly used in digital circuits because it is easier to fabricate with electronic components and they were mostly used in all IC digital logic families

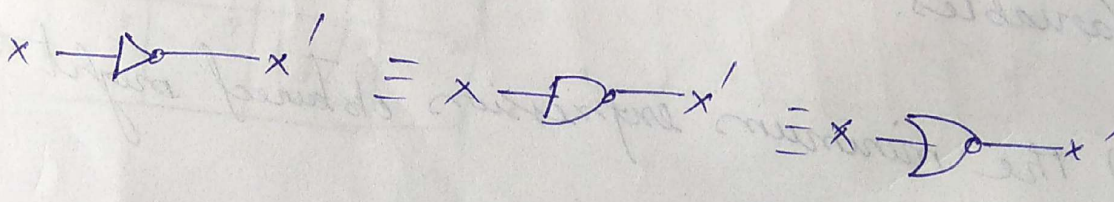
NAND GATES



NOR Gates



INVERTER Gates is NAND and NOR



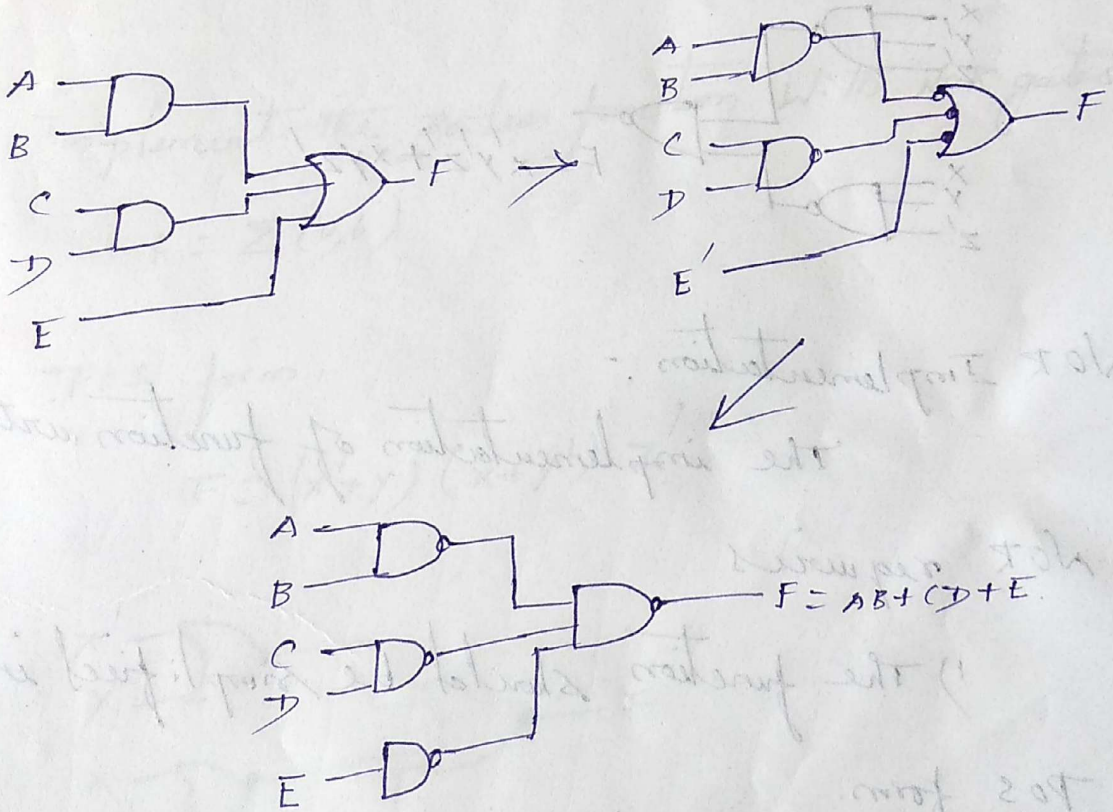
NAND Implementation:

The implementation of Boolean function with NAND requries.

1) The function should be simplified in the sum of products (SOP) form.

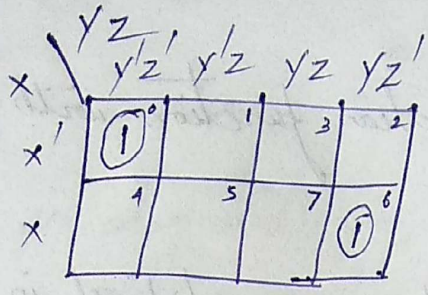
example:-

1. $F = AB + CD + E$



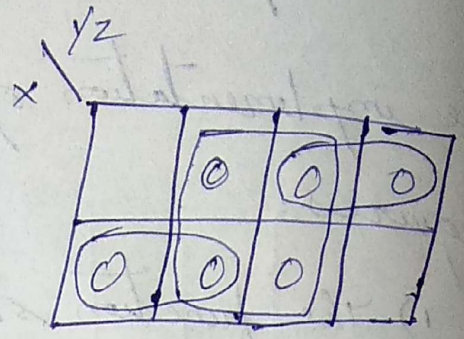
2. Implement the following function with NAND gates.

$F(x, y, z) = \Sigma(0, 6)$



Sop form

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



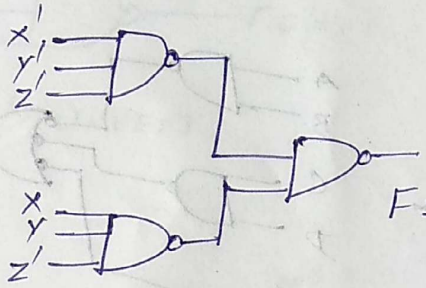
pos form.

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

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

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

NAND Implementation



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

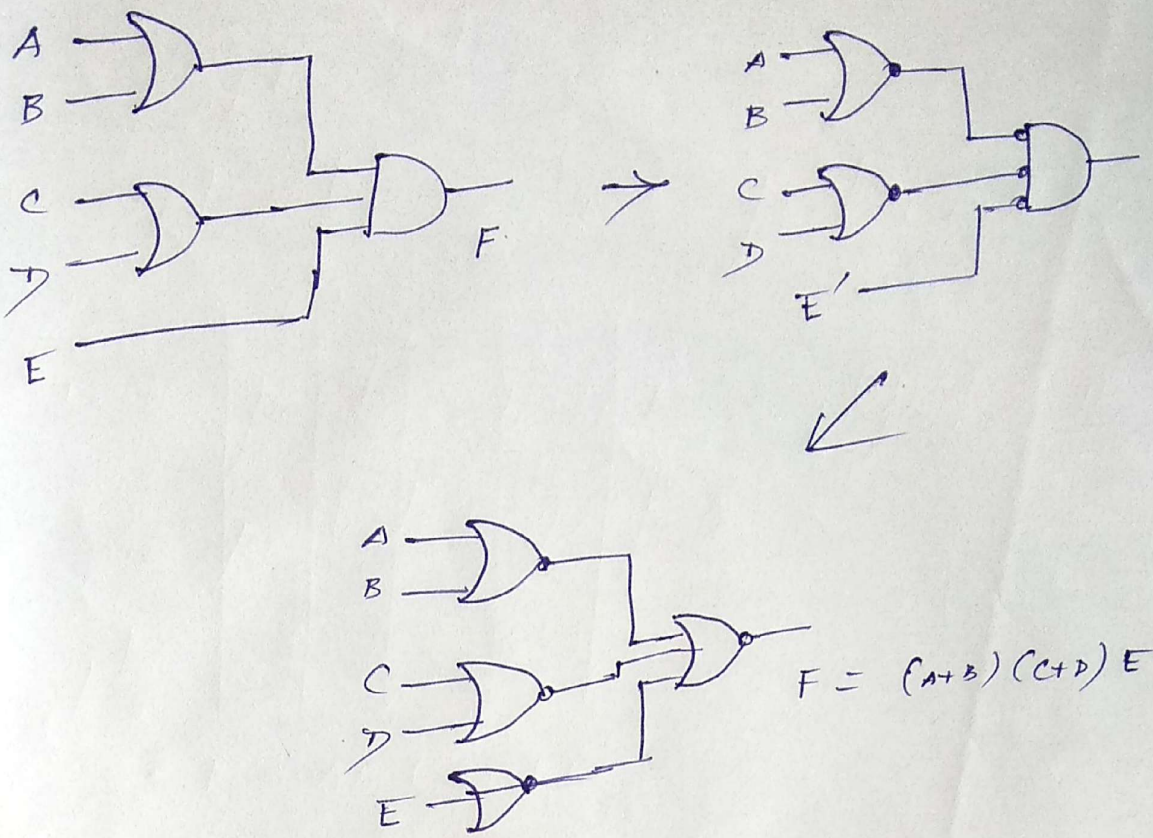
NOR Implementation :-

The implementation of function with NOR requires

1) The function should be simplified in POS form.

example

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



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

2. Implement the Boolean function with NOR gates.

$F = \Sigma(0,6)$

POS form.

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