UNIT - III

Number systems: It represents the magnitude (or) quantity of a number.

- Decimal number system contains 10 unique symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. As counting in decimal involves 10 symbols, its base or radix is 10.
- In this system, any number can be represented by the use of these 10 symbols only. Each symbol is called a digit.
- The left most digit in any number which has greatest positional weight out of all digits present is called MSB (Most significant Bit) and the right most digit with least positional weight is called LSB (Least significant Bit).
- The first digit to the left of the decimal point has a weight of unity or 10^0, second digit has a weight of 10^1 or 10 so on.
- The first digit to the right of decimal point has a weight of 10^{-1}, second has a weight of 10^{-2} and so on.
- It is a positional number system (Positional weighted {... thousand Hund Ten Unit}.
- In this system any number of any magnitude can be represented by the use of these 10 symbols.
- Group 4 bits is called as Nibble.
- Group of 8 bits is called as Bytes.
- Group of 16 bits is called as word
- Group of 32 bits is called as Double word.
- In general the value of any decimal number system can be represented as $d_n d_{n-1} d_{n-2} \cdots d_2 d_1 d_0$ $d_{-1} d_{-2} \cdots, d_{-n}$
- Representation: $d_n \times 10^n + d_{n-1} \times 10^{n-1} + \cdots d_1 \times 10^1 + d_0 \times 10^0 .$
$d_{-1} \times 10^{-1} + d_{-2} \times 10^{-2} + \cdots d_{-n} \times 10^{-n}$

Example: i) Represent 4367 number in decimal

$(4367) = 4\times10^3 + 3\times10^2 + 6\times10^1 + 7\times10^0$

$= 4000 + 300 + 60 + 7 = (4367)_{10}$

ii) Represent (6379.581) number in decimal:

$(6379.581) = 6\times10^3 + 3\times10^2 + 7\times10^1 + 9\times10^0 . 5\times10^{-1} + 8\times10^{-2} + 1\times10^{-3}$

$= (6379.581)_{10}$

Binary Number system: the base or radix of this system is 2. The symbols used are 0 and 1. A binary digit is called a bit. A binary number consist of sequence of bits, each of which either 0 or 1.

- Decimal equivalent of $d_n d_{n-1} d_{n-2} \cdots d_1 d_0 . d_{-1} d_{-2} \cdots d_{-n}$ is $d_n\times2^n + d_{n-1}2^{n-1} + \cdots d_0\times2^0 + d_{-1}2^{-1} + \cdots d_{-n}\times2^{-n}$.

- This can be used in digital computers because switching circuits used in the computer uses two state devices such as (on & off) diodes, transistors etc.

Example: Decimal equivalent of $(10101)_2$ is

$1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 = 16+0+4+0+1 = 21$

$\therefore (10101)_2 = (21)_{10}$

→ Convert $(101.100)_2$ into decimal

$1\times2^2 + 0\times2^1 + 1\times2^0 . 1\times2^{-1} + 0\times2^{-2} + 0\times2^{-3}$

$4+0+1 . 1/2 + 0 + 0$

5.5

$\therefore (101.100)_2 = (5.5)_{10}$

→ Convert $(43)_{10}$ to binary

2 | 43
2 | 21 - 1
2 | 10 - 1
2 | 5 - 0
2 | 2 - 1
2 | 1 - 0
0 - 1

$\therefore (43)_{10} = (101011)_2$

Convert $(13.52)_{10}$ into binary

$$
\begin{array}{r|l}
2 & 13 \\ \hline
2 & 6 - 1 \\ \hline
2 & 3 - 0 \\ \hline
2 & 1 - 1 \\ \hline
 & 0 - 1
\end{array}
\uparrow
$$

Integer part

$0.52 \times 2 = 1.04 - 1$
$0.04 \times 2 = 0.08 - 0$
$0.08 \times 2 = 0.16 - 0$
$0.16 \times 2 = 0.32 - 0$
$0.32 \times 2 = 0.64 - 0$
$0.64 \times 2 = 1.28 - 1$ $\downarrow$

Fractional Part

$\therefore (13.52)_{10} = (1101.100001)_2$

Octal Number System:

- Base or radix $\longrightarrow$ $r = 8$
- It is also a positional weighted number system
- Symbols — 0, 1, 2, 3, 4, 5, 6, 7
- Representation — $(\text{number})_8$

Example: convert $(43)_{10}$ into octal

$$
\begin{array}{r|l}
8 & 43 \\ \hline
8 & 5 - 3 \\ \hline
 & 0 - 5
\end{array}
\uparrow
$$

$(43)_{10} = (53)_8$

- Convert $(13.52)_{10}$ into octal

Integer Part

$$
\begin{array}{r|l}
8 & 13 \\ \hline
8 & 1 - 5 \\ \hline
 & 0 - 1
\end{array}
\uparrow
$$

Fractional Part

$0.52 \times 8 = 4.16 - 4$
$0.16 \times 8 = 1.28 - 1$
$0.28 \times 8 = 2.24 - 2$
$0.24 \times 8 = 1.92 - 1$
$0.92 \times 8 = 7.36 - 7$ $\downarrow$

$\therefore (13.52)_{10} = (15.41217)_8$

→ Convert $(356)_8$ into Decimal

$$3\times 8^2+5\times 8^1+6\times 8^0$$
$$=3\times 64+5\times 8+6\times 1$$
$$=192+40+6$$
$$=238$$
$$\therefore (356)_8=(238)_{10}$$

→ Convert $(\overset{8^2}{2}\,\overset{8^1}{4}\,\overset{8^0}{1}.\overset{8^{-1}}{1}\,\overset{8^{-2}}{2})_8$ into Decimal

$$2\times 8^2+4\times 8^1+1\times 8^0\,.\,1\times 8^{-1}+2\times 8^{-2}$$
$$2\times 64+4\times 8+1\times 1\,.\,1/8+2/64$$
$$128+32+1\,.\,0.125+0.03125$$
$$161.15625$$
$$\therefore (241.12)_8=(161.15625)_{10}$$

Hexa Decimal Number System:

- It is also a positional weighted number system
- Base or radix is 16.
- Symbols → 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F,
  ($A=10$, $B=11$, $C=12$, $D=13$, $E=14$, $F=15$.)
- Representation – $(\text{Number})_{16}$
- If radix is 5 the symbols are – 0, 1, 2, 3 4.

→ Convert $(26.52)_{10}$ into Hexa Decimal

Integer Part

$$\begin{array}{r|l} 16 & 26 \\ \hline 16 & 1-A \\ \hline & 0-1 \end{array}\uparrow$$

Fractional Part

$0.52\times 16=8.32-8$
$0.32\times 16=5.12-5$
$0.12\times 16=1.92-1$
$0.92\times 16=14.72-E$ ↓

$\therefore (26.52)_{10}=(1A.851E)_4$

(+) 0
1
← 1

→ Convert $(AF3)_{16}$ into Decimal

$$\begin{array}{ccc} A & F & 3 \\ \downarrow & \downarrow & \\ 10 & 15 & \end{array}$$

$= 10\times16^2 + 15\times16^1 + 3\times16^0$

$= 10\times256 + 15\times16 + 3\times1$

$= 2560 + 240 + 3$

$= (2803)_{10}$

→ <u>Internal conversions</u>

— Convert $(10110)_2$ into octal

→ we know that $8 = 2^3$

— convert binary number in terms of groups of the power (overhere 3)

— Always add zero to the left side so that there is no increase in weight.

$$\begin{array}{c|c} 0\,1\,0 & 1\,1\,0 \\ 2 & 6 \end{array}$$

$\therefore (10110)_2 = (26)_8$

→ $(1101110)_2$ to $(\quad)_8$

$$\begin{array}{c|c|c} 0\,0\,1 & 1\,0\,1 & 1\,1\,0 \\ 1 & 5 & 6 \end{array}$$

$\therefore (1101110)_2 = (156)_8$

→ convert $(101101.11011)_2$ into octal

$$\begin{array}{c|c|c|c} 1\,0\,1 & 1\,0\,1 & .\;1\,1\,0 & 1\,1\,0 \\ 5 & 5 & .\;6 & 6 \end{array}$$

$\therefore (101101.11011)_2 = (55.66)_8$

→ Convert $(356)_8$ into binary

$$\begin{array}{ccc} 3 & 5 & 6 \\ | & | & | \\ 011 & 101 & 110 \end{array}$$

– Convert each digit to binary only upto 3-bits.

$\therefore (356)_8 = (011101110)_2$

→ Convert $(101101 \cdot 11011)_2$ to Hexa decimal

$$\begin{array}{c|c|c} 0010 & 1101 \cdot 1101 & 1000 \\ 2 & D \cdot D & 8 \end{array}$$

$\therefore (101101 \cdot 11011)_2 = (2D \cdot D8)_H$

→ Convert $(AF3)_{16}$ to binary

$$\begin{array}{ccc} A & F & 3 \\ \downarrow & \downarrow & \downarrow \\ 1010 & 1111 & 0011 \end{array}$$

$(AF3)_{16} = (101011110011)_2$

→ Convert $(AF3)_{16}$ into octal

– First convert Hexa decimal number into binary then convert binary into octal.

$$\begin{array}{cccc} A & & F & 3 \\ 101 & 011 & 110 & 011 \\ 5 & 3 & 6 & 3 \end{array}$$

$\therefore (AF3)_{16} = (5363)_8$

→ Convert $(356 \cdot 12)_8$ into Hexa decimal

– First convert octal number into binary then convert that binary number into hexadecimal.

$$\begin{array}{ccccc} 3 & 5 & 6 & \cdot & 1 \quad 2 \\ 0000 & 1110 & 1110 & \cdot & 0010 \quad 1000 \\ 0 & E & E & \cdot & 2 \quad 8 \end{array}$$

$\therefore (356 \cdot 12)_8 = (EE \cdot 28)_{16}$

→ Find the radix for the given number:

$$(35)_6 = (43)_x$$

– Multiply number with base

$$\begin{matrix} 3 & 5 & = & 4 & 3 \\ 6^1 & 6^0 & & x^1 & x^0 \end{matrix}$$

$$3\times6^1+5\times6^0 = 4\times x^1+3\times x^0$$
$$3\times6+5\times1 = 4x+3\times1$$
$$18+5 = 4x+3$$
$$23 = 4x+3$$
$$20 = 4x$$
$$x = 20/4 = 5$$

→ Convert $(101101)_2$ into $(\quad)_5$

– First convert binary to decimal then convert it into base 5.

$$(101101)_2 = 1\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0$$
$$= 1\times32\ 0\times16+1\times8+1\times4+0\times2+1\times1$$
$$= 32+0+8+4+0+1$$
$$= (45)_{10}$$

$$\begin{array}{r|l} 5 & 45 \\ 5 & 9-0 \\ 5 & 1-4 \\ & 0-1 \end{array}$$

$\therefore (101101)_2 = (140)_5$

Binary Arithmetic

2-bit add

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

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

→ Add 13, 15 in binary

$13 \rightarrow 1101$
$15 \rightarrow (+) 1111$
$28 \rightarrow 11100$

$\therefore 13+15 = (11100)_2$

→ subtract 11, 7 in binary

$11 \rightarrow 1011$
$7 \rightarrow 0111$
$4 \rightarrow 0100$

Signed and Unsigned Numbers in Binary

- Unsigned binary numbers represent only magnitude.
- Signed binary numbers represent both magnitude and sign (direction).
- In signed number, the Most significant Bit (MSB) represent the sign of the number.

If MSB = 0 → Positive Number
If MSB = 1 → Negative Number

- In general an n-bit signed magnitude integer lies within the range of $-(2^{n-1}-1)$ through $(2^{n-1}-1)$

Number indications in signed & Unsigned:

Number		Signed	Unsigned
−0	→	1000 0000	0000 0000
+0	→	0000 0000	0000 0000
−3	→	1000 0011	0000 0011
+3	→	0000 0011	0000 0011

- In case of unsigned 8-bit binary number the decimal range is 0 to 255.
- In case of Signed number the largest magnitude is reduced from 255 to 127. So the range is from −127 to 0 to +127.

Complements: One's Complement: The number that results by changing all ones to zeroes and all zeroes to ones is called one's (1's) complement.

Calculate the 1's complement of $(101101)_2$

Given Number → 101101

1's Complement → 010010

→ **Two's complement: (2's)**

- 2's complement is a binary number that results when we add one to 1's complement.

2's complement = 1's complement + 1

Ex: Given number → 101101

1's Complement → 010010

+1

2's Complement → 010011

→ **Nine's complement:** subtracting each & every digit from nine gives 9's complement.

Ex: Calculate the 9's complement of 543

$$\begin{array}{r} 999 \\ (-)\ 543 \\ \hline 456 \end{array}$$

543 — Given number

456 — 9's complement

→ **Ten's Complement:** Adding 1 to the nine's complement will give 10's complement.

10's complement = 9's complement + 1

Ex: Calculate the 10's Complement of $(624)_{10}$

$$\begin{array}{r} 999 \\ (-)\ 624 \\ \hline 375 \\ +1 \\ \hline 376 \end{array}$$

624 — Given number

375 — 9's complement

376 — 10's complement

- In general we have r's (radix's complement) complement of any given number the formula used is

$$r^n - N$$

where r — radix of given number

n - number of digits in number

N - the given number (in any number system)

→ $(r-1)$'s - (Diminished radix) Complement: To find the diminished radix complement of any given number system we use the formula → $(r^n - 1) - N$

Ex: Find the r's & $(r-1)$'s complement for the given number $(345)_6$

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

$r = 6, \; n = 3, \; N = 345$

$$\therefore r\text{'s complement} = 6^3 - 345$$
$$= 216 - 345$$
$$= -129$$

$$(r-1)\text{'s complement} = (r^n - 1) - N$$
$$= (6^3 - 1) - 345$$
$$= (216 - 1) - 345$$
$$= 215 - 345$$
$$= -130$$

→ Subtract 1011011 from 1100101

```
    1100101   —  101
(-) 1011011   —   91
    0001010   —   10
```

→ Subtract $(101)_{10}$ $(91)_{10}$

- In this case we have to subtract ~~Larger~~ Smaller number from larger number:

Procedure: 1) Determine the 1's complement of the smallest number

2) Add 1's complement of the smaller number to the larger number

3) If carry is present, add it to the result in LSB position. This carry is called as end-around carry.

91 – is a smaller number

91 → 1011011

1's Complement → 0100100

Larger Number → 1100101

1's complement → (+) 0100100

$\boxed{1}$0001001

+1

0001010 = 10

⟹ Subtraction of larger number from smaller number

Procedure: 1) Determine 1's complement of larger number

2) Add 1's complement of the larger number to smaller number

3) To get the answer in true form, take 1's complement of the result and assign negative sign to the result.

Ex: Subtract $(101)_{10}$ from $(91)_{10}$

91 – 101 = –10

1's complement of 101 → 1100101

0011010

Add 1's complement to $(91)_{10}$

91 → 1011011

1's complement of 101 → (+) 0011010

1110101

→ we don't have carry, so use step 3.

result = 1110101 → 1's Complement = 0001010

Final result = 1010

Add sign (-ve) = -1010 = -10

2's complement Subtraction:

⇒ Subtraction of smaller number from larger number

Steps: 1) Determine 2's complement of smaller number

2) Add the 2's complement of smaller to larger

3) In this case, discard the carry.

Ex: 76 - 42 = 34

Sol: Find 2's complement of 42

Given number → 42 = 0101010

1's complement of 42 = 1010101
+1

2's complement → 1010110

Add 2's complement to larger number

2's complement of 42 = 1010110

Given larger number 76 = 1001100
(+)

← ①0100010

Discard carry

Final result = $(100010)_2$ = ~~(42)~~$_{10}$ $(34)_{10}$ ✓

⇒ Subtraction of larger number from smaller:

Steps: 1) Determine 2's complement of larger number

2) Add 2's complement of larger to smaller

3) To get answer in true form, take 2's complement of result and assign negative sign.

Ex: Subtract 42 - 76 = -34

Given larger number → 76 = 1001100

1's complement → = 0110011
+1

2's complement → 0110100

Given smaller number 42 = 0101010

2's complement of 76 = (+) 0110100

No carry ← 1011110

Result → 1011110

1's complement → 0100001

+1

0100010

Final result = $(100010)_2 = (34)_{10}$

Add -ve sign = $-(100010)_2 = -34$

Codes: we are having different types of codes

1) weighted
   - Binary
   - BCD
     - 8421, 2421, 5211, 5421

2) Non-weighted
   - Excess-3
   - Gray
   - 5-bit BCD

3) Reflective
   - 2421
   - 5211
   - Excess-3
   - Gray

4) Sequential
   - 8421
   - Excess-3

5) Alphanumeric
   - ASCII
   - EBCDIC

6) Error detecting & correcting
   - Parity
   - Hamming

→ In this gray code is used in k-map simplifications. the advantage of the gray code over the straight binary number sequence is that only one bit in the code group changes in going from one number to the next. For example, in going from 7 to 8, the gray code changes from 0100 to 1100. Only the first bit changes, from 0 to 1, the other

3 bits remain the same. By contrast, with binary numbers the change from 7 to 8 will from 0111 to 1000, which causes all four bits to change values.

- It is a special case of unit distance.
- Bit patterns for 2 consecutive numbers differ in only 1-bit position.

Ex: convert the number $(10101101)_2$ to Gray code

Given Binary: 1 0 1 0 1 1 0 1

Gray code : 1 1 1 1 1 0 1 1

→ Convert 11111011 number from Gray to binary

Given Gray code : 1 1 1 1 1 0 1 1

Binary : 1 0 1 0 1 1 0 1.

Booolean Algebra:

- Boolean Algebra, like any other deductive mathematical system, may be defined with a set of elements, a set of operators, and a no. of unproved axioms or postulates. A set of elements is any collection of objects, usually having a common property.

Properties and theorems Boolean Algebra

1) $x+0=x$ 2) $x\cdot 1=x$

3) $x+x'=1$ 4) $x\cdot x'=0$

5) $x+x=x$ 6) $x\cdot x=x$

7) $x+1=1$ 8) $x\cdot 0=0$

9) Involution theorem: $(x')'=x$

Commutative law $x+y = y+x$

$xy = yx$

11) Associative law: $x+(y+z) = (x+y)+z$

$x(yz) = (xy)z$

12) Distributive law: $x(y+z) = xy + xz$

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

13) DeMorgan's law: $(x+y)' = x' \cdot y'$

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

14) Absorption law: $x+yx = x$ $\quad x(x+y) = x$

→ Principle of Duality: In boolean algebra, If one expression can be obtained from the other in each pair by replacing every 0 with 1, every 1 with 0, every (+) with (·), and every (·) with (+), then the pair of expression satisfying this property is called dual expression. This characteristic of Boolean algebra is called the principle of duality.

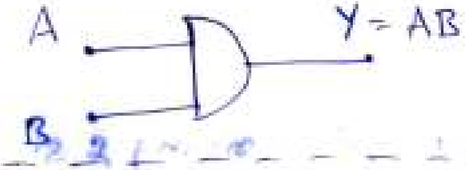
Logic Gates

- Boolean algebra is used in describing & simplifying logic circuits. Simplification of boolean logic expressions is very important because it reduces the hardware required to design a specific system. The boolean expression corresponding to a given gate network can be derived by systematically progressing from the i/p to the o/p of gates. The gating or logic network can be formed by interconnecting the OR, AND and NOT gates.

- A logic gate is an electronic ckt which makes logical decisions. To arrive at these decisions, the most common logic gates used are OR, AND, NOT, NAND & NOR gates. Among these, AND, OR, NOT gates are called as basic gates. The NAND & NOR gates are called as universal gates because all the gates can be realized by using only these gates. The other two gates are EX-OR & EX-NOR.

AND Gate: The AND gate performs logical multiplication, commonly known as AND function. The AND gate has two or more i/ps & a single output. The o/p of an AND gate is high only when all the i/ps are high. Even if any one of inputs is low, the o/p will be low. If A & B are the i/p variables of an AND gate & Y is its o/p, then $Y = A \cdot B$

Logic symbol

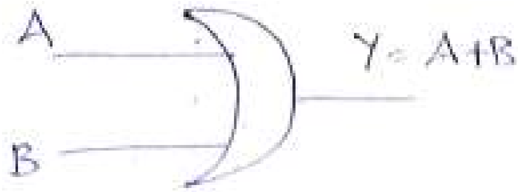


Truth Table

Inputs		Output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate: The OR gate performs logical addition, commonly known as OR function. The OR gate has two or more i/ps & only one o/p. The operation of OR gate is such that a high on the o/p is produced when any of the i/ps is high. The o/p is low only when all the i/ps are low. If A & B are the i/p variables of an OR gate & Y is its o/p, then $Y = A + B$.

Logic Symbol

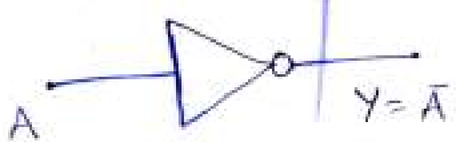


Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT gate (Inverter): The NOT gate performs the basic logical function called inversion or complementation. the purpose of this gate is to convert one logic level into the opposite logic level. It has one i/p & one o/p. When a high level is applied to an inverter, a low level appears at its o/p & vice-versa.

Logic Symbol

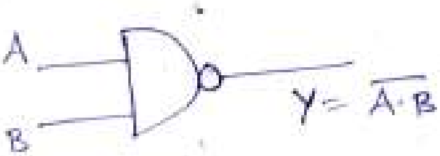


Truth Table

Input	output
A	Y
0	1
1	0

NAND Gate: NAND is a contraction of the NOT-AND gates. It has 2 or more i/ps & only one o/p. i.e. $Y = \overline{A \cdot B}$. When all the i/ps are high, the o/p is low. If any one or both the i/ps are low, then the o/p is high.

Logic Symbol

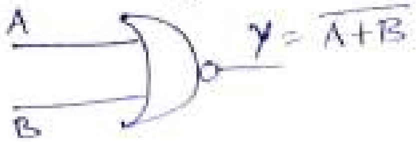


Truth Table

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

NOR Gate: NOR is a contraction of NOT-OR gate. It has 2 or more i/ps and only one o/p, i.e. $Y = \overline{A+B}$. the o/p is high only when all the i/ps are low. If any one or both the i/ps are high, then the o/p is low.

Logic symbol



Truth Table

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Exclusive-OR (Ex-OR) Gate: An Ex-OR gate is a gate with 2 or more i/ps & one o/p. The o/p of a 2 i/p Ex-OR gate assumes a high state if one and only one i/p assumes a high state. This is equivalent to saying that the o/p is high if either i/p A or i/p B is high exclusively, and low when both are 1 or 0 simultaneously.

Logic symbol

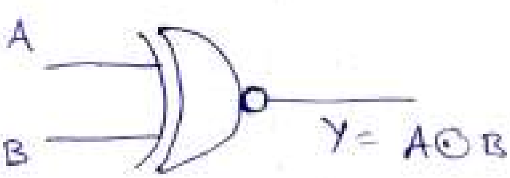
$A \oplus B = \bar{A}B + A\bar{B}$

Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive-NOR (Ex-NOR) gate: It has two or more i/ps and one o/p. The o/p of a two i/p Ex-NOR gate assumes a high state if both i/ps assume the same logic state or have an even no. of 1s, and its o/p is low when the i/ps assume different logic states or have an odd no. of 1's

Logic symbol



$A \odot B = \overline{A \oplus B} = AB + \bar{A}\bar{B}$

Truth Table

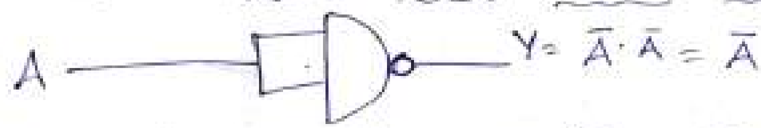
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Universal Gates/Universal Building Blocks:

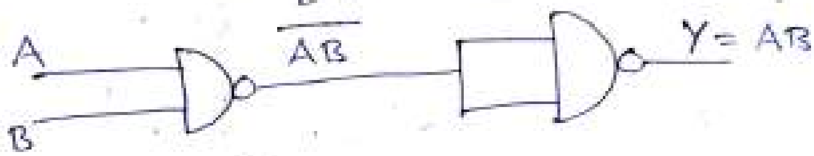
NAND and NOR gates are called Universal gates or Universal building blocks because both can be used to implement any gate like AND, OR and NOT gates or any combination of these basic gates.

Realization of logic gates using NAND gates

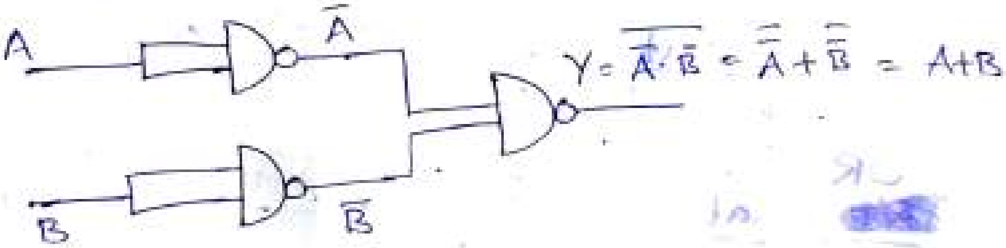
NOT gate:-



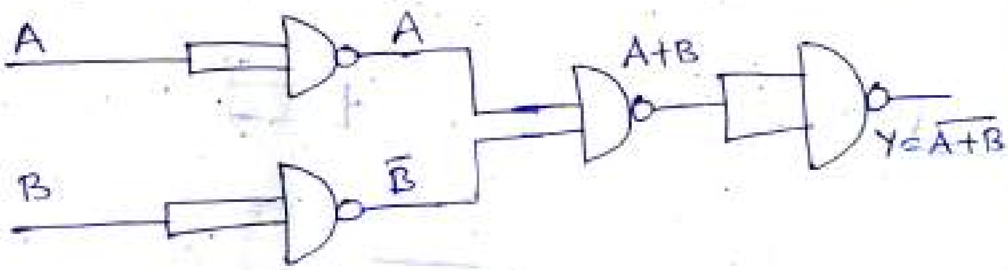
AND Gate:-



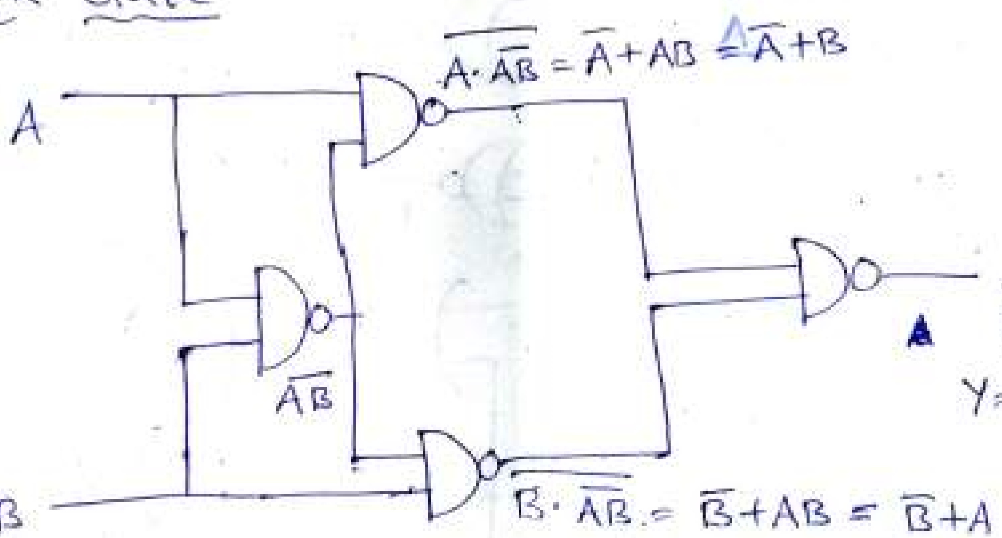
OR Gate:-



NOR Gate:-



EX-OR Gate

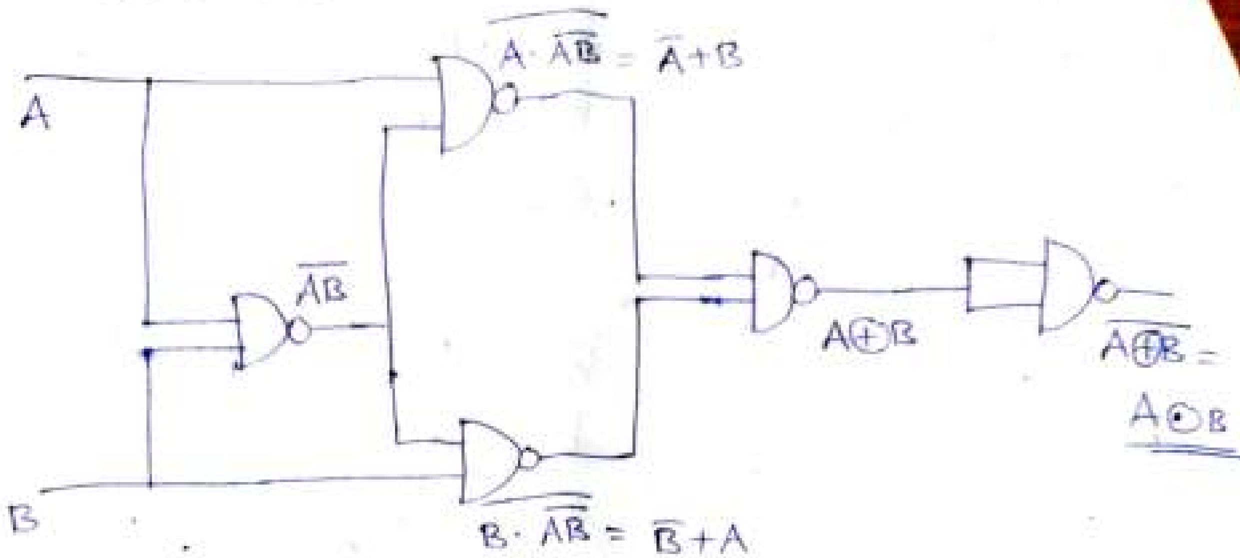


$\overline{\overline{A}+B \cdot \overline{B}+A}$

$Y = A\overline{B} + B\overline{A}$

$= A \oplus B$

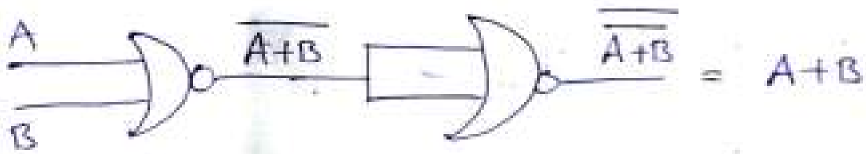
EX-NOR Gate:



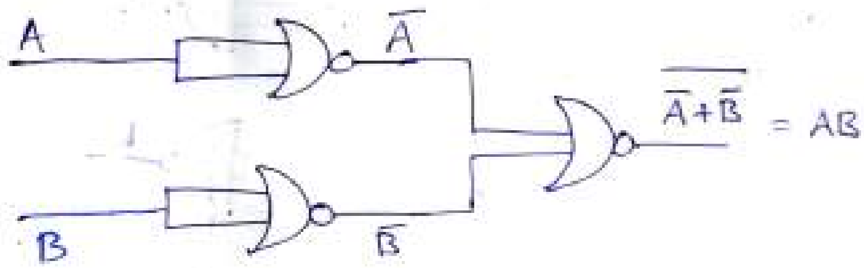
⇒ Realization of logic gates using NOR Gates

→ NOT Gate:

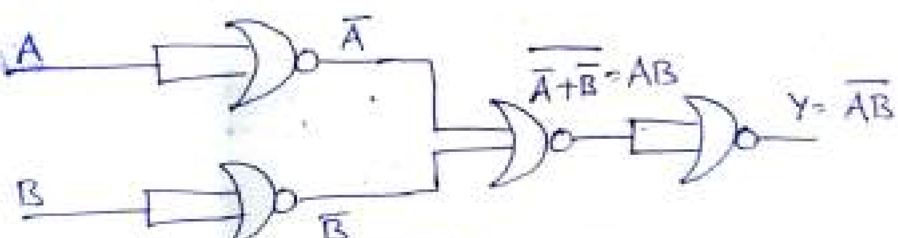
OR Gate



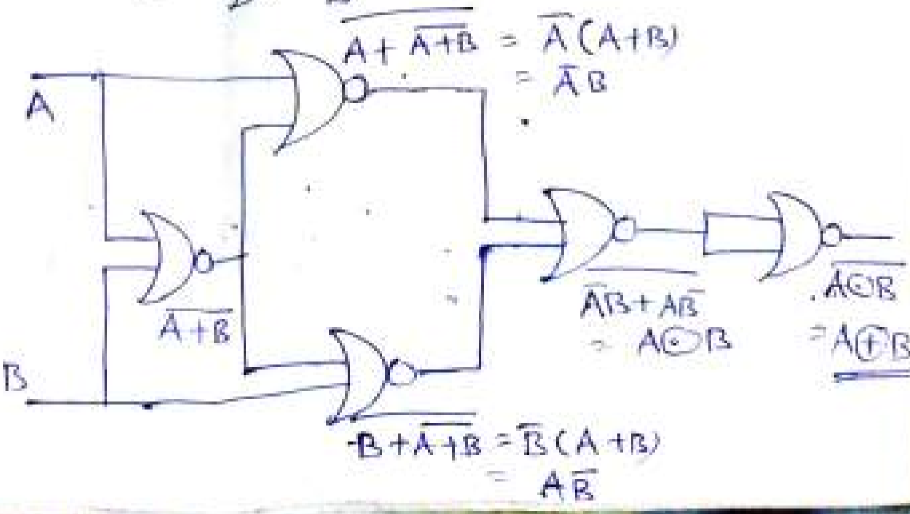
AND Gate:



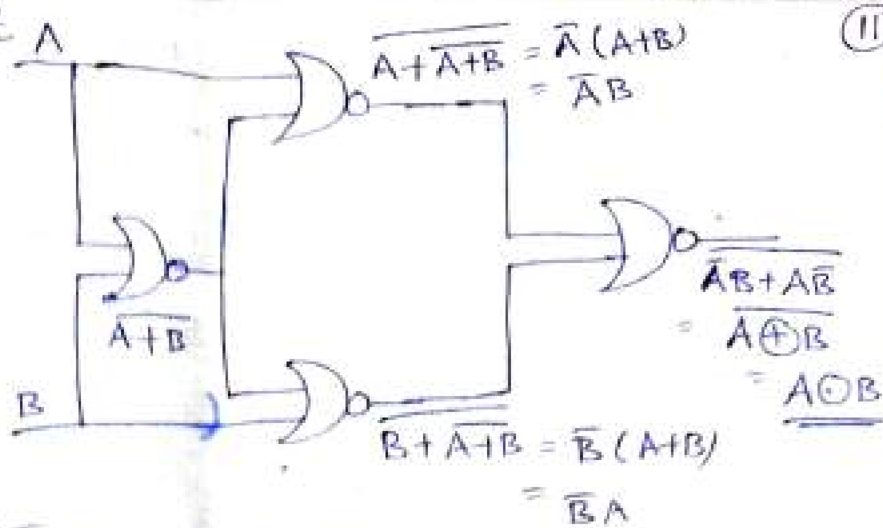
NAND Gate:



EX-OR Gate:



Ex-NOR Gate



Problem: $(A+B)(\bar{A}+B)(A+\bar{B})$ simplify the expression and write the duality of the function.

Sol:

$$(A+B)(\bar{A}+B)\cdot(A+\bar{B})$$
$$= (A\cdot\bar{A} + AB + \bar{A}B + B\cdot B)(A+\bar{B})$$
$$= (0 + AB + \bar{A}B + B)(A+\bar{B})$$
$$= \bar{A}B + B(1+A)(A+\bar{B})$$
$$= \bar{A}B + B(A+\bar{B}) \quad \{\because 1+A=1\}$$
$$= B(1+\bar{A})(A+\bar{B}) \quad \{\because$$
$$= B(A+\bar{B}) = AB+0$$
$$= \underline{AB}$$

$\Rightarrow (A+B)(\bar{A}+B)(A+\bar{B}) = \big((A+B)(\bar{A}+B)(A+\bar{B})\big)_d = (AB)_d$

$$= AB+\bar{A}B+A\bar{B} = A+B$$
$$= B(A+\bar{A})+A\bar{B} = A+B$$
$$= B+A\bar{B} = A+B$$
$$= (B+A)(B+\bar{B}) = A+B$$
$$= (B+A) \doteq A+B$$
$$\therefore \underline{A+B = A+B}$$

$\Rightarrow$ Simplify and write the duality of the given expression $(A+C)(A+D)(B+C)(B+D)$

Sol: $(A+C)(A+D)(B+C)(B+D)$

$$= (A \cdot A + AD + AC + CD)(B \cdot B + BD + BC + CD)$$

$$= (A + AD + AC + CD)(B + BD + BC + CD)$$

$$= (A(1+D) + AC + CD)(B(1+D) + BC + CD)$$

$$= (A + AC + CD)(B + BC + CD)$$

$$= (A(1+C) + CD)(B(1+C) + CD)$$

$$= (A + CD)(B + CD)$$

$$= \underline{AB + CD}$$

$\Rightarrow$ $(A+C)(A+D)(B+C)(B+D) = AB + CD$

$$((A+C)(A+D)(B+C)(B+D))_d = (AB+CD)_d$$

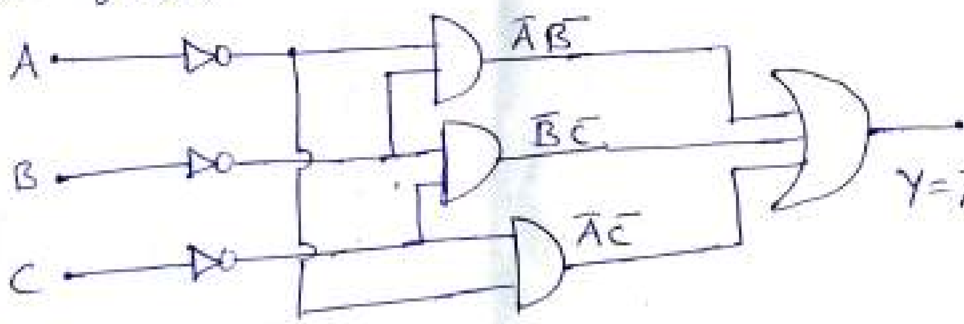
$$= AC + AD + BC + BD = (A+B)(C+D)$$

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

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

$\Rightarrow$ Realize the expression $Y = \overline{B}\,\overline{C} + \overline{A}\,\overline{C} + \overline{A}\,\overline{B}$ using basic gates.

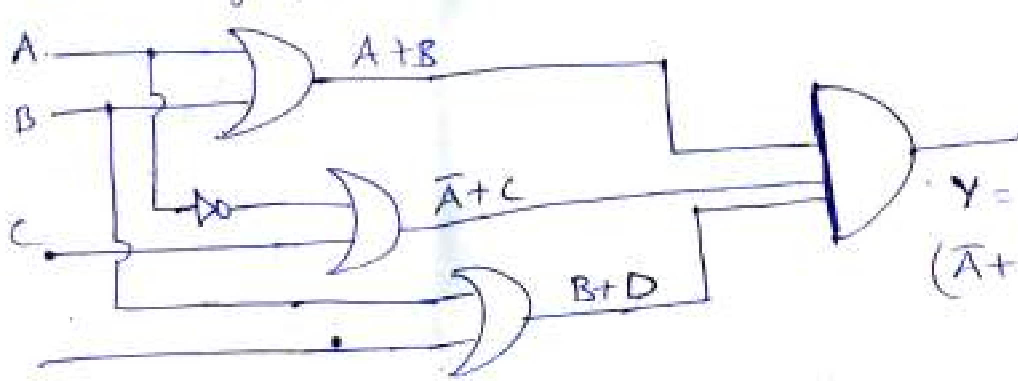
Sol:



$Y = \overline{A}\,\overline{B} + \overline{B}\,\overline{C} + \overline{A}\,\overline{C}$

$\Rightarrow$ Realize the logic expression $Y = (A+B)(\overline{A}+C)(B+D)$ using basic gates.

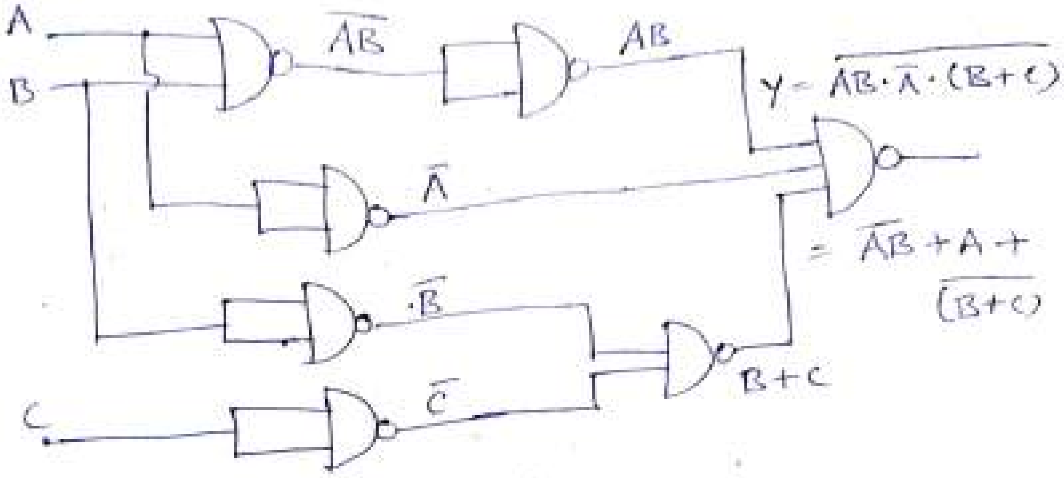
Sol:



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

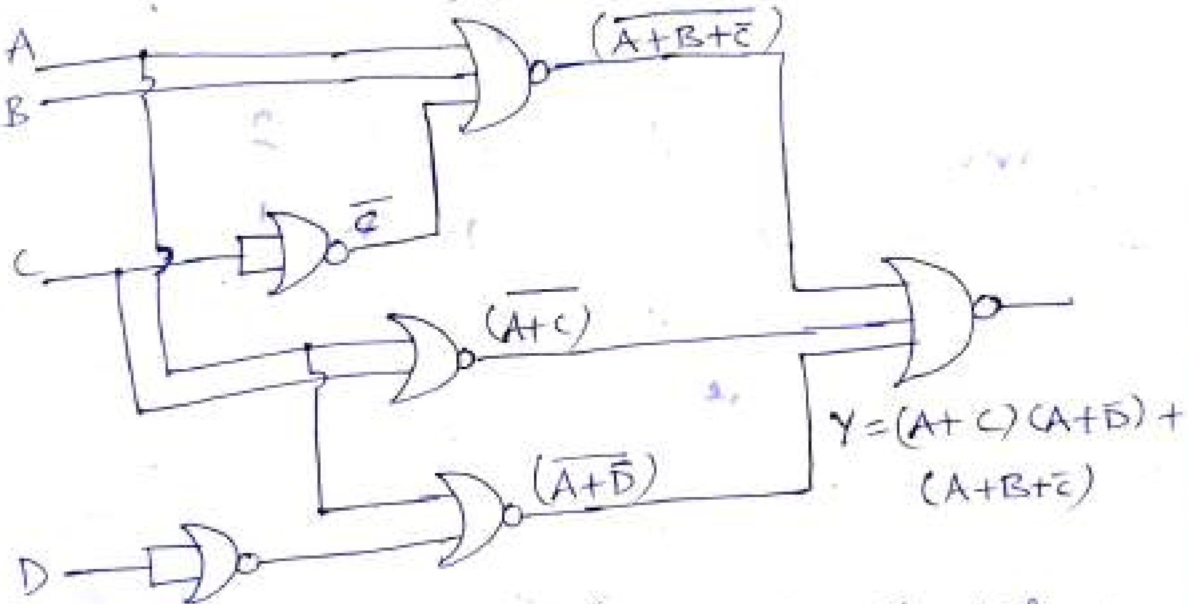
⇒ Implement $Y = AB + A + (\overline{B+C})$ using NAND gates only.

Sol:

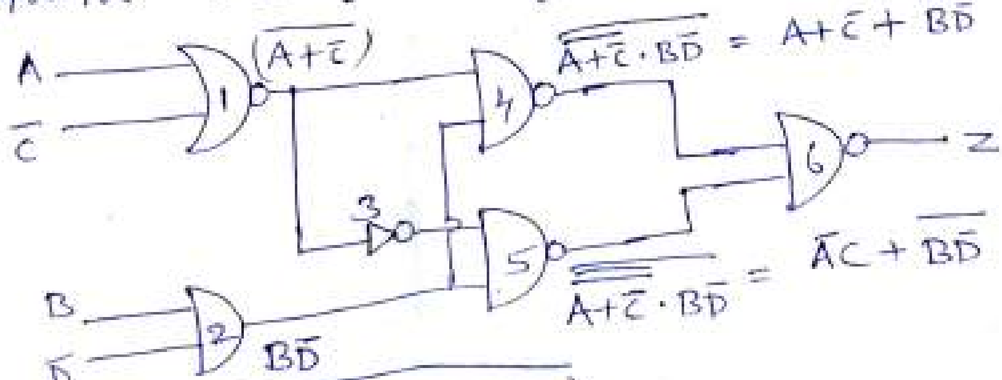


⇒ Realize $Y = (A+C)(A+\bar{D})(A+B+\bar{C})$ using NOR gates.

Sol



⇒ Simplify the given logic circuit below.



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

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

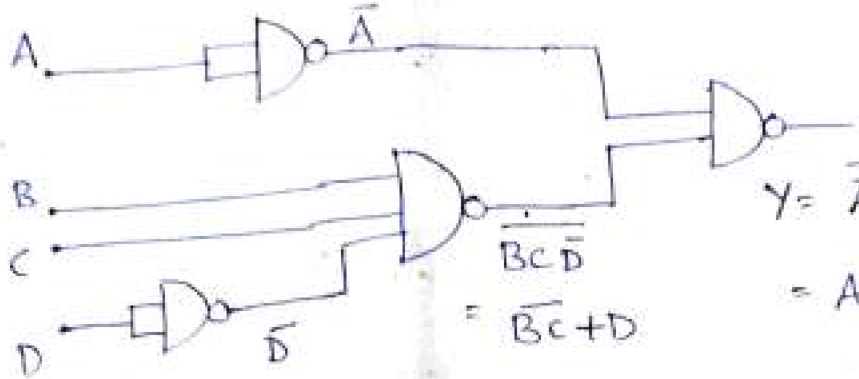
$$= (\bar{A}C)(B\bar{D}) + (A+\bar{C})(B\bar{D})$$

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

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

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

⇒ Simplify the given logic circuit below:



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

$= A + BC\bar{D}$

$\therefore Y = A + BC\bar{D}$

Canonical and Standard Forms:

- To implement a logic functions in less no. of gates, we have to minimize literals or variables and, the number of terms.
- Visually, literals and terms are arranged in one of the two standard forms:

i) sum of products (SOP)

ii) Product of sum terms (POS)

SOP: It is a group of product terms OR together

POS: It is a group of sum terms AND together.

Ex: $AB + B\bar{C}$ — SOP

$(\bar{A}+\bar{B})(\bar{B}+C)$ — POS

⇒ In the truth table is Y = (0 1 1 0 1 0 1 1) then represent SOP & POS forms.

Y =	0	1	1	0	1	0	1	1
i/ps →	0	1	2	3	4	5	6	7

$\therefore SOP = \Sigma m(1,2,4,6,7)$

$POS = \Pi M(0,3,5)$

Table:

A	B	C	SOP (Min terms)	POS (Max. terms)
0	0	0	$m_0 = \bar A \bar B \bar C$	$M_0 = A+B+C$
0	0	1	$m_1 = \bar A \bar B C$	$M_1 = A+B+\bar C$
0	1	0	$m_2 = \bar A B \bar C$	$M_2 = A+\bar B+C$
0	1	1	$m_3 = \bar A B C$	$M_3 = A+\bar B+\bar C$
1	0	0	$m_4 = A \bar B \bar C$	$M_4 = \bar A+B+C$
1	0	1	$m_5 = A \bar B C$	$M_5 = \bar A+B+\bar C$
1	1	0	$m_6 = A B \bar C$	$M_6 = \bar A+\bar B+C$
1	1	1	$m_7 = A B C$	$M_7 = \bar A+\bar B+\bar C$

Problem: Simplify the expression: $\overline{\overline{A\bar B + ABC} + A(B+A\bar B)}$

Sol: $\overline{\overline{A\bar B + ABC} + A(B+A\bar B)} = \overline{\overline{A(\bar B + BC)} + A(B+A)}$

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

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

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

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

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

$$= \overline{(\bar A + B\bar C) + A} \quad \{\because (A+B)(A+C) = A+BC$$

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

$$= \overline{1 + B\bar C}$$

$$= \bar 1 \quad \{\because 1 + X = 1\}$$

$$= \underline{\underline{0}}$$

Problem: Simplify the expression: $Y = \bar A \bar B \bar C + \bar A B \bar C + A \bar B \bar C + A B \bar C$

Sol: $Y = \bar A \bar B \bar C + \bar A B \bar C + A \bar B \bar C + A B \bar C$

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

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

Prob: Prove the boolean expression: $(A+B)(\bar{A}\bar{C}+C)\overline{(\bar{B}+AC)} = \bar{A}B.$

Sol: $(A+B)(\bar{A}\bar{C}+C)\overline{(\bar{B}+AC)} = (A+B)(\bar{A}\bar{C}+C)(\bar{\bar{B}} \cdot \overline{AC})$

$$= (A+B)(\bar{A}\bar{C}+C)(B \cdot \overline{AC})$$
$$= [A \cdot \bar{A}\bar{C} + AC + B\bar{A}\bar{C} + BC][B(\bar{A}+\bar{C})]$$
$$= (AC + \bar{A}\bar{C}B + BC)(B\bar{A} + B\bar{C})$$
$$= AC \cdot B\bar{A} + AC \cdot B\bar{C} + \bar{A}\bar{C}B \cdot B\bar{A} + \bar{A}\bar{C}B \cdot B\bar{C} + BC \cdot B\bar{A} + BC \cdot B\bar{C}$$
$$= 0 + 0 + \bar{A}B\bar{C} + \bar{A}\bar{C}B + \bar{A}BC + 0$$
$$= \bar{A}B(\bar{C}+\bar{C}+C)$$
$$= \underline{\bar{A}B}$$

Standard SOP: Sum of product expression is referred to as canonical or standard SOP expression if every product term involves every variable in complement or uncomplement form.

Standard POS: Product of sum expression is referred to as a canonical or standard POS expression if every sum term involves every variable in complement or uncomplement form.

Prob: Convert the given expression into std SOP.

$$AB + BC$$

Sol: — It consists of 3 variables A, B, C. In first term C is missing & in second term A is missing.

– We know that $A+\bar{A}=1$ & $C+\bar{C}=1$, hence we multiply them to second and first term respectively.

Given Eqn — $AB + BC$

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

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

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

$= 111 \quad 110 \quad 011$

$\quad 7 \quad 6 \quad 3$

$\therefore f = \Sigma m(3,6,7)$

Prob: convert given expression, into standard POS.

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

Sol: It consists of 3 variables A, B, C. We know that $A.\bar{A}=0$. Find out the missing terms in first C, in second B, in third A. Hence add $C\bar{C}$, $B\bar{B}$, $A\bar{A}$ respectively in 1, 2nd and 3rd terms

Given eqn — $(A+B)(A+C)(B+\bar{C})$

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

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

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

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

$= 000 \quad 001 \quad 010 \quad 101$

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

$\quad 0 \quad 1 \quad 2 \quad 5$

$\therefore F = \Pi M(0,1,2,5)$

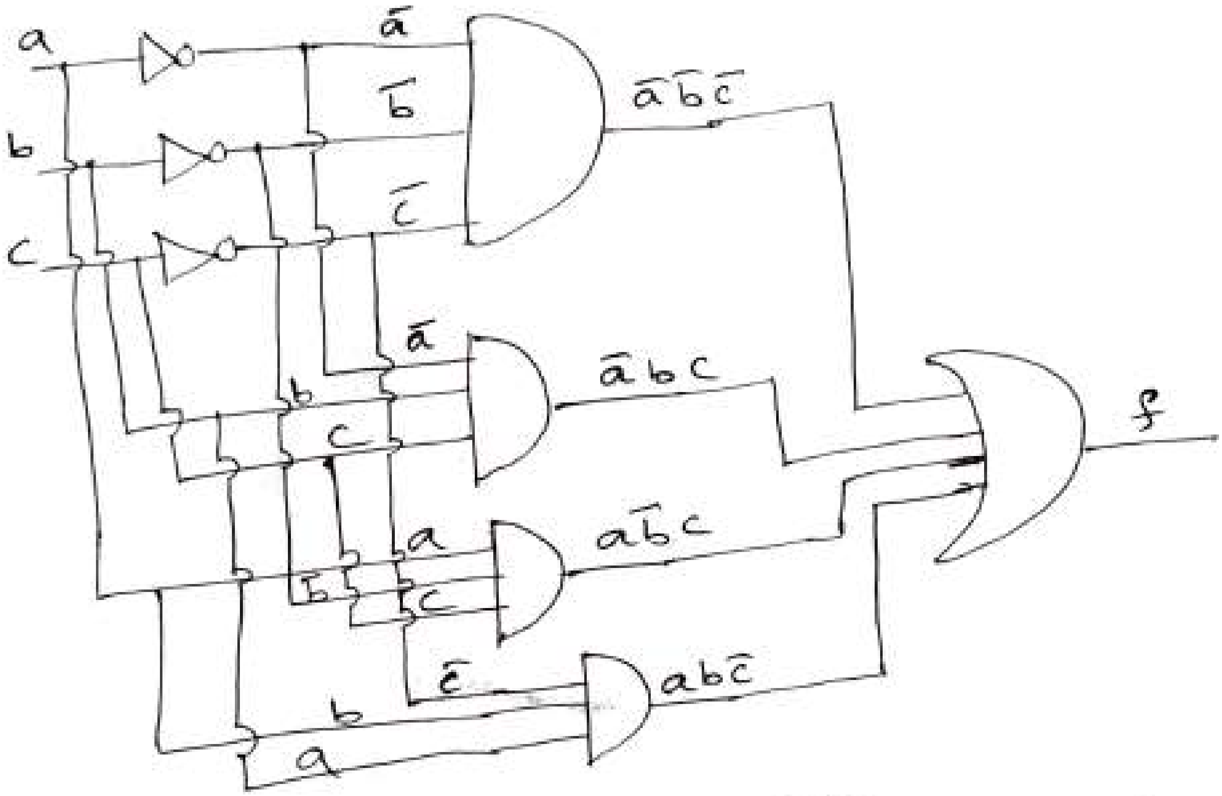
Prob: Represent the given function in terms of expression. Simplify it and draw the logic ckt diagram. $f = \Sigma m(0,3,5,6)$

Sol: $f(a,b,c) = \Sigma m(0,3,5,6)$

$f = m_0 + m_3 + m_5 + m_6$

$f = 000 \mid 011 \mid 101 \mid 110$

$f = \bar{a}\bar{b}\bar{c} + \bar{a}bc + a\bar{b}c + ab\bar{c}$



- If we minimize the expression,

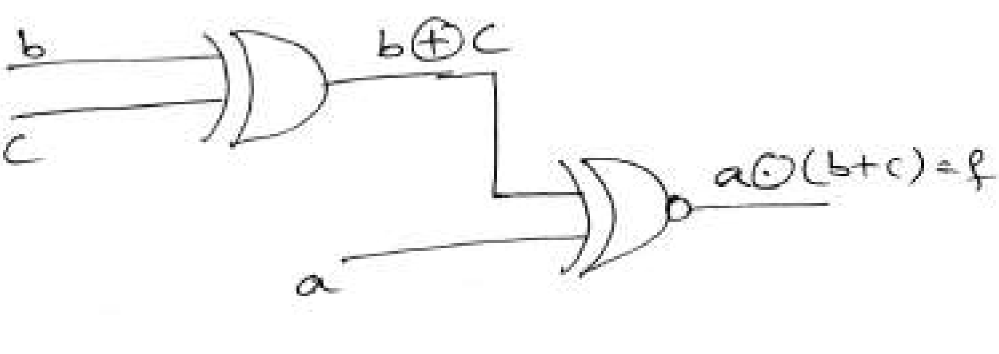
$f = \bar{a}\bar{b}\bar{c} + \bar{a}bc + a\bar{b}c + ab\bar{c}$, we will get

$= \bar{a}(\bar{b}\bar{c} + bc) + a(\bar{b}c + b\bar{c})$

$= \bar{a}(b \odot c) + a(b \oplus c)$

$= \bar{a}(\overline{b \oplus c}) + a(b \oplus c)$

$= a \odot (b \oplus c)$



The Karnaugh (K-map) Map Method:

— The simplification of the switching functions using Boolean laws & theorems becomes complex with the increase in the no. of variables and terms. The K-map technique provides a systematic method for simplifying and manipulating switching expressions. In this technique, the information contained in a truth table or available in the POS or SOP form is represented on the K-map. The K-map is actually a modified form of a truth-table. In an n-variable K-map, there are 2^n cells. Each cell corresponds to one combination of n variables. Therefore, for each row of the truth table, i.e. for each min term and for each maxterm, there is one specific cell in the K-map. The K-maps for 2, 3 and 4 variables are shown in fig. below. The decimal codes corresponding to the combination of variables are given inside the cells. The variables have been marked as A, B, C & D, and the binary numbers formed by them are taken as AB, ABC and ABCD for 2, 3 and 4 variables respectively.

A \ B	0	1
0	0	1
1	2	3

2-variable

A \ BC	00	01	11	10
0	0	1	3	2
1	4	5	7	6

3-variable

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

4-variable

Fig. Karnaugh Maps

- The 3 and 4 variable K-maps show that the column and row headings, used in representing the cells, are cyclic or unit distance code which result in adjacent cells, differing in just one variable. This helps the grouping of the adjacent cells and in their simplification. The left and right most cells of the 3-variable K-map are adjacent. For example, the cells 0 and 2 are adjacent, and the cells 4 & 6 are adjacent. This is because each pair differs in just a single variable. In the 4-variable K-map, the cells to the extreme left and right as well as those, at the top and bottom most positions are adjacent.

- Simplification is based on the principle of combining the terms present in adjacent cells. The 1s in the adjacent cells can be grouped by drawing a loop around those cells following the given rules:

1) Construct the K-map and enter the 1s in those cells corresponding to the combinations for which function value is 1, then enter the 0s in the other cells.

2) Examine the map for 1s that cannot be combined with any other 1 cells & form groups with such single 1.

3) Next, look for those 1s which are adjacent to only one other 1 and form groups containing only 2 cells & which are not part of any group of 4 or 8 cells. A group of 2 cells is called a pair.

4) Group of 1s which results in group of 4 cells

but are not part of an 8-cells group. A group of 4 cells is called a quad.

5) Group the 1s which results in groups of 8 cells. A group of 8 cells is called an octet.

6) Form more pairs, quads and octets to include these 1s that have not yet been grouped, and use only a minimum no. of groups. These can be overlapping of groups if they include common 1s.

7) Omit any redundant group.

8) Form the logical sum of all the terms generated by each group.

- When one or more than one variable appear in both complemented & uncomplemented form within a group, then that variable is eliminated from the term corresponding to that group.
- Variables that are the same for all the cells of the group must appear in the term corresponding to that group.
- A larger group of 1s eliminate more variables. To be precise, a group of 2 eliminates one variable, a group of 4 eliminates two variables, similarly a group of 8 eliminates three variables.

Prob: Simplify $Y = \Sigma m(2, 3)$ by using K-Map

A \ B	0	1
0	(0)	(1)
1	1 (2)	1 (3)

$Y = A$

⇒ Simplify $Y = \overline{A}\,\overline{B} + AB$ by using K-Map

A \ B	0	1
0	1	
1		1

$Y = \overline{A}\,\overline{B} + AB$

$\Rightarrow$ Simplify the boolean fn $F(x,y,z) = \Sigma m(2,3,4,5)$

x \ yz	00	01	11	10
0	$_0$	$_1$	1_3	1_2
1	1_4	1_5	$_7$	$_6$

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

$\Rightarrow$ Simplify the boolean fn $F(x,y,z) = \Sigma m(3,4,6,7)$

x \ yz	00	01	11	10
0	$_0$	$_1$	1_3	$_2$
1	1_4	$_5$	1_7	1_6

$$F = yz + xz'$$

$\Rightarrow$ Simplify the Boolean fn $F(x,y,z) = \Sigma m(0,2,4,5,6)$

x \ yz	00	01	11	10
0	1_0	$_1$	$_3$	1_2
1	1_4	1_5	$_7$	1_6

$$F = z' + xy'$$

$\Rightarrow$ Simplify the Boolean fn $F = A'C + A'B + AB'C + BC$

Sol: $A'C + A'B + AB'C + BC$

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

$$= \underset{(1)}{\underline{A'BC}} + A'B'C + \underset{(1)}{\underline{A'BC}} + A'BC' + AB'C + ABC + \underset{(1)}{\underline{A'BC}}$$

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

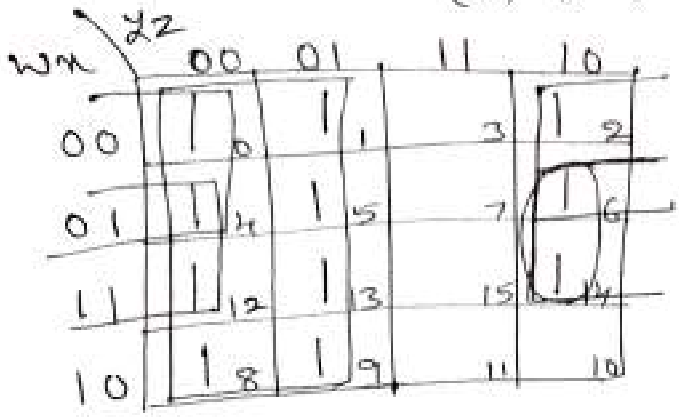
$A'BC$	$A'B'C$	$A'BC'$	$AB'C$	ABC
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
011	001	010	101	111
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
m_3	m_1	m_2	m_5	m_7

$F(A,B,C) = \Sigma m(1,2,3,5,7)$

A \ BC	00	01	11	10
0	$_0$	1_1	1_3	1_2
1	$_4$	1_5	1_7	$_6$

$$F = C + A'B$$

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



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

⇒ Simplify the Boolean function: $F = A'B'C' + B'CD' + A'BCD' + AB'C'$

<u>Sol</u>: First convert the above function into SSOP.

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

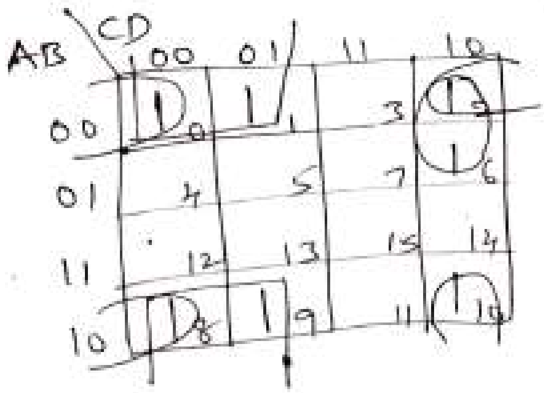
$$= A'B'C'(D+D') + B'CD'(A'+A) + A'BCD' + AB'C'(D+D')$$

$$= A'B'C'D + A'B'C'D' + A'B'CD' + AB'CD' + A'BCD' + AB'C'D + AB'C'D'$$

$$= 0001 \quad 0000 \quad 0010 \quad 1010 \quad 0110 \quad 1001 \quad 1000$$

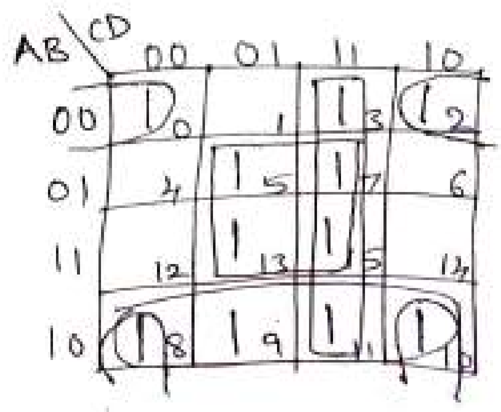
$$= m_1 \quad m_0 \quad m_2 \quad m_{10} \quad m_6 \quad m_9 \quad m_8$$

$$F = \Sigma m(0,1,2,6,8,9,10)$$



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

⇒ Simplify $F(A,B,C,D) = \Sigma m(0,2,3,5,7,8,9,10,11,13,15)$



$F = BD + CD + AB' + B'D'$

⇒ Reduce the $f = \Sigma m(0,1,2,3,5,7,8,9,10,12,13)$.

AB \ CD	00	01	11	10
00	1_{0}	1_{1}	1_{3}	1_{2}
01	$_{4}$	1_{5}	1_{7}	$_{6}$
11	1_{12}	1_{13}	$_{15}$	$_{14}$
10	1_{8}	1_{9}	$_{11}$	1_{10}

$f = A'D + AC' + B'D'$

⇒ Plot the logical expression $ABCD + AB\bar{C}\bar{D} + A\bar{B}C + AB$ on a 4-variable k-map.

Sol: Given eq. $F = ABCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C + AB$

- First convert the given eq. into SSOP.

$F = ABCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C(D+\bar{D}) + [AB(C+\bar{C})](D+\bar{D})$

$= ABCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}C\bar{D} + (ABC + AB\bar{C})(D+\bar{D})$

$= ABCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}C\bar{D} + ABC(D+\bar{D}) + AB\bar{C}(D+\bar{D})$

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

$\underbrace{1111}_{m_{15}}$ $\underbrace{1000}_{m_8}$ $\underbrace{1011}_{m_{11}}$ $\underbrace{1010}_{m_{10}}$ $\underbrace{1111}_{m_{15}}$ $\underbrace{1110}_{m_{14}}$ $\underbrace{1101}_{m_{13}}$ $\underbrace{1100}_{m_{12}}$

$F = \Sigma m(8,10,11,12,13,14,15)$

AB \ CD	00	01	11	10
00	$_{0}$	$_{1}$	$_{3}$	$_{2}$
01	$_{4}$	$_{5}$	$_{7}$	$_{6}$
11	1_{12}	1_{13}	1_{15}	1_{14}
10	1_{8}	$_{9}$	1_{11}	1_{10}

$F = AB + AC + A\bar{D}$

⇒ Simplify the expression $Y = \Sigma m(7,9,10,11,12,13,14,15)$ using the k-map method

Sol:

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	1	6
11	1 12	1 13	1	1 14
10	8	1 9	1 11	1 10

$Y = AB + AD + AC + BCD$

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

⇨ Simplify the expression $Y = \Sigma m(3, 4, 5, 7, 9, 13, 14, 15)$

AB \ CD	00	01	11	10
00	0	1	1	2
01	1 4	1 5	1	6
11	12	1 13	1 15	1 14
10	8	1 9	11	10

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

Five-Variable K-Map

- Maps for more than four variables are not as simple to use as maps for four or fewer variables. A five-variable map needs 32 squares. When the no. of variables becomes large, the no. of squares becomes excessive and the geometry for combining adjacent squares becomes more involved.
- It consists of 2 four-variable maps with variables A, B, C, D, E. A distinguishes b/n the two maps, as indicated at the top of the diagram.

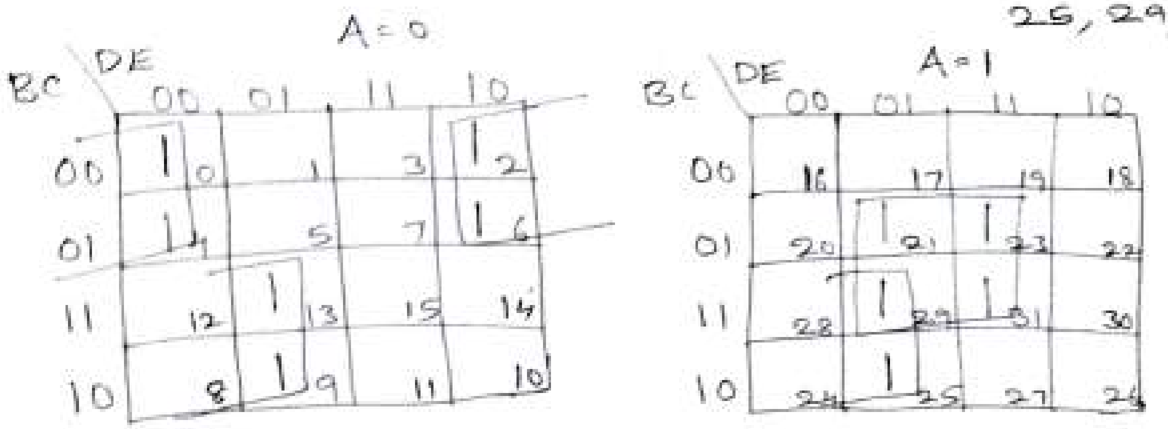
A = 0

BC \ DE	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

A = 1

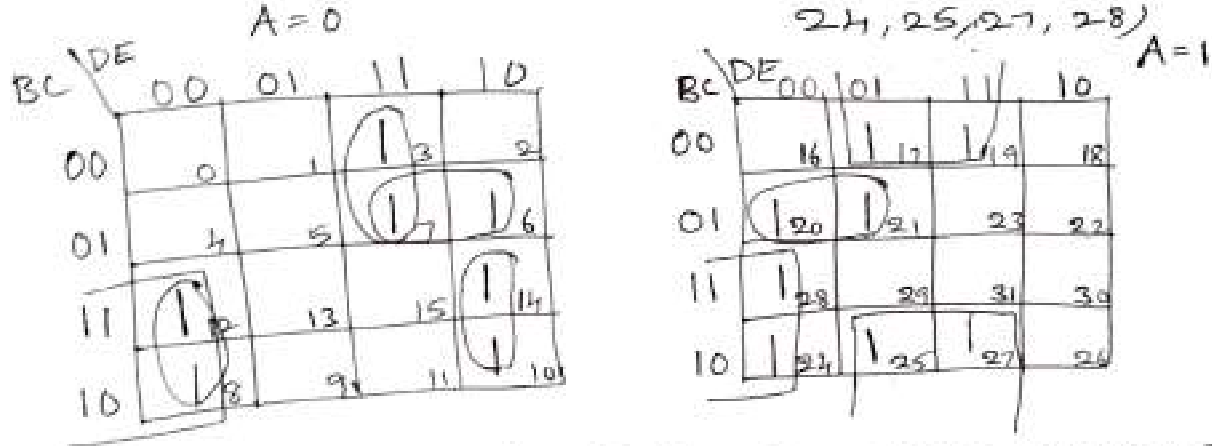
BC \ DE	00	01	11	10
00	m_{16}	m_{17}	m_{19}	m_{18}
01	m_{20}	m_{21}	m_{23}	m_{22}
11	m_{28}	m_{29}	m_{31}	m_{30}
10	m_{24}	m_{25}	m_{27}	m_{26}

⇒ Simplify $F(A,B,C,D,E) = (0,2,4,6,9,13,21,23,25,29,31)$



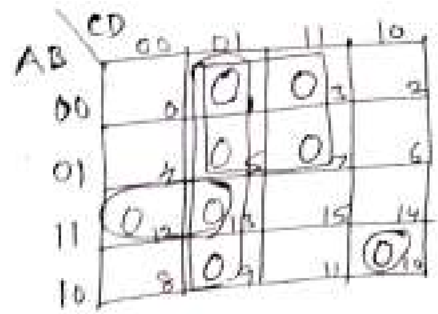
$$F = A'B'E' + ACE + BD'E$$

⇒ Simplify $Y = \Sigma m(3,6,7,8,10,12,14,17,19,20,21,24,25,27,28)$



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

⇒ Minimize the given fn $f(A,B,C,D) = \Pi M(1,3,5,7,9,10,12,13)$



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

<u>Don't care Combinations</u>: In certain digital systems, some i/p combinations never occur during the process of a normal operation because those i/p conditions are guaranteed never to occur. Such i/p combinations are don't care combinations. We don't care what the fn o/p is for such combinations. These combinations can be plotted

on a map to provide further simplification of the fn.

- The function considered so far in the various examples, for simplification using the k-map method, are completely specified, i.e. it assumes the value 1 for some i/p combinations & the value 0 for others. Also, there are functions which assume the value 1 for some combinations, the value 0 for some others and either 0 or 1 for the remaining combinations. Such functions are called incompletely specified functions, and the combinations for which the value of the function is not specified are called don't care combinations. The don't care combinations are represented by d or x.
- When an incompletely specified fn, i.e. a fn with don't care combinations, is simplified to obtain minimal SOP expression, the value 1 can be assigned to select don't care combinations. This is done in order to increase the no. of 1s in the selected groups, wherever further simplification is possible. Also, a don't care combination need not be used in grouping if it does not cover a large no. of 1s. In each case, the choice depends only on the simplification that has to be achieved. Similarly, when a function is simplified to obtain a minimal POS expression, the value 0 can be assigned to selected don't care combinations in order to increase the number of 0s in the selected groups which results in further simplification.

Prob: Simplify $F(A,B,C,D) = \Sigma m(1,3,7,11,15) + \Sigma d(0,2,5)$

AB \ CD	00	01	11	10
00	X_0	1_1	1_3	X_2
01	4	X_5	1_7	6
11	12	13	1_{15}	14
10	8	9	1_{11}	10

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

⇒ Simplify $Y = \Sigma m(0,2,3,6,7) + \Sigma d(8,10,11,15)$

AB \ CD	00	01	11	10
00	1_0	1	1_3	1_2
01	4	5	1_7	1_6
11	12	13	X_{15}	14
10	X_8	9	X_{11}	X_{10}

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

⇒ Simplify $Y = \Pi M(1,4,5,9,12,13,14) + \Sigma d(8,10,11,15)$

AB \ CD	00	01	11	10
00	0	0_1	3	2
01	0_4	0_5	7	6
11	0_{12}	0_{13}	X_{15}	0_{14}
10	X_8	0_9	X_{11}	X_{10}

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

⇒ Simplify $Y = \Sigma m(1,5,7,13,14,15,17,18,21,22,25,29) + \Sigma d(6,9,19,23,30)$

A = 0

BC \ DE	00	01	11	10
00	0	1_1	3	2
01	4	1_5	1_7	X_6
11	12	1_{13}	1_{15}	1_{14}
10	8	X_9	11	10

A = 1

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

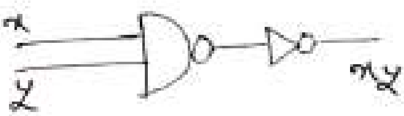
$Y = \bar{D}E + \bar{A}CD + A\bar{B}\bar{D}$

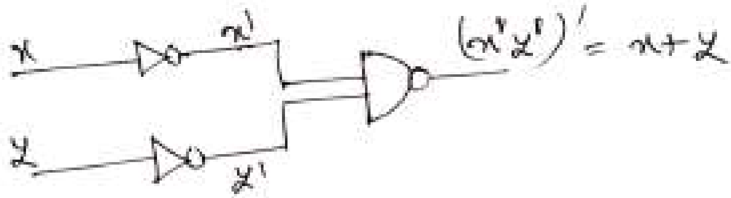
NAND and NOR Implementation

– Digital circuits are frequently constructed with NAND or NOR gates rather than with AND and OR gates. NAND & NOR gates are easier to fabricate with electronic components & are the basic gates used in all IC digital logic families. Because of the prominence of NAND and NOR gates in the design of digital circuits, rules and procedures have been developed for the conversion from boolean functions given in terms of AND, OR and NOT into eqt NAND and NOR logic diagrams.

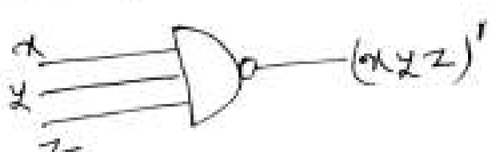
NAND circuits

Inverter 

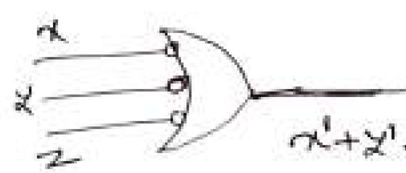
AND 

OR 

Three-input NAND Gate:



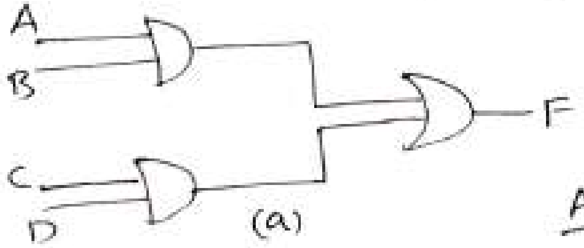
AND-invert



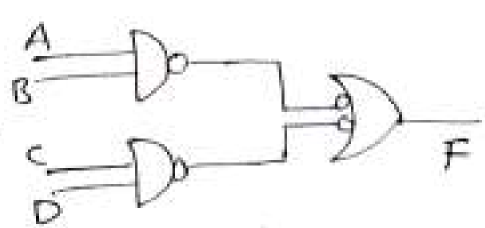
$x'+y'+z' = (xyz)'$

Invert-OR

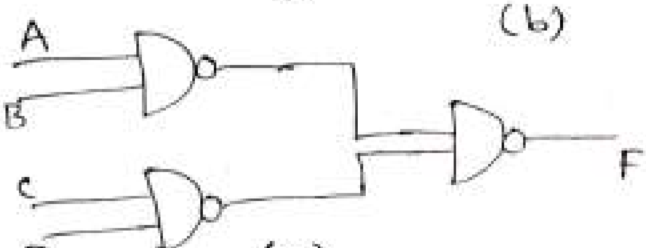
⇒ Implement F = AB + CD



(a)



(b)



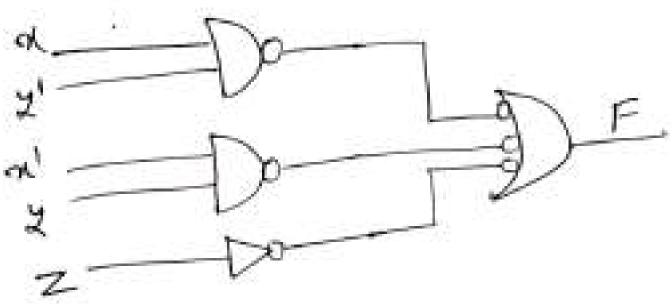
(c)

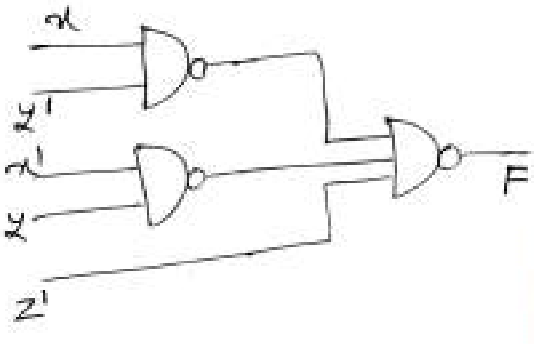
Fig. Three ways to implement F = AB + CD

Prob: Implement $F(x,y,z) = \Sigma m(1,2,3,4,5,7)$

$x \backslash yz$	00	01	11	10
0	$_0$	1_1	1_3	1_2
1	1_4	1_5	1_7	$_6$

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

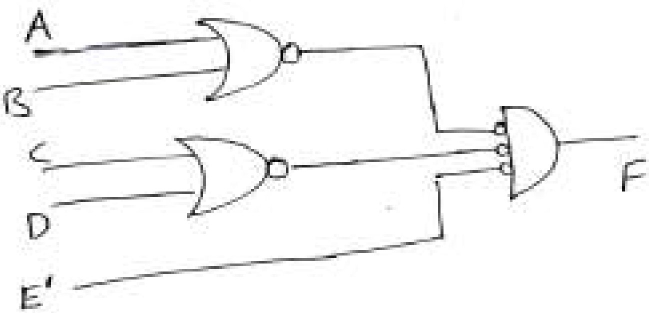


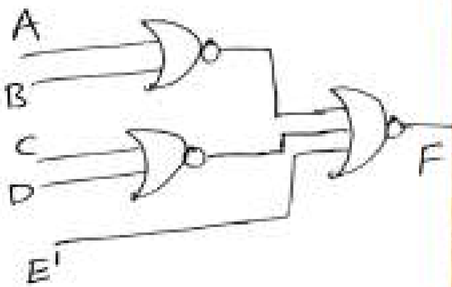


NOR Implementation

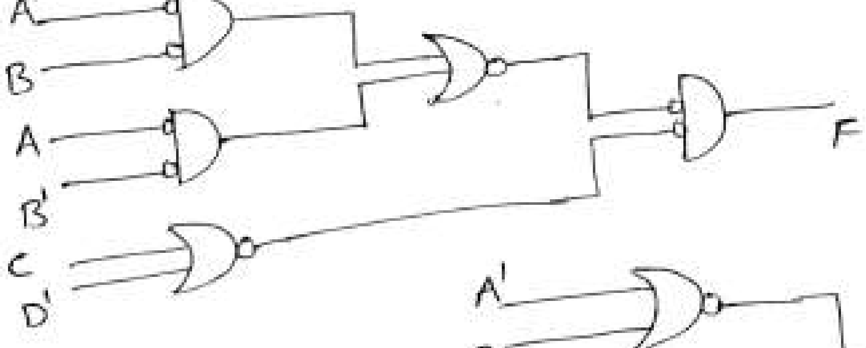
Implement $F = (A+B)(C+D)E$

Sol:

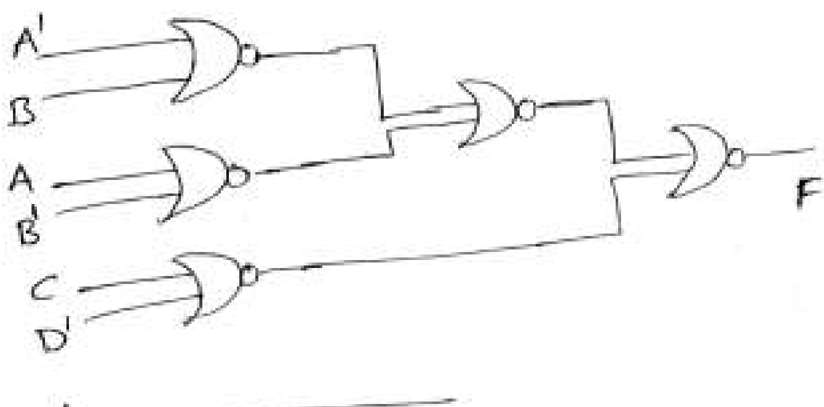




⇒ Implement $F = (AB + A'B)(C + D')$ with NOR gates



$\cong$



*