ARULMIGU PALANIANDAVR ARTS COLLEGE FOR WOMEN
(Autonomous)
(Re-Accredited with 'B++' Grade by NAAC 3rd Cycle)
Run by Arulmigu Dhandayuthapani Swamy Thirukoil, H.R & C.E Dept. Government of Tamil Nadu
A Government Aided College - Affiliated to Mother Teresa Women's University, Kodaikanal
CHINNAKALAYAMPUTHUR(PO), PALANI - 624615

DEPARTMENT OF COMPUTER SCIENCE

AND

COMPUTER APPLICATION

LEARNING RESOURCE

NUMBER SYSTEMS:

- The term digital refers to a process that is achieved by using discrete unit.
- In number system there are different symbols and each symbol has an absolute value and also hasplace value.

RADIX OR BASE:-

The radix or base of a number system is defined as the number of different digits which can occur ineach position in the number system.

RADIX POINT :-

The generalized form of a decimal point is known as radix point. In any positional number system theradix point divides the integer and fractional part.

N_r = [Integer part **.** Fractional part]

↑
Radix point

NUMBER SYSTEM:-

In general a number in a system having base or radix ‘ r ’ can be written as

$$a_n \; a_{n-1} \; a_{n-2} \; \ldots\ldots\ldots \; a_0 \; . \; a_{-1} \; a_{-2} \ldots\ldots\ldots \; a_{-m}$$

This will be interpreted as

$$Y = a_n \times r^n + a_{n-1} \times r^{n-1} + a_{n-2} \times r^{n-2} + \ldots\ldots + a_0 \times r^0 + a_{-1} \times r^{-1} + a_{-2} \times r^{-2} + \ldots \quad + a_{-m} \times r^{-m}$$

where Y = value of the entire number

a_n = the value of the

nth digitr = radix

TYPES OF NUMBER SYSTEM:-

There are four types of number systems. They are

1. Decimal number system
2. Binary number system
3. Octal number system
4. Hexadecimal number system

DECIMAL NUMBER SYSTEM:-

- The decimal number system contain ten unique symbols 0,1,2,3,4,5,6,7,8 and 9.
- In decimal system 10 symbols are involved, so the base or radix is 10.
- It is a positional weighted system.
- The value attached to the symbol depends on its location with respect to the decimal point.

In general,

$(d_n \times 10^n) + (d_{n-1} \times 10^{n-1}) + (d_{n-2} \times 10^{n-2}) + \dots + (d_0 \times 10^0) + (d_{-1} \times 10^{-1}) + (d_{-2} \times 10^{-2}) + \dots + (d_{-m} \times 10^{-m})$

For example:-

$9256.26 = 9 \times 1000 + 2 \times 100 + 5 \times 10 + 6 \times 1 + 2 \times (1/10) + 6 \times (1/100)$

$= 9 \times 10^3 + 2 \times 10^2 + 5 \times 10^1 + 6 \times 10^0 + 2 \times 10^{-1} + 6 \times 10^{-2}$

BINARY NUMBER SYSTEM:-

- The binary number system is a positional weighted system.
- The base or radix of this number system is 2.
- It has two independent symbols.
- The symbols used are 0 and 1.
- A binary digit is called a bit.
- The binary point separates the integer and fraction parts.

In general,

$d_n \quad d_{n-1} \quad d_{n-2} \quad \dots\dots\dots\dots \quad d_0 \;.\; d_{-1} \quad d_{-2} \qquad d_{-k}$

is given

$(d_n \times 2^n) + (d_{n-1} \times 2^{n-1}) + (d_{n-2} \times 2^{n-2}) + \ldots + (d_0 \times 2^0) + (d_{-1} \times 2^{-1}) + (d_{-2} \times 2^{-2}) + \ldots + (d_{-k} \times 2^{-k})$

OCTAL NUMBER SYSTEM:-

- It is also a positional weighted system.
- Its base or radix is 8.
- It has 8 independent symbols 0,1,2,3,4,5,6 and 7.
- Its base $8 = 2^3$, every 3- bit group of binary can be represented by an octal digit.

HEXADECIMAL NUMBER SYSTEM:-

- The hexadecimal number system is a positional weighted system.
- The base or radix of this number system is 16.
- The symbols used are 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E and F
- The base 16 = 24 , every 4 - bit group of binary can be represented by an hexadecimal digit.

CONVERSION FROM ONE NUMBER SYSTEM TO ANOTHER :-

1. **BINARY NUMBER SYSTEM:-**

(a) Binary to decimal conversion:-

In this method, each binary digit of the number is multiplied by its positional

weight and the product termsare added to obtain decimal number.

$$d_n \; d_{n-1} \; d_{n-2} \; \ldots\ldots \; d_0 \; . \; d_{-1} \; d_{-2} \ldots\ldots d_{-m}$$

For example:

(i) Convert $(10101)_2$ to

decimal. Solution :

(Positional weight) $2^4 \; 2^3 \; 2^2 \; 2^1 \; 2^0$

Binary number 10101

$= (1 \times 2^4) + (0 \times 2^3) + (1\times 2^2) + (0 \times 2^1) + (1 \times 2^0)$

$= 16 + 0 + 4 + 0 + 1$

$= (21)_{10}$

(ii) Convert $(111.101)_2$ to

decimal. Solution:

$(111.101)_2 = (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3})$

$= 4 + 2 + 1 + 0.5 + 0 + 0.125$

$= (7.625)_{10}$

Binary to Octal conversion:-

For conversion binary to octal the binary numbers are divided into groups of 3 bits each, starting at the binary point and proceeding towards left and right.

Octal	Binary	Octal	Binary
0	000	4	100
1	001	5	101
2	010	6	110
3	011	7	111

For example:

(i) Convert $(101111010110.110110011)_2$ into octal.

Solution :

Group of 3 bits are 101 111 010 110 . 110 110 011

Convert each group into octal = 5 7 2 6 . 6 6 3

The result is **(5726.663)$_8$**

(iii) Convert (10101111001.0111)$_2$

into octal.Solution :

Binary number	10	101	111	001	.	011	1
Group of 3 bits are =	010	101	111	001	.	011	100
Convert each group into octal =	2	5	7	1	.	3	4

The result is **(2571.34)$_8$**

(b) Binary to Hexadecimal conversion:-

For conversion binary to hexadecimal number the binary numbers starting from the binary point, groups are made of 4 bits each, on either side of the binary point.

Hexadecimal	Binary	Hexadecimal	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110

7	0111	F	1111

7	0111	F	1111

or example:
(i) Convert $(1011011011)_2$ into hexadecimal. Solution:

Given Binary number		10	1101	10
Group of 4 bits are		0010	1101	10
Convert each group into hex	=	2	D	E

- **Convert $(01011111011.011111)_2$ into hexadecimal. Solution:**

Given Binary number		010	1111	1011	.	0111	11
Group of 3 bits are	=	0010	1111	1011	.	0111	1100
Convert each group into octal	=	2	F	B	.	7	C

The result is **$(2FB.7C)_{16}$**

2. DECIMAL NUMBER SYSTEM:-

(a) Decimal to binary conversion:-

In the conversion the integer number are converted to the desired base using successive division by the base or radix.

For example:
(i) Convert $(52)_{10}$ into binary.

Solution:

Divide the given decimal number successively by 2 read the integer part remainder upwards to get equivalent binary number. Multiply the fraction part by 2. Keep the integer in the product as it is and multiply the new fraction in the product by 2. The process is continued and the integer are read in the products from top to bottom.

2 | 52
2 | 26 — 0
2 | 13 — 0
2 | 6 — 1
2 | 3 — 0
2 | 1 — 1
0 — 1

Result of $(52)_{10}$ is **$(110100)_2$**

:

Integer part		**Fraction part**
2 | 105		0.15 x 2 = 0.30
2 | 52	— 1	0.30 x 2 = 0.60
2 | 26	— 0	0.60 x 2 = 1.20
2 | 13	— 0	0.20 x 2 = 0.40
2 | 6	— 1	0.40 x 2 = 0.80
2 | 3	— 0	0.80 x 2 = 1.60
2 | 1	— 1	
0	— 1	

Result of $(105.15)_{10}$ is **$(1101001.001001)_2$**

(b) <u>Decimal to octal conversion</u>:-

To convert the given decimal integer number to octal, successively divide the given number by 8 till the quotient is 0. To convert the given decimal fractions to octal successively multiply the decimal fraction and the subsequent decimal fractions by 8 till the product is 0 or till the required accuracy is obtained.

For example:
(i) Convert $(378.93)_{10}$ into octal.Solution:

8 | 378		0.93 x 8 = 7.44
8 | 47	— 2	0.44 x 8 = 3.52
8 | 5	— 7	0.52 x 8 = 4.16
0	— 5	0.16 x 8 = 1.28

Result of $(378.93)_{10}$ is **$(572.7341)_8$**

(c) Decimal to hexadecimal conversion:-

The decimal to hexadecimal conversion is same as octal.

For example:

(i) Convert $(2598.675)_{10}$ into hexadecimal.

Solution:

		Remainder			
		Decimal	**Hex**		**Hex**
16	I 2598			0.675 x 16 = 10.8	A
16	I 162	— 6	6	0.800 x 16 = 12.8	C
16	I 10	— 2	2	0.800 x 16 = 12.8	C
	0	— 10	A	0.800 x 16 = 12.8	C

Result of $(2598.675)_{10}$ is **$(A26.ACCC)_{16}$**

3. OCTAL NUMBER SYSTEM:-

(a) Octal to binary conversion:-

To convert a given a octal number to binary, replace each octal digit by its 3- bit binary equivalent.

For example:

Convert $(367.52)_8$ into binary.

Solution:

Given Octal number is 3 6 7 . 5 2

Convert each group octal to binary	= 011 110 111 . 101 010

Result of $(367.52)_8$ is $\mathbf{(011110111.101010)_2}$

(b) Octal to decimal conversion:-

For conversion octal to decimal number, multiply each digit in the octal number by the weight of its position and add all the product terms

For example: -

Convert $(4057.06)_8$ to decimal Solution:

$(4057.06)_8 = 4 \times 8^3 + 0 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 + 0 \times 8^{-1} + 6 \times 8^{-2}$
$= 2048 + 0 + 40 + 7 + 0 + 0.0937$
$= (2095.\ 0937)_{10}$

Result is **$(2095.0937)_{10}$**

(c) Octal to hexadecimal conversion:-

For conversion of octal to Hexadecimal, first convert the given octal number to binary and then binary number to hexadecimal.

For example :-

Convert $(756.603)_8$ to hexadecimal.

Solution :-

Given octal no.	=	7	5	6	.	6	0	3
Convert each octal digit to binary	=	111	101	110	.	110	000	011
Group of 4bits are	=	0001	1110	1110	.	1100	0001	1000
Convert 4 bits group to hex.	=	1	E	E	.	C	1	8

Result is **$(1EE.C18)_{16}$**

(4) HEXADECIMAL NUMBER SYSTEM :-

(a) Hexadecimal to binary conversion:-

For conversion of hexadecimal to binary, replace hexadecimal digit by its 4 bit binary group. **For example:**

Convert $(3A9E.B0D)_{16}$ into binary.

Solution:

Given Hexadecimal number is3 A 9 E . B 0 D

Convert each hexadecimal = 0011 1010 1001 1110 . 1011 0000 1101digit to 4 bit binary

Result of $(3A9E.B0D)_8$ is $\mathbf{(0011101010011110.101100001101)_2}$

(b) <u>Hexadecimal to decimal conversion</u>:-

For conversion of hexadecimal to decimal, multiply each digit in the hexadecimal number by its position weight and add all those product terms.

For example: -

Convert $(A0F9.0EB)_{16}$ to decimal

Solution:

$(A0F9.0EB)_{16} = (10 \times 16^3) + (0 \times 16^2) + (15 \times 16^1) + (9 \times 16^0) + (0 \times 16^{-1}) + (14 \times 16^{-2}) + (11 \times 16^{-3})$

$= 40960 + 0 + 240 + 9 + 0 + 0.0546 + 0.0026$

$= (41209.0572)_{10}$

Result is $\mathbf{(41209.0572)_{10}}$

(c) Hexadecimal to Octal conversion:-

For conversion of hexadecimal to octal, first convert the given hexadecimal number to binary and then binary number to octal.

For example :-
Convert $(B9F.AE)_{16}$ to octal.

Solution :-

Given hexadecimal no.is		B		9	F	.	A		E
Convert each hex. digit to binary	=	1011		1001	1111	.	1010		1110
Group of 3 bits are	=	101	110	011	111	.	101	011	100
Convert 3 bits group to octal.	=	5	6	3	7	.	5	3	4

Result is **$(5637.534)_8$**

BINARY ARITHEMATIC OPERATION :-

1. **BINARY ADDITION:-**

The binary addition rules are as follows

0 + 0 = 0 ; 0 + 1 = 1 ; 1 + 0 = 1 ; 1 + 1 = 10 , i.e 0

with a carry of 1**For example :-**

Add $(100101)_2$ and $(1101111)_2$.

Solution :-

1 0 0 1 0 1

```
+   1 1 0 1 1 1 1
1 0 0 1 0 1 0 0
```

Result is **(10010100)$_2$**

2. **BINARY SUBTRACTION:-**

The binary subtraction rules are as follows
0 - 0 = 0 ; 1 - 1 = 0 ; 1 - 0 = 1 ; 0 - 1 = 1 , with a borrow of 1

For example :-
Substract (111.111)$_2$ from (1010.01)$_2$.Solution :-

```
1 0 1 0 . 0 1 0
-     1 1 1 . 1 1 1
  0 0 1 0 .0 1 1
```

Result is **(0010.011)$_2$**

3. **BINARY MULTIPLICATION:-**

The binary multiplication rules are as follows 0 x 0 = 0 ; 1 x 1 = 1 ; 1 x 0 = 0 ; 0 x 1 = 0

For example :-

Multiply (1101)$_2$ by (110)$_2$.Solution :-

```
1 1 0 1
```

```
x          1 1 0
         ________
            0 0 0 0
          1 1 0 1
+       1 1 0 1
         __________
         1 0 0 1 1 1
         0
```

Result is **$(1001110)_2$**

4. **<u>BINARY DIVISION</u>:-**

The binary division is very simple and similar to decimal number system. The division by '0' is meaningless. So we have only 2 rules

0 ÷ 1 = 0

1 ÷ 1 = 1

For example :-

Divide $(10110)_2$ by $(110)_2$. Solution :-

```
110 ) 101101 ( 111.1
    -  110
       1010
        110
        1001
         110
         110
          110
          000
```

Result is **$(111.1)_2$**

1's COMPLEMENT REPRESENTATION :-

The 1's complement of a binary number is obtained by changing each 0 to 1 and each 1 to 0.

For example :-

Find $(1100)_2$ 1's complement.

Solution :-

```
            Given               1     1    0    0
       1's complement is 0     0    1    1
```

Result is **$(0011)_2$**

2's COMPLEMENT REPRESENTATION :-

The 2's complement of a binary number is a binary number which is obtained by adding 1 to the 1's complement of a number i.e.

2's complement = 1's complement + 1

For example :-

Find $(1010)_2$ 2's complement.

Solution :-

```
                    Given               1     0    1    0
               1's complement is 0     1    0    1
           +                    1
                 ____________________
2's complement      0   1   1   0
```

Result is **$(0110)_2$**

SIGNED NUMBER :-

In sign – magnitude form, additional bit called the sign bit is placed in front of the number. If the sign bit is 0, the number is positive. If it is a 1, the number is negative.

For example:-

```
0  1  0  1  0  0  1   =  +41
↑
```

Sign
bit

$1\ 1\ 0\ 1\ 0\ 0\ 1 = -41$
↑
Sign
bit

SUBSTRACTION USING COMPLEMENT METHOD :-

1's COMPLEMENT:-

In 1's complement subtraction, add the 1's complement of subtrahend to the minuend. If there is a carry out, then the carry is added to the LSB. This is called end around carry. If the MSB is 0, the result is positive. If the MSB is 1, the result is negative and is in its 1's complement form. Then take its 1's complement to get the magnitude in binary.

example:-

Subtract $(10000)_2$ from $(11010)_2$ using 1's complement.

Solution:-

```
 1 1 0 1 0          1 1 0 1 0                              =  26
-  1 0 0 0    =>   +  0 1 1 1 1     (1's complement) = -      16
0
              Carry →  1  0 1 0 0                         +   10
                   1
                   +        1
                      0 1 0 1 0       = +10
```

Result is **+10**

For

2's COMPLEMENT:-

In 2's complement subtraction, add the 2's complement of subtrahend to the minuend. If there is a carry out, ignore it. If the MSB is 0, the result is positive. If the MSB is 1, the result is negative and is in its 2's complement form. Then take its 2's complement to get the magnitude in binary.

For example:-

Subtract $(1010100)_2$ from $(1010100)_2$ using 2's complement.

Solution:-

```
   1 0 1 0 1 0 0               1 0 1 0 1 0 0             =      84
 - 1 0 1 0 1 0 0    =>      +  0 1 0 1 1 0 0  (2's=            -
                            complement)                      84
                         =  1 0 0 0 0 0 0 0 ( Ignore the       0
                            carry)
                                   0 (result = 0)
```

Hence MSB is 0. The answer is positive. So it is +0000000 = **0**

DIGITAL CODES:-

In practice the digital electronics requires to handle data which may be numeric, alphabets and special characters. This requires the conversion of the incoming data into binary format before it can be processed. There is various possible ways of doing this and this process is called encoding. To achieve the reverse of it, we use decoders.

WEIGHTED AND NON-WEIGHTED CODES:-

There are two types of binary codes

1) Weighted binary codes
2) Non- weighted binary codes

In weighted codes, for each position (or bit) ,there is specific weight attached.

For example, in binary number, each bit is assigned particular weight 2n where 'n' is the bit number for n = 0,1,2,3,4 the weights are 1,2,4,8,16 respectively.

Example :- BCD

Non-weighted codes are codes which are not assigned with any weight to each digit position, i.e., each digit position within the number is not assigned fixed value.

Example:- Excess – 3 (XS -3) code and Gray codes

BINARY CODED DECIMAL (BCD):-

BCD is a weighted code. In weighted codes, each successive digit from right to left represents weights equal to some specified value and to get the equivalent decimal number add the products of the weights by the corresponding binary digit. 8421 is the most common because 8421 BCD is the most natural amongst the other possible codes.

For example:-
$(567)_{10}$ is encoded in various 4 bit codes.

Solution:-

Decimal	→	5	6	7
8421 code	→	0101	0110	0111
6311 code	→	0111	1000	1001
5421 code	→	1000	0100	1010

BCD ADDITION:-

Addition of BCD (8421) is performed by adding two digits of binary, starting from least significant digit. In case if the result is an illegal code (greater than 9) or if there is a carry out of one then add 0110(6) and add the resulting carry to the next most significant

.

For example:-

Add 679.6 from 536.8 using BCD addition.

Solution:-

```
6 7 9 . 6        0110   0111   1001 .  0110       ( 679.6 in
BCD)
+   5 3 6 . 8  =>+ 0101   0011                0110   .   1000
____________(536.8 in BCD)
    1 2 1 6 . 4        1011   1010                    1111 . 1110 ( All are illegal codes)
                     + 0110 +0110  +0110  .+0110        ( Add 0110 to each)
                  0001 0010 0001 0110 .  0100
                    1       2       1      6    .     4      ( corrected sum = 1216.4)
```

Result is **1216.4**

<u>BCD SUBTRACTION</u>:-

The BCD subtraction is performed by subtracting the digits of each 4 – bit group of the subtrahend from corresponding 4 – bit group of the minuend in the binary starting from the LSD. If there is no borrow from the next higher group[then no correction is required. If there is a borrow from the next group, then 6_{10} (0110) is subtracted from the difference term of this group.

For example:-

Subtract 147.8 from 206.7 using 8421 BCD code.

Solution:-

```
2 0 6 . 7        0010   0000                  0110    .    0111
                 ( 206.7 in BCD)
-   1 4 7 . 8  =>-  0001   0100                0111   .   1000
____________(147.8 in BCD)
        5 8 . 9        0000   1011   1110 . 1111 ( Borrows are present)
_________- 0110  -0110  .- 0110
0101  1000 .  1001
                                 5      8    .     9      ( corrected difference = 58.9)
```

Result is $\mathbf{(58.9)_{10}}$

EXCESS THREE(XS-3) CODE:-

The Excess-3 code, also called XS-3, is a non- weighted BCD code. This derives it name from the fact that each binary code word is the corresponding 8421 code word plus 0011(3). It is a sequential code. It is a self complementing code.

XS-3 ADDITION:-

In XS-3 addition, add the XS-3 numbers by adding the 4 bit groups in each column starting from the LSD. If there is no carry out from the addition of any of the 4 bit groups, subtract 0011 from the sum term of those groups. If there is a carry out, add 0011 to the sum term of those groups

For example:-
Add 37 and 28 using XS-3 code.

Solution:-

```
3 7                     0110  1010      ( 37 in XS-3)
+  2 8          =>  +   0101   1011     ( 28 in XS-3)
                    _____________
6 5                     1011 11010      ( Carry is generated)
                    +   1                ( Propagate carry)
                        1100 0101       ( Add 0110 to correct 0101 and
                      - 0011 +0011       subtract 0011 to correct 1100)
                        1001 1000       ( Corrected sum in XS-3 = 65₁₀)
```

<u>XS-3 SUBTRACTION</u>:-

To subtract in XS-3 number by subtracting each 4-bit group of the subtrahend from the corresponding 4-bit group of the minuend starting from the LSD. If there is no borrow from the next 4-bit group. add 0011 to the difference term of such groups. If there is a borrow, subtract 0011 from the difference term.

For example :-

. **Subtract 175 from 267 using XS-3 code. Solution :-`**

```
  267               0101   1010       1010      ( 267 in XS-3)
 -175      =>   -  0100   1010    1000      ( 175 in XS-3)
   092              0000   1111   0010   (Correct 0010 and 0000 by adding 0011 and
                   +0011  -0011 +0011   correct 1111 by subtracting 0011)
                    0011  1100   0101    (Corrected difference in XS-3 = 92₁₀ )
```

<u>ASCII CODE:</u>-

The American Standard Code for Information Interchange (ASCII) pronounced as 'ASKEE' is widely used alphanumeric code. This is basically a 7 bit code. The number of different bit patterns that can be created with 7 bits is 27 = 128 , the ASCII can be used to encode both the uppercase and lowercase characters of the alphabet (52 symbols) and some special symbols in addition to the 10 decimal digits. It is used extensively for printers and terminals that interface with small computer systems. The table shown below shows the ASCII groups.

The ASCII code

LSBs	**MSBs**							
	000	001	010	011	100	101	110	111
0000	NUL	DEL	Space	0	@	P	P	
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s

0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DLE

EBCDIC CODE:-

The Extended Binary Coded Decimal Interchange Code (EBCDIC) pronounced as ‘eb – si- dik’ is an 8 bit alphanumeric code. Since 28 = 256 bit patterns can be formed with 8 bits. It is used by most large computers to communicate in alphanumeric data. The table shown below shows the EBCDIC code.

The EBCDIC code

LSD (Hex)	**MSD(Hex)**															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	DLE	DS		SP	&							[	]	\	0
1	SOH	DC1	SOS				/		a	j	~		A	J		1
2	STX	DC2	FS	SYN					b	k	s		B	K	S	2
3	ETX	DC3							c	l	t		C	L	T	3
4	PF	RES	BYP	PN					d	m	u		D	M	U	4
5	HT	NL	LF	RS					e	n	v		E	N	V	5
6	LC	BS	EOB	YC					f	o	w		F	O	W	6
7	DEL	IL	PRE	EOT					g	p	x		G	P	X	7

8		CAN							h	q	y		H	Q	Y	8
9		EM							i	r	z		I	R	Z	9
A	SMM	CC	SM		Ø	!	I	:								
B	VT				.	$	,	#								
C	FF	IFS		DC4	<	*	%	@								
D	CR	IGS	ENQ	NAK	(	)	_	‘								
E	SO	IRS	ACK		+	;	>	=								
F	SI	IUS	BEL	SUB	I	‘	?	‘								

GRAY CODE

The gray code is a non-weighted code. It is not a BCD code. It is cyclic code because successive words in this differ in one bit position only i.e it is a unit distance code.

Gray code is used in instrumentation and data acquisition systems where linear or angular displacement is measured. They are also used in shaft encoders, I/O devices, A/D converters and other peripheral equipment.

-

BINARY- TO - GRAY CONVERSION:-

If an n-bit binary number is represented by B_n B_{n-1} - - - - - B1 and its gray code equivalent by G_n G_{n-1} -- G1, where B_n and G_n are the MSBs , then gray code bits are obtained from the binary code as follows $G_n = B_n$

$G_{n-1} = B_n \oplus B_{n-1}$

.

.

.

.

$G_1 = B_2 \oplus B_1$

Where the symbol $\oplus$ stands for Exclusive OR (X-OR)

For example :-

Convert the binary 1001 to the Gray code.

Solution :-`

$$\begin{tikzcd}
\text{Binary} \arrow[r] & 1 \arrow[r, dash] \arrow[d] & \oplus \arrow[r] & 0 \arrow[r, dash] \arrow[d] & \oplus \arrow[r] & 0 \arrow[r, dash] \arrow[d] & \oplus \arrow[r] & 1 \arrow[d] \\
\text{Gray} \arrow[r] & 1 & & 1 & & 0 & & 1
\end{tikzcd}$$

The gray code is **1101**

GRAY- TO - BINARY CONVERSION:-

If an n-bit gray number is represented by G_n G_{n-1} ------- G_1 and its binary equivalent by B_n B_{n-1} -- B1, then binary bits are obtained from Gray bits as follows : $B_n = G_n$

$B_{n-1} = B_n \oplus G_{n-1}$

.

.

.

.

$B_1 = B_2 \oplus G_1$

For example :-

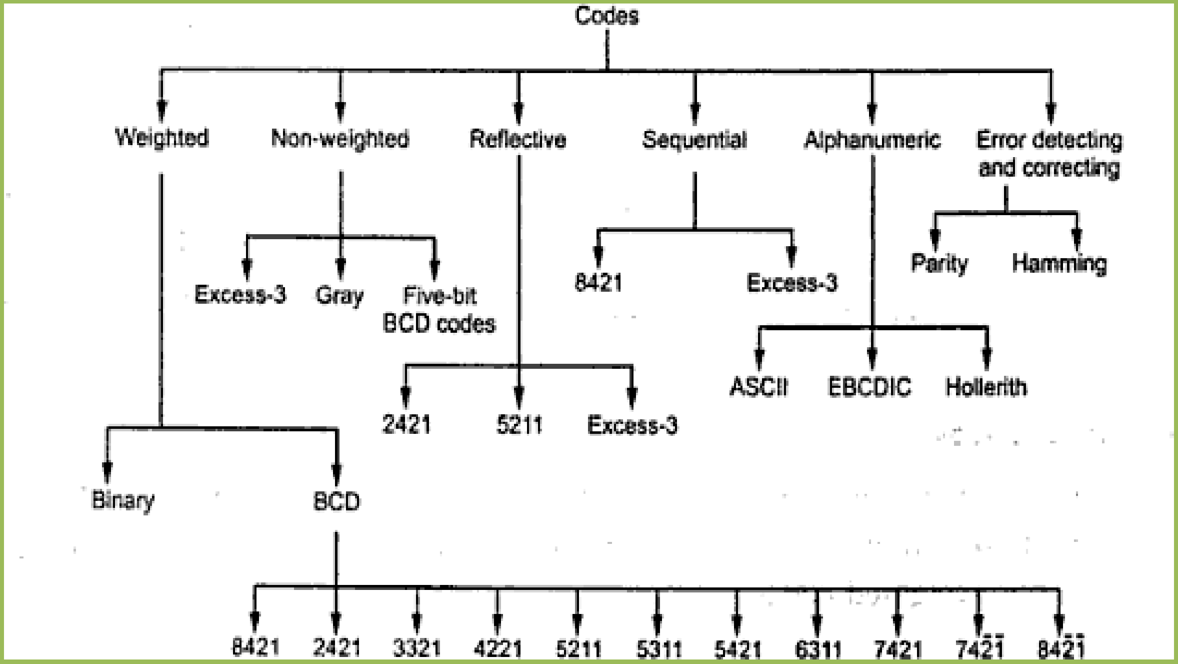
Convert the Gray code 1101 to the binary

Solution :-

$$\begin{tikzcd}
\text{Gray} \rightarrow & 1 \arrow[dd] & & 1 \arrow[dd] & & 0 \arrow[dd] & & 1 \arrow[dd] \\
& & \oplus \arrow[ur] & & \oplus \arrow[ur] & & \oplus \arrow[ur] & \\
\text{Binary} \rightarrow & 1 \arrow[ur, dash] & & 0 \arrow[ur, dash] & & 0 \arrow[ur, dash] & & 1
\end{tikzcd}$$

The binary code is **1001**

Codes:



Binary codes block diagram

.

Error - Detecting codes: When binary data is transmitted & processed,it is susceptible to noise that can alter or distort its contents. The 1‗s may get changed to 0‗s & 1‗s .because digital systems must be accurate to the digit, error can pose a problem. Several schemes have been devised to detect the occurrence of a single bit error in a binary word, so that whenever such an error occurs the concerned binary word can be corrected & retransmitted.

Parity: The simplest techniques for detecting errors is that of adding an extra bit known as parity bit to each word being transmitted.Two types of parity: Oddparity, evenparity forodd parity, the parity bit is set to a ‗0‗ or a ‗1‗ at the transmitter such that the total no. of 1 bit in the word including the parity bit is an odd no.For even parity, the parity bit is set to a ‗0‗ or a ‗1‗ at the transmitter such that the parity bit is an even no.

Decimal	8421 code	Odd parity	Even parity
0	0000	1	0
1	0001	0	1
2	0010	0	1
3	0011	1	0
4	0100	0	1
5	0100	1	0
6	0110	1	0
7	0111	0	1
8	1000	0	1
9	1001	1	0

When the digit data is received . a parity checking circuit generates an error signal if the total no of 1‗s is even in an odd parity system or odd in an even parity system. This parity check can always detect a single bit error but cannot detect 2 or more errors with in the same word.Odd parity is used more often than even parity does not detect the situation. Where all 0‗s are created by a short ckt or some other fault condition.

Ex: Even parity scheme

(a) 10101010 (b) 11110110 (c)10111001

Ans:

(a) No. of 1‘s in the word is even is 4 so there is no error
(b) No. of 1‘s in the word is even is 6 so there is no error
(c) No. of 1‘s in the word is odd is 5 so there is error

Ex: odd parity

(a)10110111(b) 10011010 (c)11101010

Ans:

(a) No. of 1‘s in the word is even is 6 so word has error
(b) No. of 1‘s in the word is even is 4 so word has error
(c) No. of 1‘s in the word is odd is 5 so there is no error

Checksums:

Simple parity can‘t detect two errors within the same word. To overcome this, use a sort of 2 dimensional parity. As each word is transmitted, it is added to the sum of the previously transmitted words, and the sum retained at the transmitter end. At the end of transmission, the sum called the check sum. Up to that time sent to the receiver. The receiver can check its sum with the transmitted sum. If the two sums are the same, then no errors were detected at the receiver end. If there is an error, the receiving location can ask for retransmission of the entire data, used in teleprocessing systems.

Block parity:

Block of data shown is create the row & column parity bits for the data using odd parity. The parity bit 0 or 1 is added column wise & row wise such that the total no. of 1‘s in each column & row including the data bits & parity bit is odd as

When the digit data is received . a parity checking circuit generates an error signal if the total no of 1‘s is even in an odd parity system or odd in an even parity system. This parity check can always detect a single bit error but cannot detect 2 or more errors with in the same word.Odd parity is used more often than even parity does not detect the situation. Where all 0‘s are created by a short ckt or some other fault condition.

Ex: Even parity scheme

(a) 10101010 (b) 11110110 (c)10111001

Ans:

(d) No. of 1‘s in the word is even is 4 so there is no error
(e) No. of 1‘s in the word is even is 6 so there is no error
(f) No. of 1‘s in the word is odd is 5 so there is error

Ex: odd parity

(a)10110111(b) 10011010 (c)11101010

Ans:

(d) No. of 1‘s in the word is even is 6 so word has error
(e) No. of 1‘s in the word is even is 4 so word has error
(f) No. of 1‘s in the word is odd is 5 so there is no error

Checksums:

Simple parity can‘t detect two errors within the same word. To overcome this, use a sort of 2 dimensional parity. As each word is transmitted, it is added to the sum of the previously transmitted words, and the sum retained at the transmitter end. At the end of transmission, the sum called the check sum. Up to that time sent to the receiver. The receiver can check its sum with the transmitted sum. If the two sums are the same, then no errors were detected at the receiver end. If there is an error, the receiving location can ask for retransmission of the entire data, used in teleprocessing systems.

Block parity:

Block of data shown is create the row & column parity bits for the data using odd parity. The parity bit 0 or 1 is added column wise & row wise such that the total no. of 1‘s in each column & row including the data bits & parity bit is odd as

When the digit data is received . a parity checking circuit generates an error signal if the total no of 1‘s is even in an odd parity system or odd in an even parity system. This parity check can always detect a single bit error but cannot detect 2 or more errors with in the same word.Odd parity is used more often than even parity does not detect the situation. Where all 0‘s are created by a short ckt or some other fault condition.

Ex: Even parity scheme

(a) 10101010 (b) 11110110 (c)10111001

Ans:

(g) No. of 1‘s in the word is even is 4 so there is no error

(h) No. of 1‘s in the word is even is 6 so there is no error

(i) No. of 1‘s in the word is odd is 5 so there is error

Ex: odd parity

(a)10110111(b) 10011010 (c)11101010

Ans:

(g) No. of 1‘s in the word is even is 6 so word has error

(h) No. of 1‘s in the word is even is 4 so word has error

(i) No. of 1‘s in the word is odd is 5 so there is no error

Checksums:

Simple parity can‘t detect two errors within the same word. To overcome this, use a sort of 2 dimensional parity. As each word is transmitted, it is added to the sum of the previously transmitted words, and the sum retained at the transmitter end. At the end of transmission, the sum called the check sum. Up to that time sent to the receiver. The receiver can check its sum with the transmitted sum. If the two sums are the same, then no errors were detected at the receiver end. If there is an error, the receiving location can ask for retransmission of the entire data, used in teleprocessing systems.

Block parity:

Block of data shown is create the row & column parity bits for the data using odd parity. The parity bit 0 or 1 is added column wise & row wise such that the total no. of 1‘s in each column & row including the data bits & parity bit is odd as

Data	Parity bit	data
10110	0	10110
10001	1	10001
10101	0	10101
00010	0	00010
11000	1	11000
00000	1	00000
11010	0	11010

Error –Correcting Codes:

A code is said to be an error –correcting code, if the code word can always be deduced from an erroneous word. For a code to be a single bit error correcting code, the minimum distance of that code must be three. The minimum distance of that code is the smallest no. of bits by which any two code words must differ. A code with minimum distance of 3 can't only correct single bit errors but also detect (can't correct) two bit errors, The key to error correction is that it must be possible to detect & locate erroneous that it must be possible to detect & locate erroneous digits. If the location of an error has been determined. Then by complementing the erroneous digit, the message can be corrected , error correcting , code is

the Hamming code , In this , to each group of m information or message or data bits, K parity checking bits denoted by P1,P2, pk located at positions 2^{k-1} from left are added to form an (m+k) bit code word.

To correct the error, k parity checks are performed on selected digits of each code word, & the position of the error bit is located by forming an error word, & the error bit is then complemented. The k bit error word is generated by putting a 0 or a 1 in the 2^{k-1}th position depending upon whether the check for parity involving the parity bit Pk is satisfied or not.Error positions & their corresponding values :

Error Position	For 15 bit code C4 C3 C2 C1	For 12 bit code C4 C3 C2 C1	For 7 bit code C3 C2 C1
0	0 0 0 0	0 0 0 0	0 0 0
1	0 0 0 1	0 0 0 1	0 0 1
2	0 0 1 0	0 0 1 0	0 1 0
3	0 0 1 1	0 0 1 1	0 1 1
4	0 1 0 0	0 1 0 0	1 0 0
5	0 1 0 1	0 1 0 1	1 0 1
6	0 1 1 0	0 1 1 0	1 1 0
7	0 1 1 1	0 1 1 1	1 1 1
8	1 0 0 0	1 0 0 0	
9	1 0 0 1	1 0 0 1	
10	1 0 1 0	1 0 1 0	
11	1 0 1 1	1 0 1 1	
12	1 1 0 0	1 1 0 0	
13	1 1 0 1		
14	1 1 1 0		
15	1 1 1 1		

Alphanumeric Codes:

These codes are used to encode the characteristics of alphabet in addition to the decimal digits. It is used for transmitting data between computers & its I/O device such as printers, keyboards & video display terminals.Popular modern alphanumeric codes are ASCII code & EBCDIC code.

BOOLEAN ALGEBRA

INTRODUCTION:-

- Switching circuits are also called logic circuits, gates circuits and digital circuits.
- Switching algebra is also called Boolean algebra.
- Boolean algebra is a system of mathematical logic. It is an algebraic system consisting of the set of elements (0,1), two binary operators called OR and AND and unary operator called NOT.
- It is the basic mathematical tool in the analysis and synthesis of switching circuits.
- It is a way to express logic functions algebraically.
- Any complex logic can be expressed by a Boolean function.
- The Boolean algebra is governed by certain well developed rules and laws.

AXIOMS AND LAWS OF BOOLEAN ALGEBRA:-

Axioms or postulates of Boolean algebra are set of logical expressions that are accepted without proof and upon which we can build a set of useful theorems. Actually, axioms are nothing more than the definitions of the three basic logic operations AND, OR and INVERTER. Each axiom can be interpreted as the outcome of an operation performed by a logic gate.

AND operation	**OR operation**	**NOT operation**
Axiom 1: 0 . 0 = 0	Axiom 5: 0 + 0 = 0	Axiom 9: $\overline{1} = 0$

Axiom 2: 0 . 1 = 0
Axiom 3: 1 . 0 = 0
Axiom 2: 1 . 1 = 1

Axiom 6: 0 + 1 = 1 Axiom 10:0 = 1
Axiom 7: 1 + 0 = 1
Axiom 8: 1 + 1 = 1

1.Complementation Laws:-

The term complement simply means to invert, i.e. to changes 0s to 1s and 1s to 0s. The five laws of complementation are as follows:

Law 1: $\overline{0}$ = 1
Law 2: $\overline{1}$ = 0
Law 3: if A = 0, then
A = 1 **Law 4:** if $\underline{A}$ =
1,then$\underline{A}$ = 0

Law 5: A = 0 (double complementation law)

2. OR Laws:-

The four OR laws are as
follows **Law 1:** A + 0
= 0(Null law)
Law 2: A + 1 = 1(Identity
law)**Law 3:** A + A = A

Law 4: $\overline{A}$+A = 1

3. AND Laws:-

The four AND laws are as
follows **Law 1:** A . 0 =
0(Null law) **Law 2:** A . 1
= 1(Identity law)**Law 3:**
A . A = A

Law 4: A .A = 0

4. Commutative Laws:-

Commutative laws allow change in position of AND or OR variables. There are two commutative laws.

Law 1: A + B = B + A

Proof

A	**B**	**A + B**
0	0	0
0	1	1
1	0	1
1	1	1

B	**A**	**B+ A**
0	0	0
0	1	1
1	0	1
1	1	1

=

Law 2: A . B = B . A

Proof

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

=

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

This law can be extended to any number of variables. For example
A.B. C = B. C. A = C. A. B = B. A. C

5. Associative Laws:-

The associative laws allow grouping of variables. There are 2 associative laws.

Law 1: (A + B) + C = A + (B + C)

DUALITY:-

The implication of the duality concept is that once a theorem or statement is proved, the dual also thus standproved. This is called the principle of duality.

DUALS:-

Given expression	*Dual*
1. $\overline{0} = 1$	$\overline{1} = 0$
2. $0 \cdot 1 = 0$	$1 + 0 = 1$
3. $0 \cdot 0 = 0$	$1 + 1 = 1$

4.	$1 \cdot 1 = 1$	$0 + 0 = 0$
5.	$A \cdot 0 = 0$	$A + 1 = 1$
6.	$A \cdot 1 = A$	$A + 0 = A$
7.	$A \cdot A = A$	$A + A = A$
8.	$A \cdot A = 0$	$A + A = 1$
9.	$A \cdot B = B \cdot A$	$A + B = B + A$
10.	$A \cdot (B \cdot C)=(A \cdot B) \cdot C$	$A + (B + C)=(A + B) + C$
11.	$A \cdot (B + C) = AB + AC$	$A + BC = (A + B) (A + C)$
12.	$A(A + B) = A$	$A + AB = A$
13.	$\underline{A} \cdot (\underline{A} \cdot \underline{B}) = A \cdot B$	$\underline{A + A} + \underline{B} \underline{=} A + B$
14.	$AB = A + B$	$A + B = A B$
15.	$(A + B) (A+ C) (B + C) = (A+ B)(A + C)$	$AB + AC + BC = AB + AC$
16.	$A + BC = (A + B)(A + C)$	$A(B+ C) = A B +A C$
17.	$(A+C)(A+B) = AB+AC$	$AC+AB=(A+B) (A+C)$
18.	$(A+B)(C+D) = AC + AD + BC + BD$	$(AB+CD) = (A+C)(A+D)(B+C)(B+D)$
19.	$\underline{A + B = AB +} AB + AB$	$\underline{AB =(A+B) (A+B)} (A+B)$
20.	$AB + A + AB = 0$	$A + B \cdot A \cdot (A + B) = 1$

SUM - OF - PRODUCTS FORM:-

- This is also called disjunctive Canonical Form (DCF) or Expanded Sum of Products Form or Canonical Sum of Products Form.
- In this form, the function is the sum of a number of products terms where each product term contains all variables of the function either in complemented or uncomplemented form.
- This can also be derived from the truth table by finding the sum of all the terms that corresponds to those combinations for which ‘f ’ assumes the value 1.

  For example

  $$f(A, B, C) = A\overline{B} + BC$$

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

  $$= A\,BC + ABC + ABC + ABC$$
- The product term which contains all the variables of the functions either in complemented or uncomplemented form is called a minterm.
- The minterm is denoted as mo, m1, m2 … .
- An ‘n’ variable function can have 2n minterms.
- Another way of representing the function in canonical SOP form is the showing the sum of minterms for which the function equals to 1.

  For example

  $f(A, B, C) = m_1 + m_2 + m_3 + m_5$

  or

  $f(A, B, C) = \sum m(1, 2, 3, 5)$

  where ∑m represents the sum of all the minterms whose decimal codes are given the parenthesis.

PRODUCT- OF - SUMS FORM:-

- This form is also called as Conjunctive Canonical Form (CCF) or Expanded Product - of – Sums Form or Canonical Product Of Sums Form.
- This is by considering the combinations for which f = 0
- Each term is a sum of all the variables.
- The function $f(A, B, C) = (\overline{A} + \overline{B} + C\cdot\overline{C}) + (\overline{A} + B + C\cdot C)$

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

- The sum term which contains each of the ‘n’ variables in either complemented or uncomplemented form is called a maxterm.
- Maxterm is represented as $M_0, M_1, M_2, \ldots\ldots.$

  Thus CCF of ‘f’ may be

  written as f(A, B, C)=

  $M_0 \cdot M_4 \cdot M_6 \cdot M_7$

or

$f(A, B, C) = (0, 4, 6, 7)$

Where represented the product of all maxterms.

CONVERSION BETWEEN CANONICAL FORM:-

The complement of a function expressed as the sum of minterms equals the sum of minterms missing from the original function.

KARNAUGH MAP OR K- MAP:-

- The K- map is a chart or a graph, composed of an arrangement of adjacent cells, each representing a particular combination of variables in sum or product form.
- The K- map is systematic method of simplifying the Boolean expression.

TWO VARIABLE K- MAP:-

A two variable expression can have $2^2 = 4$ possible combinations of the input variables A and B.

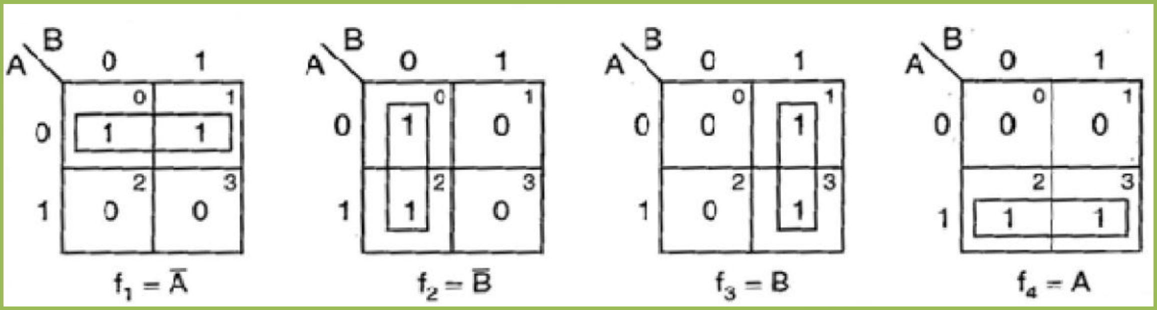
Mapping of SOP Expression:-

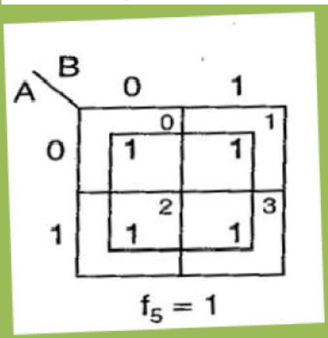
- The 2 variable K-map has $2^2 = 4$ squares. These squares are called cells.
- A ‘1’ is placed in any square indicates that corresponding minterm is included in the output expression, and a 0 or no entry in any square indicates that the corresponding minterm does not appear in the expression for output.

Minimization of SOP Expression:-

To minimize a Boolean expression given in the SOP form by using K- map, the adjacent squares having 1s, that is minterms adjacent to each other are combined to form larger squares to eliminate some variables.

The possible minterm grouping in a two variable K- map are shown below

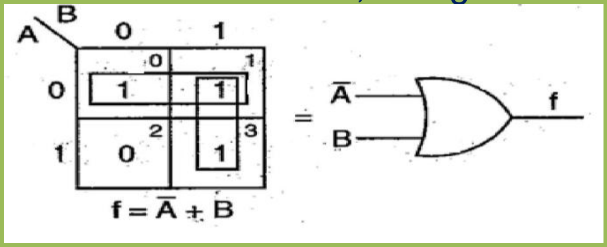




- Two minterms, which are adjacent to each other, can be combined to form a bigger square called 2 – square or a pair. This eliminates one variable that is not common to both the minterms.
- Two 2-squares adjacent to each other can be combined to form a 4- square. A 4-square eliminates 2 variables. A 4-square is called a quad.
- Consider only those variables which remain constant throughout the square, and ignore the variables which are varying. The non-complemented variable is the variable remaining constant as 1.The complemented variable is the variable remaining constant as a 0 and the variables are written as a product term.

Example:-
Reduce the expression $f= \bar{A}\bar{B} + \bar{A} B + AB$ using mapping.

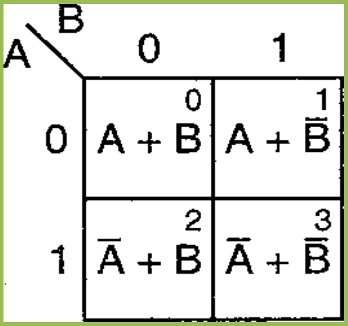
Solution:-
Expressed in terms of minterms, the given



expression is f = $m_0 + m_1 + m_3$ =
$\sum m$ (0, 1, 3)

Mapping of POS Expression:-

Each sum term in the standard POS expression is called a Maxterm. A function in two variables (A,B) has 4 possible maxterms, A + B, A + B, A + B and A + B . They are represented as M_0, M_1, M_2 and M_3 respectively.

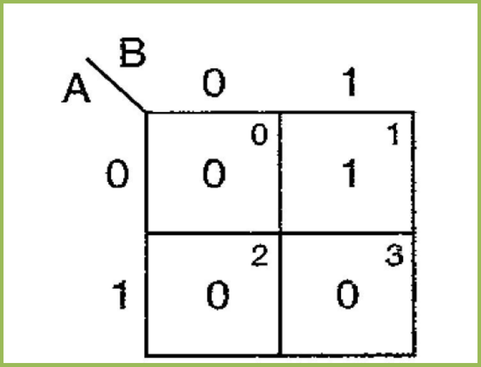


The maxterm of a two variable

K-map Example:-

Plot the expression f= (A + B)(A + B (A + B) Solution:-

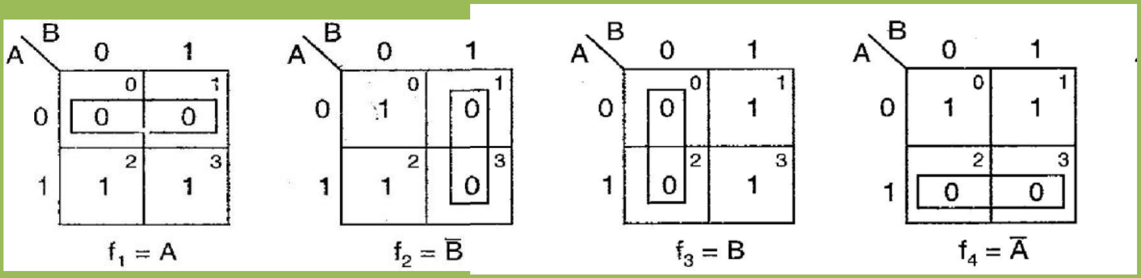
Expression interms of maxterms is f = пM (0, 2, 3)

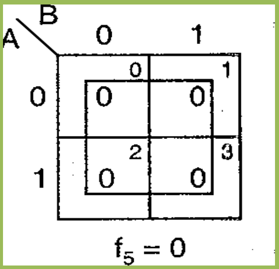


Minimization of POS Expressions:-

In POS form the adjacent 0s are combined into large square as possible. If the squares having complemented variable then the value remain constant as a 1 and the non-complemented variable if its value remains constant as a 0 along the entire square and

then their sum term is written.
The possible maxterms grouping in a two variable K-map are shown below

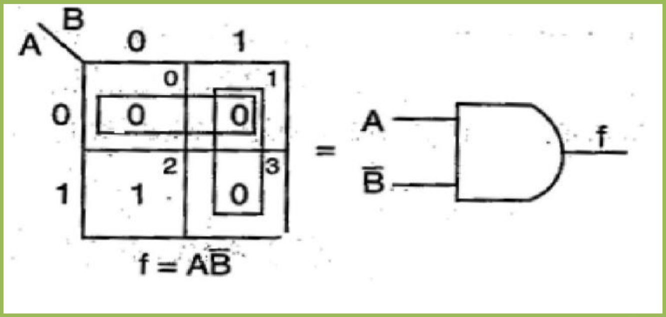




Example:- _ _ _

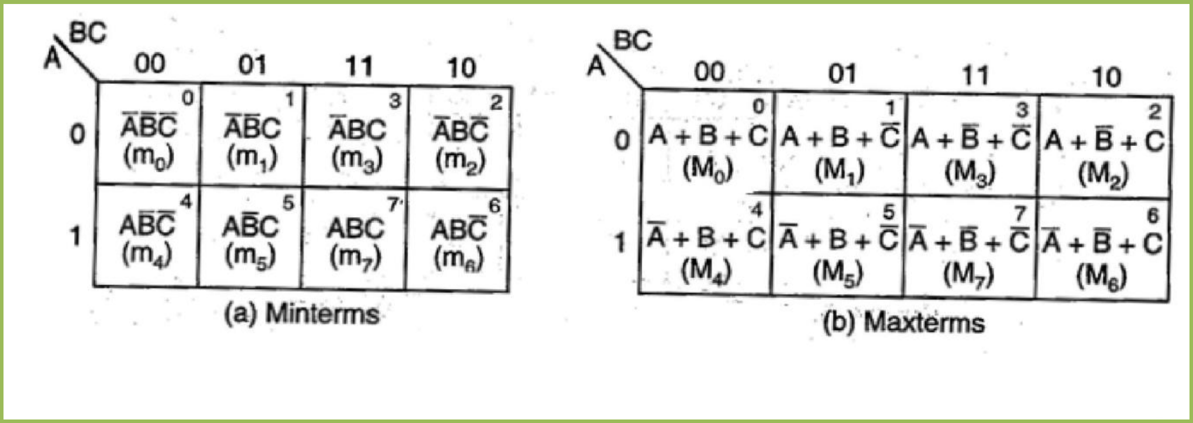
Reduce the expression f = (A + B)(A + B)(A +B) using mapping Solution:-

The given expression in terms of maxterms is f = пM (0, 1, 3)



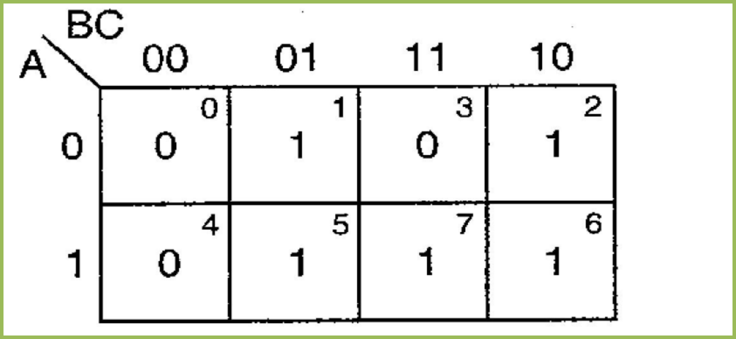
<u>**THREE VARIABLE K- MAP**</u>**:-**

A function in three variables (A, B, C) can be expressed in SOP and POS form having eight possible combination. A three variable K- map have 8 squares or cells on the map and each square represents a minterm or maxterm is shown in the figure below.



Example:- _ _ __ _ _ _

Map the expression f = ABC+ABC + ABC + ABC +ABC Solution:-



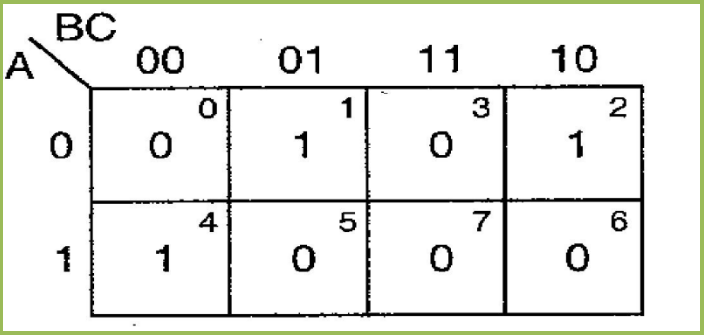
A \ BC	00	01	11	10
0	0 (0)	1 (1)	0 (3)	1 (2)
1	0 (4)	1 (5)	1 (7)	1 (6)

So in the SOP form the expression is f = ∑ m (1, 5, 2, 6, 7)

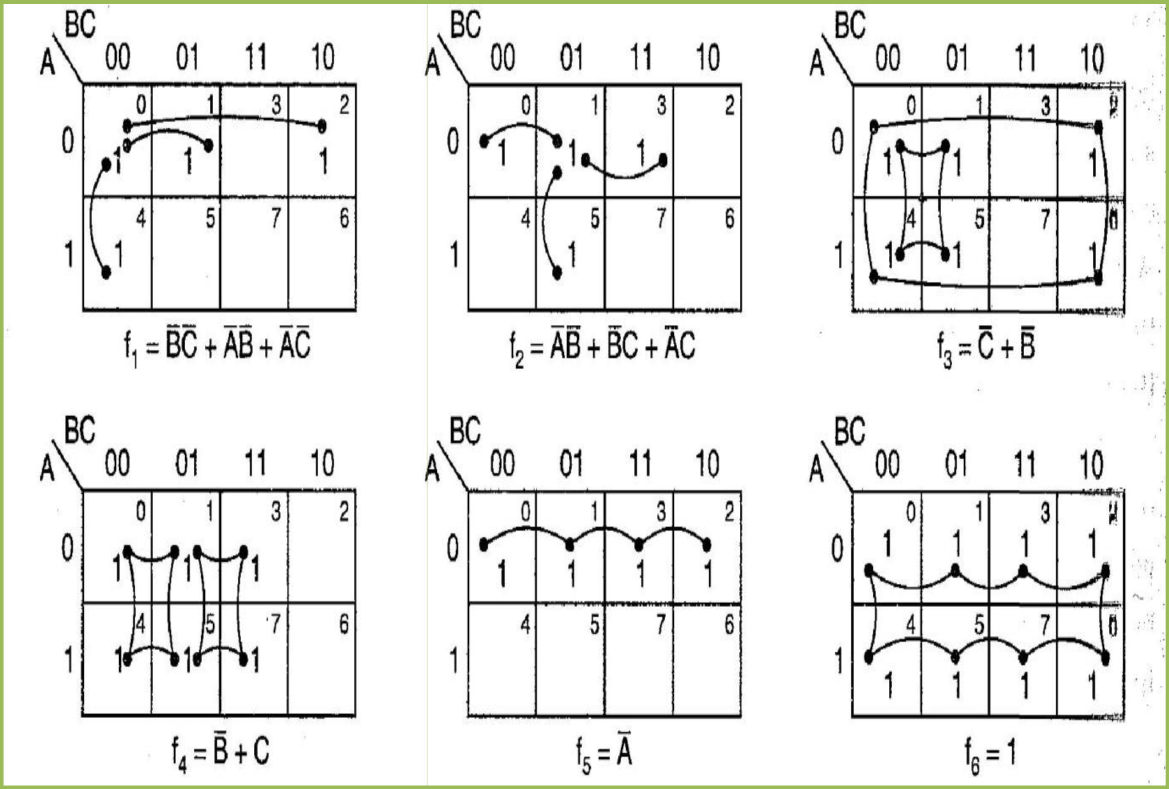
Example:-

Map the expression $f = (A + B + C)(\overline{A} + B + \overline{C})(\overline{A} + \overline{B} + \overline{C})(A + \overline{B} + \overline{C})(\overline{A} + \overline{B} + C)$ Solution:-

So in the POS form the expression is f = п M (0, 5, 7, 3, 6)



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


$f_1 = \bar{B}\bar{C} + \bar{A}\bar{B} + \bar{A}\bar{C}$
$f_2 = \bar{A}\bar{B} + \bar{B}C + \bar{A}C$
$f_3 = \bar{C} + \bar{B}$
$f_4 = \bar{B} + C$
$f_5 = \bar{A}$
$f_6 = 1$
BC
A
00
01
11
10
0
1

Minimization of SOP and POS Expressions:-

For reducing the Boolean expressions in SOP (POS) form the following steps are given below

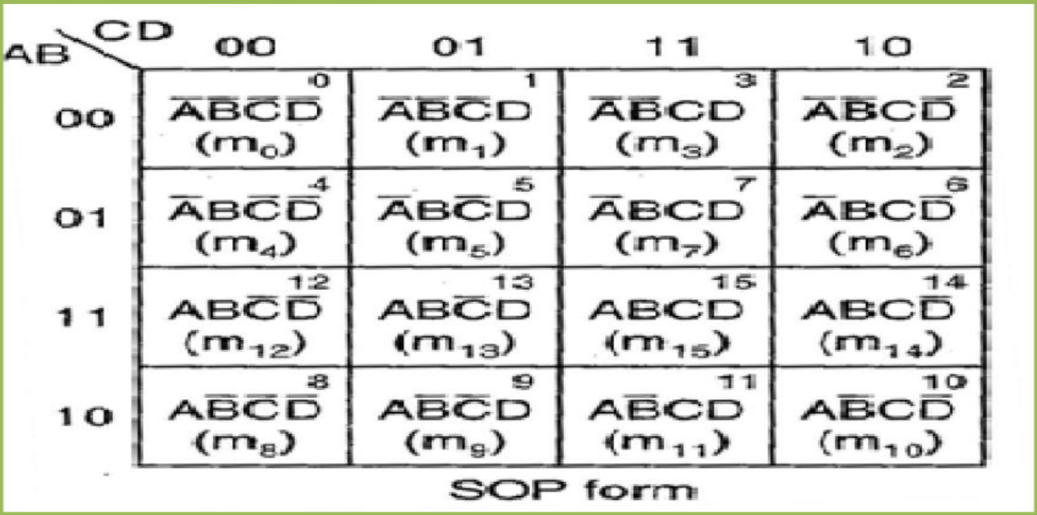
- Draw the K-map and 1s (0s) corresponding to the minterms (maxterms) of the placeexpression. SOP (POS)
- In the map 1s (0s) which are not adjacent to any other 1(0) are the isolated minterms (maxterms). They are to be read as they are because they cannot be combined even into a 2-square.
- For those 1s (0s) which are adjacent to only one other 1(0) make them pairs (2 squares).
- For quads (4- squares) and octet (8 squares) of adjacent 1s (0s) even if they contain some 1s (0s) which have already been combined. They must geometrically form a square or a rectangle.
- For any 1s (0s) that have not been combined yet then combine them into bigger squares if possible.
- Form the minimal expression by summing (multiplying) the product (sum) terms of all the groups.

Some of the possible combinations of minterms in SOP form

FOUR VARIABLE K-MAP:-

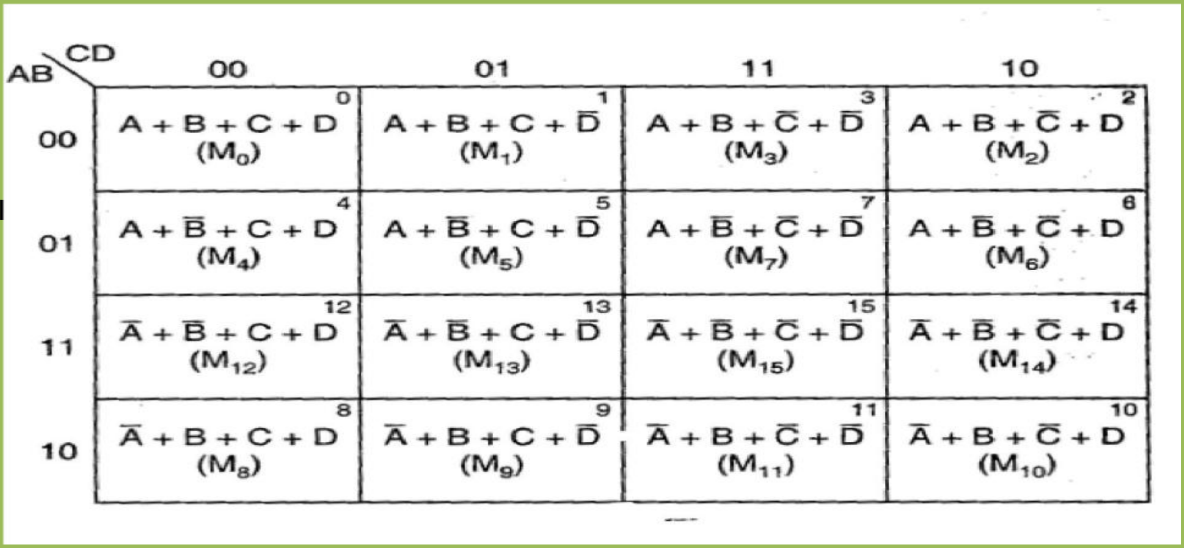
A four variable (A, B, C, D) expression can have 2^4 = 16 possible combinations of input variables. A four variable K-map has 2^4 = 16 squares or cells and each square on the map represents either a minterm or a maxterm as shown in the figure below. The binary number designations of the rows and columns are in the gray code. The binary numbers along the top of the map indicate the conditions of C and D along any column and binary numbers along left side indicate the conditions of A and B along any row. The numbers in the top right corners of the squares indicate the minterm or maxterm desginations.

SOP FORM



AB \ CD	00	01	11	10
00	0 $\bar{A}\bar{B}\bar{C}\bar{D}$ (m_0)	1 $\bar{A}\bar{B}\bar{C}D$ (m_1)	3 $\bar{A}\bar{B}CD$ (m_3)	2 $\bar{A}\bar{B}C\bar{D}$ (m_2)
01	4 $\bar{A}B\bar{C}\bar{D}$ (m_4)	5 $\bar{A}B\bar{C}D$ (m_5)	7 $\bar{A}BCD$ (m_7)	6 $\bar{A}BC\bar{D}$ (m_6)
11	12 $AB\bar{C}\bar{D}$ (m_{12})	13 $AB\bar{C}D$ (m_{13})	15 $ABCD$ (m_{15})	14 $ABC\bar{D}$ (m_{14})
10	8 $A\bar{B}\bar{C}\bar{D}$ (m_8)	9 $A\bar{B}\bar{C}D$ (m_9)	11 $A\bar{B}CD$ (m_{11})	10 $A\bar{B}C\bar{D}$ (m_{10})

SOP form



AB \ CD	00	01	11	10
00	0 $A+B+C+D$ (M_0)	1 $A+B+C+\bar{D}$ (M_1)	3 $A+B+\bar{C}+\bar{D}$ (M_3)	2 $A+B+\bar{C}+D$ (M_2)
01	4 $A+\bar{B}+C+D$ (M_4)	5 $A+\bar{B}+C+\bar{D}$ (M_5)	7 $A+\bar{B}+\bar{C}+\bar{D}$ (M_7)	6 $A+\bar{B}+\bar{C}+D$ (M_6)
11	12 $\bar{A}+\bar{B}+C+D$ (M_{12})	13 $\bar{A}+\bar{B}+C+\bar{D}$ (M_{13})	15 $\bar{A}+\bar{B}+\bar{C}+\bar{D}$ (M_{15})	14 $\bar{A}+\bar{B}+\bar{C}+D$ (M_{14})
10	8 $\bar{A}+B+C+D$ (M_8)	9 $\bar{A}+B+C+\bar{D}$ (M_9)	11 $\bar{A}+B+\bar{C}+\bar{D}$ (M_{11})	10 $\bar{A}+B+\bar{C}+D$ (M_{10})

Minimization of SOP and POS Expressions:-

For reducing the Boolean expressions in SOP (POS) form the following steps are given below

- Draw the K-map and place 1s (0s) corresponding to the minterms (maxterms) of the SOP (POS) expression.
- In the map 1s (0s) which are not adjacent to any other 1(0) are the isolated minterms (maxterms). They are to be read as they are because they cannot be combined even into a 2-square.
- For those 1s (0s) which are adjacent to only one other 1(0) make them pairs (2 squares).
- For quads (4- squares) and octet (8 squares) of adjacent 1s (0s) even if they

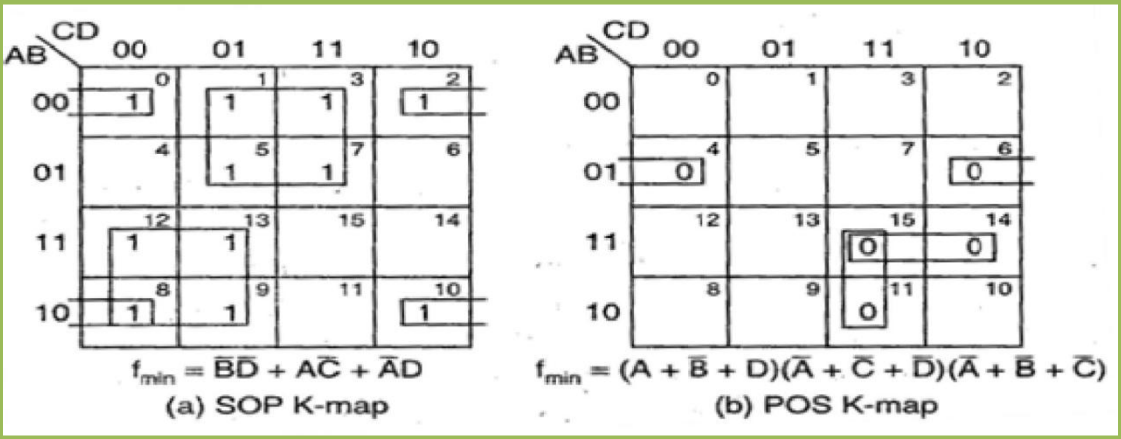
contain some 1s (0s) which have already been combined. They must geometrically form a square or a rectangle.

- For any 1s (0s) that have not been combined yet then combine them into bigger squares if possible.
- Form the minimal expression by summing (multiplying) the product (sum) terms of all the groups.

Example:-

Reduce using mapping the expression f = ∑ m (0, 1, 2, 3, 5, 7, 8, 9, 10, 12, 13)Solution:-

The given expression in POS form is f = π M (4, 6, 11, 14, 15) and in SOP form f = ∑ m (0, 1, 2, 3, 5, 7, 8, 9,
10, 12, 13)



$f_{min} = \overline{B}\,\overline{D} + A\overline{C} + \overline{A}D$
(a) SOP K-map

$f_{min} = (A + \overline{B} + D)(\overline{A} + \overline{C} + \overline{D})(\overline{A} + \overline{B} + \overline{C})$
(b) POS K-map

The minimal SOP expression is $f_{min} = \overline{B}\,\overline{D} + A\overline{C} + \overline{A}D$

The minimal POS expression is $f_{min} = (A + \overline{B} + D) (\overline{A} + \overline{C} + \overline{D}) (\overline{A} + \overline{B} + \overline{C})$

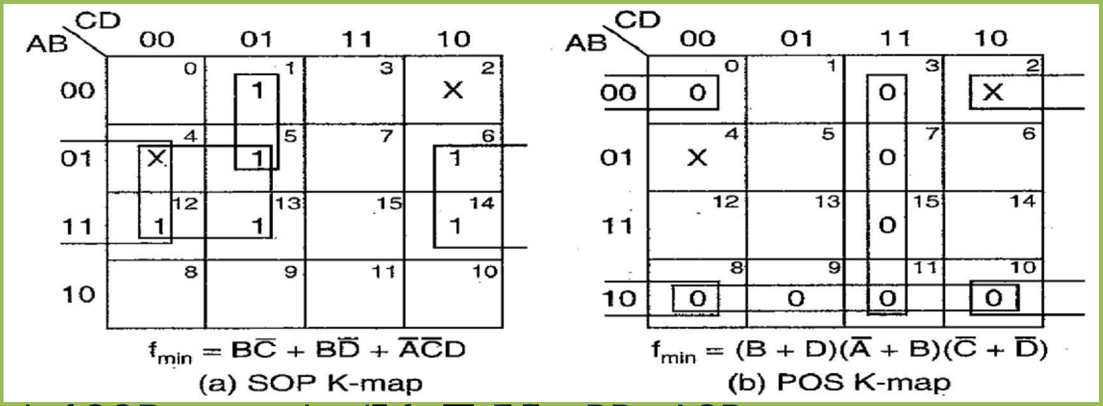
DON'T CARE COMBINATIONS:-

The combinations for which the values of the expression are not specified are called don't care combinations or optional combinations and such expression stand incompletely specified. The output is a don't care for these invalid combinations. The don't care terms are denoted by d or X. During the process of designing using SOP maps, each don't care is treated as 1 to reduce the map otherwise it is treated as 0 and left alone. During the process of designing using POS maps, each don't care is treated as 0 to reduce the map otherwise it is treated as 1 and left alone.

A standard SOP expression with don't cares can be converted into standard POS form by keeping the don't cares as they are, and the missing minterms of the SOP form are written as the maxterms of the POS form. Similarly, to convert a standard POS expression with don't cares can be converted into standard SOP form by keeping the don't cares as they are, and the missing maxterms of the POS form are written as the minterms of the SOP form.

Example:-
Reduce the expression f = ∑ m(1, 5, 6, 12, 13, 14) + d(2, 4)
using K- map. Solution:-

The given expression in SOP form is f = ∑ m (1, 5, 6, 12, 13, 14) + d(2, 4)
The given expression in POS form is f = π M (0, 3, 7, 8, 9, 10, 11,15) + d(2, 4)



$f_{min} = B\overline{C} + B\overline{D} + \overline{A}\overline{C}D$
(a) SOP K-map

$f_{min} = (B + D)(\overline{A} + B)(\overline{C} + \overline{D})$
(b) POS K-map

The minimal of SOP expression is f_{min} = BC + BD +ACD

The minimal of POS expression is f_{min} = (B + D)(A + B) (C + D)

LOGIC GATES

- Logic gates are the fundamental building blocks of digital systems.
- There are 3 basic types of gates AND, OR and NOT.
- Logic gates are electronic circuits because they are made up of a number of electronic devices and components.
- Inputs and outputs of logic gates can occur only in 2 levels. These two levels are termed HIGH and LOW, or TRUE and FALSE, or ON and OFF or simply 1 and 0.
- The table which lists all the possible combinations of input variables and the corresponding outputs is called a truth table.

LEVEL LOGIC:-

A logic in which the voltage levels represents logic 1 and logic 0. Level logic may be positive or negative logic. **Positive Logic:-**

A positive logic system is the one in which the higher of the two voltage levels represents the logic 1 and the lower of the two voltages level represents the logic 0.

Negative Logic:-

A negative logic system is the one in which the lower of the two voltage levels represents the logic 1 and the higher of the two voltages level represents the logic 0.

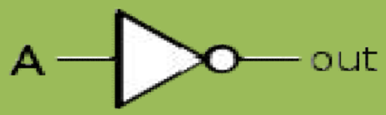
DIFFERENT TYPES OF LOGIC GATES:-

NOT GATE (INVERTER):-

- A NOT gate, also called and inverter, has only one input and one output.
- It is a device whose output is always the complement of its input.
- The output of a NOT gate is the logic 1 state when its input is in logic 0 state and the logic 0 state when its inputs is in logic 1 state.

IC No. :- 7404

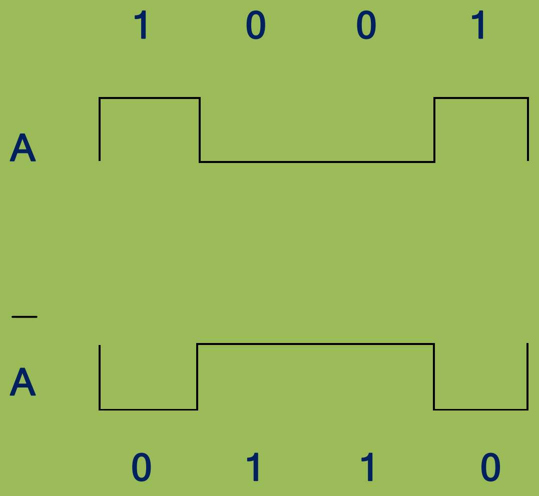
Logic Symbol



INPUT A	OUTPUT $\overline{A}$
0	1
1	0

Truth table

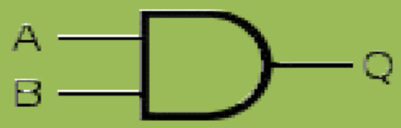
Timing Diagram



AND GATE:-

- An AND gate has two or more inputs but only one output.
- The output is logic 1 state only when each one of its inputs is at logic 1 state.
- The output is logic 0 state even if one of its inputs is at logic 0 state.

IC No.:- 7408

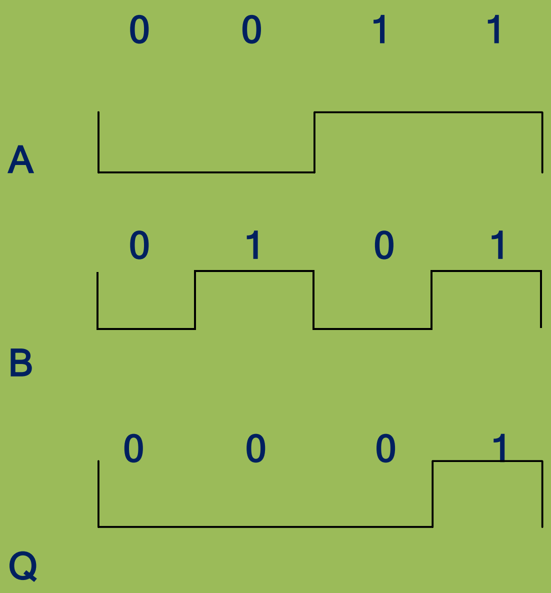


Logic Symbol

		OUTPUT
A	**B**	**Q=A . B**
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table

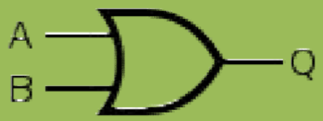
Timing Diagram



OR GATE:-

- An OR gate may have two or more inputs but only one output.
- The output is logic 1 state, even if one of its input is in logic 1 state.
- The output is logic 0 state, only when each one of its inputs is in logic state.

IC No.:- 7432

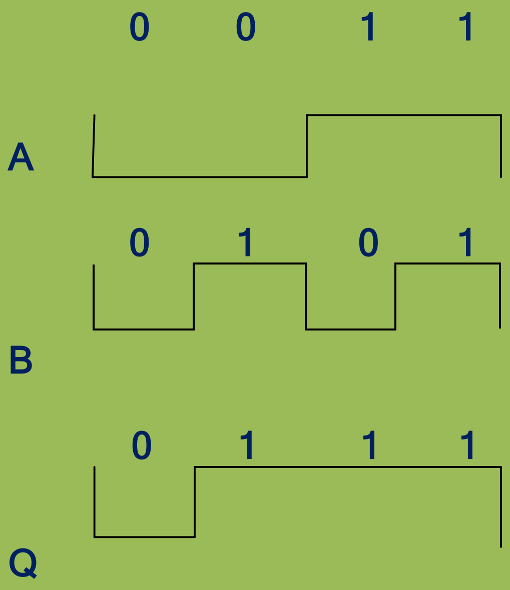


Logic Symbol

INPUT		OUTPUT
A	**B**	**Q=A + B**
0	0	0
0	1	1
1	0	1
1	1	1

Truth Table

Timing Diagram



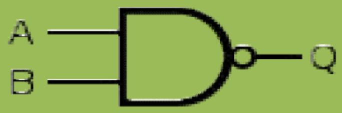
NAND GATE:-

- NAND gate is a combination of an AND gate and a NOT gate.
- The output is logic 0 when each of the input is logic 1 and for any other combination of inputs, theoutput is logic 1.

IC No.:- 7400 two input NAND

gate 7410 three input NAND gate 7420 four input NAND gate 7430 eight input NAND gate

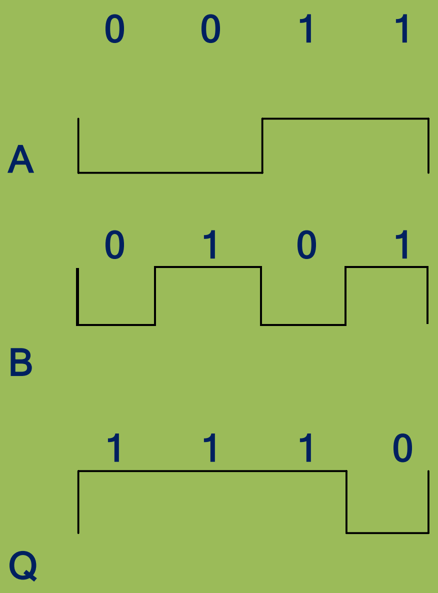
Logic Symbol



Truth Table

INPUT		OUTPUT
A	**B**	**Q=** $\underline{A . B}$
0	0	1
0	1	1
1	0	1
1	1	0

Timing Diagram



<u>NOR GATE</u>:-

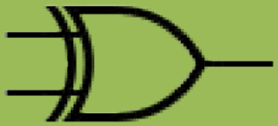
- NOR gate is a combination of an OR gate and a NOT gate.
- The output is logic 1, only when each one of its input is logic 0 and for any other combination of inputs, the output is a logic 0 level.

<u>EXCLUSIVE - OR (X-OR) GATE</u>:-

- An X-OR gate is a two input, one output logic circuit.
- The output is logic 1 when one and only one of its two inputs is logic 1. When both the inputs is logic 0 or when both the inputs is logic 1, the output is logic 0.

IC No.:- 7486

Logic Symbol



INPUTS are **A** and **B**

OUTPUT is **Q** = $A \oplus B$

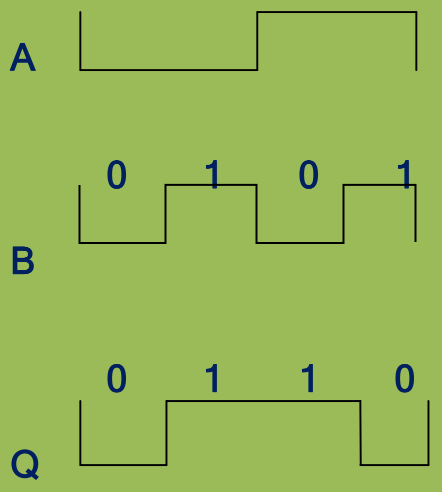
= A B + A B

Truth Table

INPUT		OUTPUT
A	**B**	**Q = $A \oplus B$**
0	0	0
0	1	1
1	0	1
1	1	0

Timing Diagram

0 0 1 1

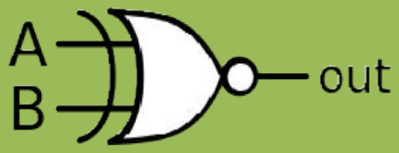


EXCLUSIVE – NOR (X-NOR) GATE:-

- An X-NOR gate is the combination of an X-OR gate and a NOT gate.
- An X-NOR gate is a two input, one output logic circuit.
- The output is logic 1 only when both the inputs are logic 0 or when both the inputs is 1.
- The output is logic 0 when one of the inputs is logic 0 and other is 1.

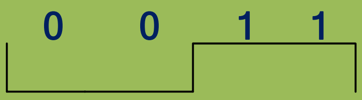
IC No.:- 74266

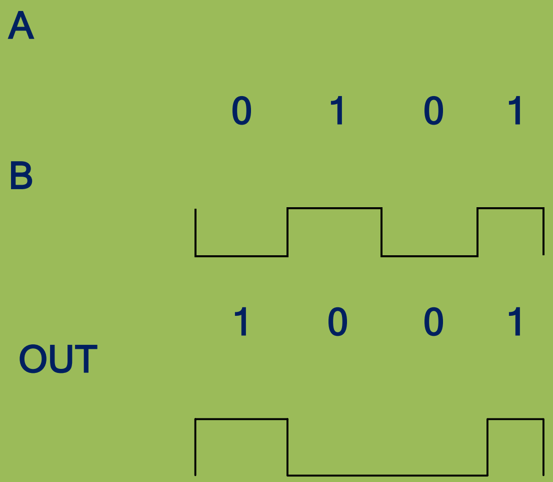
Logic Symbol



INPUT		OUTPUT
A	B	OUT =A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

Timing Diagram



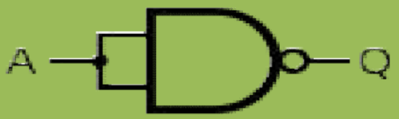

A
0 1 0 1
B
1 0 0 1
OUT

UNIVERSAL GATES:-

There are 3 basic gates AND, OR and NOT, there are two universal gates NAND and NOR, each of which can realize logic circuits single handedly. The NAND and NOR gates are called universal building blocks. Both NAND and NOR gates can perform all logic functions i.e. AND, OR, NOT, EXOR and EXNOR.

NAND GATE:-

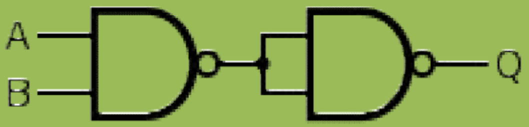
a) Inverter from NAND gate



Input = A
Output **Q** = $\overline{A}$

b) AND gate from NAND gate

Input s are **A**

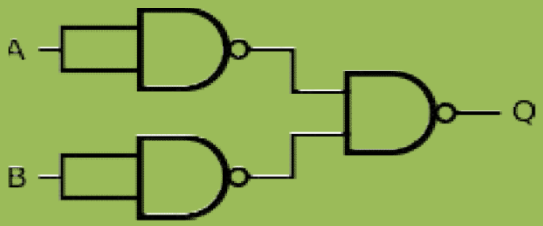


and **B** Output **Q**
= A.B

c)

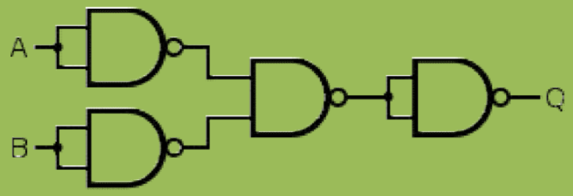
d) OR gate from NAND gate

Inputs are **A**
and **B** Output
Q = A+B



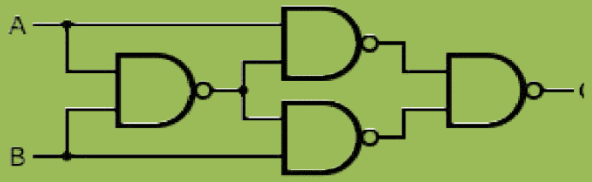
e) NOR gate from NAND gate

Inputs are A and B Output Q = A+B



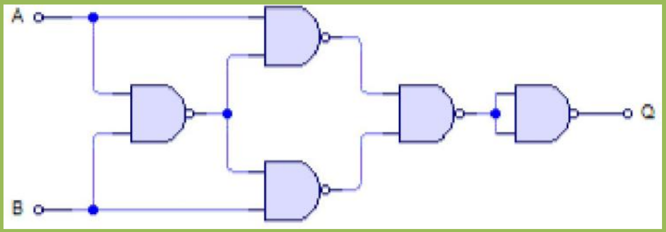
f) EX-OR gate from NAND gate

Inputs are A and B Output Q = A B + AB



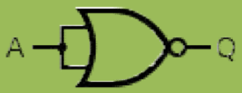
g) EX-NOR gate From NAND gate

Inputs are A and B
Output Q = $A \overline{B} + A B$



NOR GATE:-

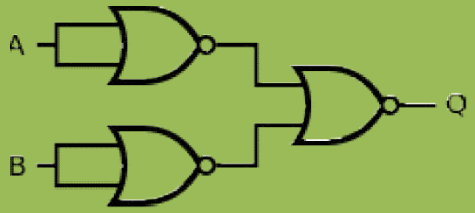
a) Inverter from NOR gate Input = A



Output Q = A

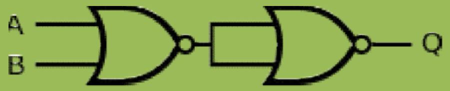
b) AND gate from NOR gateInput s are A

c) and B Output Q = A.B



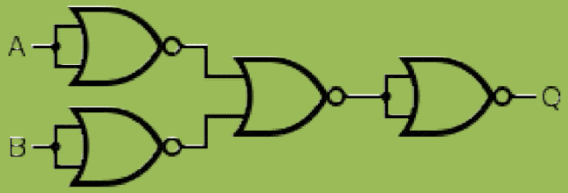
a) OR gate from NOR gate

Inputs are A and B Output Q = A+B



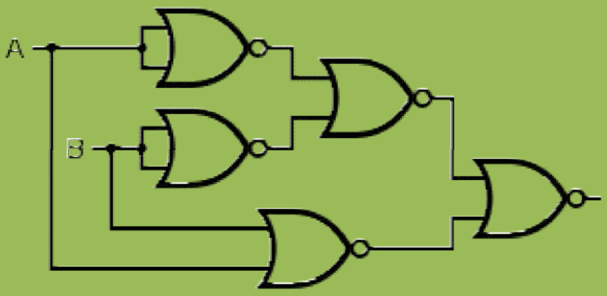
b) NAND gate from NOR gate

Inputs are A and B Output Q = A.B



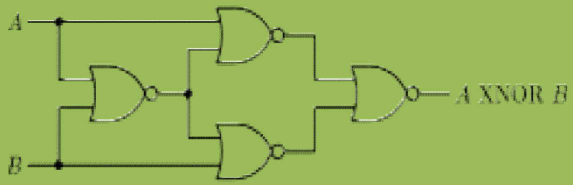
c) EX-OR gate from NOR gate

Inputs are **A** and
B Output **Q** = $\bar{\text{A}}$
B + **AB**



d) <u>**EX-NOR gate From NOR gate**</u>

Inputs are **A** and **B**
Output **Q = A $\bar{\text{B}}$ + A B**



<u>THRESHOLD LOGIC</u>:-

<u>INTRODUCTION</u>:-

- The threshold element, also called the threshold gate (T-gate) is a much more powerful device than any of the conventional logic gates such as NAND, NOR and others.
- Complex, large Boolean functions can be realized using much fewer threshold gates.
- Frequently a single threshold gate can realize a very complex function which otherwise might require a large number of conventional gates.
- T-gate offers incomparably economical realization; it has not found extensive use with the digital system designers mainly because of the following limitations.
  1. It is very sensitive to parameter variations.
  2. It is difficult to fabricate it in IC form.

  The speed of switching of threshold elements in much lower than that of conventional gates.

THE THRESHOLD ELEMENTS:-

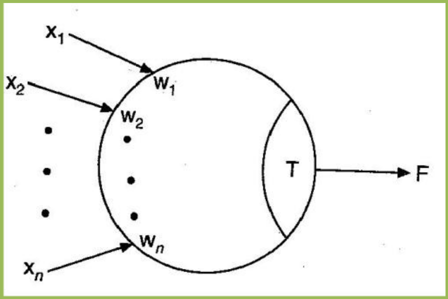
- A threshold element or gate has 'n' binary inputs x_1, x_2, ….., x_n; and a single binary output F. But in addition to those, it has two more parameters.
- Its parameters are a threshold T and weights w_1, w_2, ….,w_n. The weights w_1, w_2, …, w_n are associated with the input variables x_1, x_2, …, x_n.
- The value of the threshold (T) and weights may be real, positive or negative number.
- The symbol of the threshold element is shown in fig.(a).
- It is represented by a circle partitioned into two parts, one part represents the weights and other represents T.
- It is defined as

$$F(x_1, x_2, \ldots, x_n) = 1 \text{ if and only if } \sum_{i=1}^{n} w_i x_i \geq T$$

otherwise

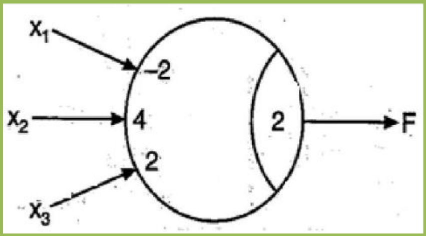
$$F(x_1, x_2, \ldots, x_n) = 0$$

- The sum and product operation are normal arithmetic operations and the sum $\sum_{i=1}^{n} w_i x_i \geq T$ is called the weighted sum of the element or gate.



Example:-

Obtain the minimal Boolean expression from the threshold gate shown in figure.



Solution:-

The threshold gate with three inputs x_1, x_2, x_3 with weights -2(w_1) , 4(w_2) and 2(w_3) respectively. The value of threshold is 2(T). The table shown is the weighted sums and outputs for all input combinations. For this threshold gate, the weighted sum is

$$w = w_1x_1 + w_2x_2 + w_3x_3$$

$$= (-2)x_1 + (4)x_2 + (2)x_3$$

$$= -2x_1 + 4x_2 + 2x_3$$

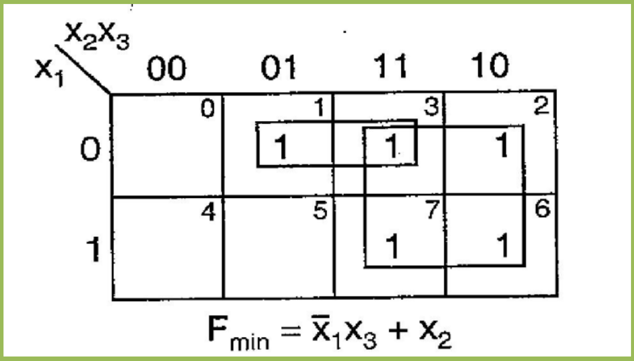
The output F is logic 1 for $w \geq 2$ and it is logic 0 for $w<2$

Input Variables			**Weighted Sum**	**Output**
x_1	x_2	x_3	$w = -2x_1 + 4x_2 + 2x_3$	**F**
0	0	0	0	0
0	0	1	2	1
0	1	0	4	1
0	1	1	6	1
1	0	0	-2	0
1	0	1	0	0
1	1	0	2	1
1	1	1	4	1

From the input – output relation is given in the table, the Boolean expression

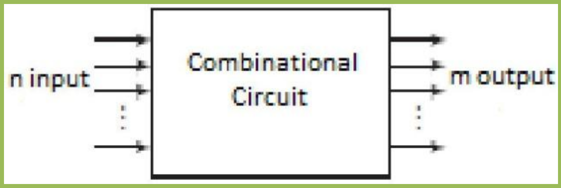
for the output is $F = \sum m\ (1, 2, 3, 6, 7)$

The K-map for F is



COMBINATIONAL LOGIC CIRCUIT

- A combinational circuit consists of logic gates whose outputs at any time are determined from only the present combination of inputs.
- A combinational circuit performs an operation that can be specified logically by a set of Boolean functions.
- It consists of an interconnection of logic gates. Combinational logic gates react to the values of the signals at their inputs and produce the value of the output signal, transforming binary information from the given input data to a required output data.
- A block diagram of a combinational circuit is shown in the below figure.
- The n input binary variables come from an external source; the m output variables are produced by the internal combinational logic circuit and go to an external destination.
- Each input and output variable exists physically as an analog signal whose values are interpreted to be a binary signal that represents logic 1and logic 0.



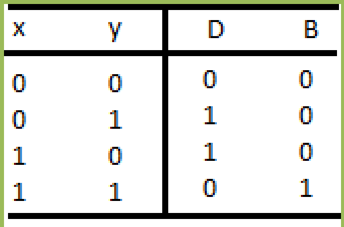
BINARY ADDER-SUBTRACTOR:-

- Digital computers perform a variety of information-processing tasks. Among the functions encountered are the various arithmetic operations.
- The most basic arithmetic operation is the addition of two binary digits. This simple addition consists of four possible elementary operations: 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1, and 1 + 1 = 10.
- The first three operations produce a sum of one digit, but when both augend and addend bits are equal to 1; the binary sum consists of two digits. The higher significant bit of this result is called a carry.
- When the augend and addend numbers contain more significant digits, the carry obtained from the addition of two bits is added to the next higher order pair of significant bits.

- A combinational circuit that performs the addition of two bits is called a half adder.
- One that performs the addition of three bits (two significant bits and a previous carry) is a full adder. The names of the circuits stem from the fact that two half adders can be employed to implement a full adder.

HALF ADDER:-

- This circuit needs two binary inputs and two binary outputs.
- The input variables designate the augend and addend bits; the output variables produce the sum and carry. Symbols x and y are assigned to the two inputs and S (for sum) and C (for carry) to the outputs.
- The truth table for the half adder is listed in the below table.
- The C output is 1 only when both inputs are 1. The S output represents the least significant bit of the sum.
- The simplified Boolean functions for the two outputs can be obtained directly from



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

Truth Table

the truth table.

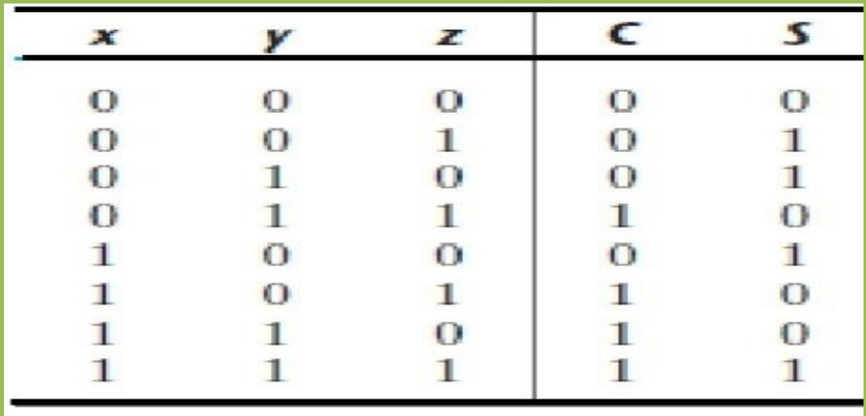
- The simplified sum-of-products expressions are

$$S = x'y + xy' \quad C = xy$$

- The logic diagram of the half adder implemented in sum of products is shown in the below figure. It can be also implemented with an exclusive-OR and an AND gate.

FULL ADDER:-

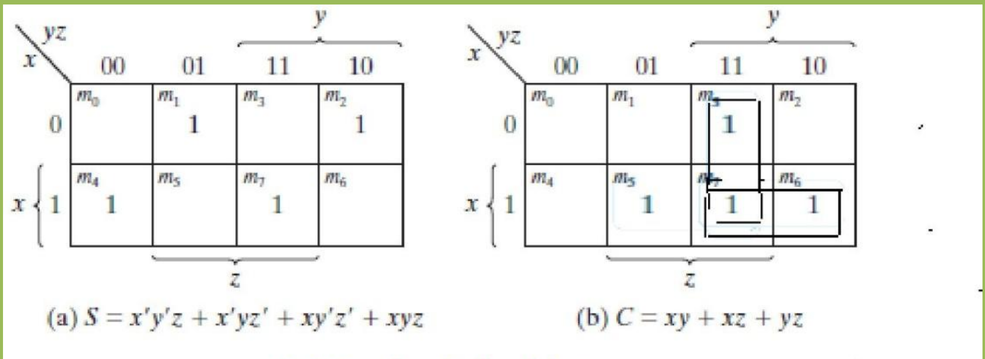
- A full adder is a combinational circuit that forms the arithmetic sum of three bits.
- It consists of three inputs and two outputs. Two of the input variables, denoted by x and y , represent the two significant bits to be added. The third input, z ,



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

Truth Table

- represents the carry from the previous lower significant position.
- Two outputs are necessary because the arithmetic sum of three binary digits ranges in value from 0 to 3, and binary representation of 2 or 3 needs two bits. The two outputs are designated by the symbols S for sum and C for carry.



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

(b) $C = xy + xz + yz$

K-Map for full adder

- The binary variable S gives the value of the least significant bit of the sum. The binary variable C gives the output carry formed by adding the input carry and the bits of the words.
- The eight rows under the input variables designate all possible combinations of the three variables. The

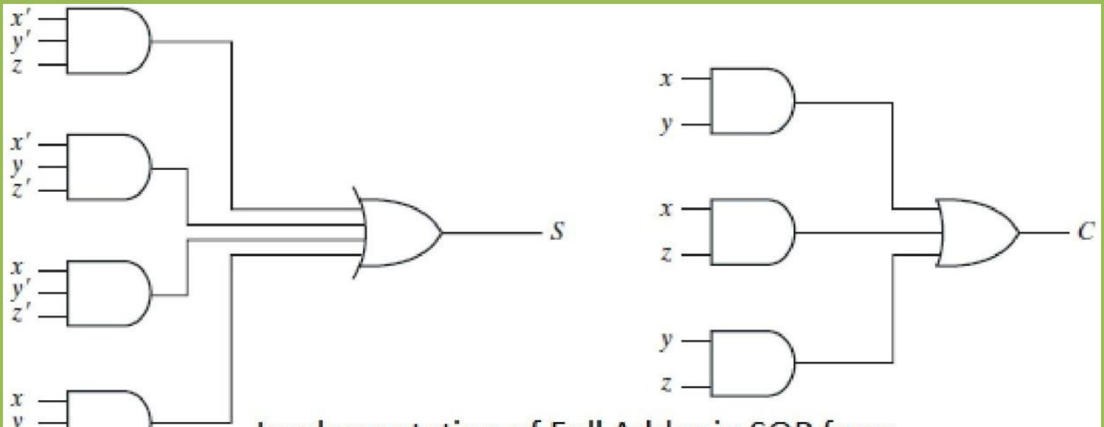
output variables are determined from the arithmetic sum of the input bits. When all input bits are 0, theoutput is 0.

- The S output is equal to 1 when only one input is equal to 1 or when all three inputs are equal to 1. TheC output has a carry of 1 if two or three inputs are equal to 1.
- The simplified expressions are

S = x’y’z + x’yz’ + xy’z’ + xyz

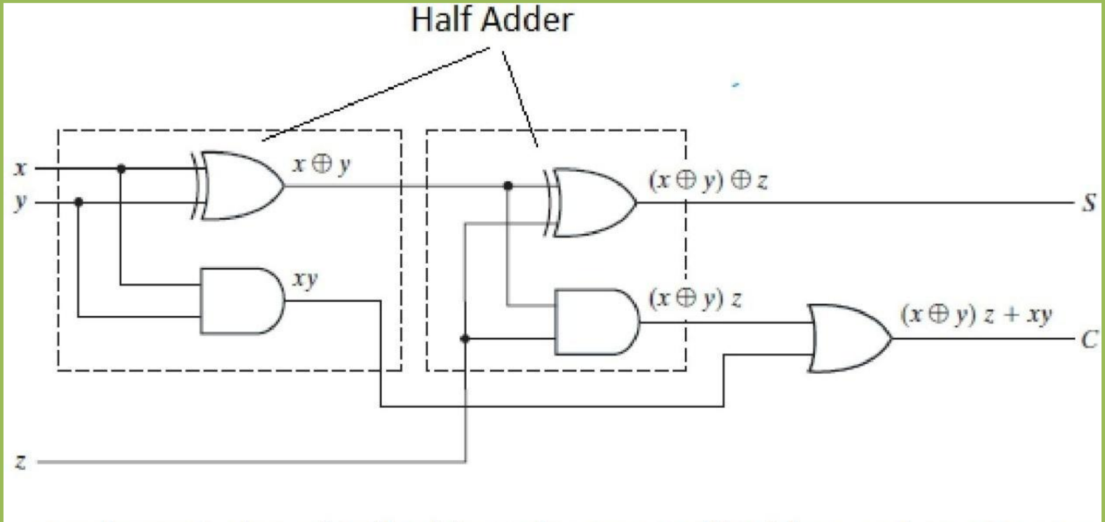
C = xy + xz + yz

- The logic diagram for the full adder implemented in sum-of-products form is shown in figure.



Implementation of Full Adder in SOP form

- It can also be implemented with two half adders and one OR gate as shown in the figure.



Implementation of Full Adder using Two Half Adders and an OR gate

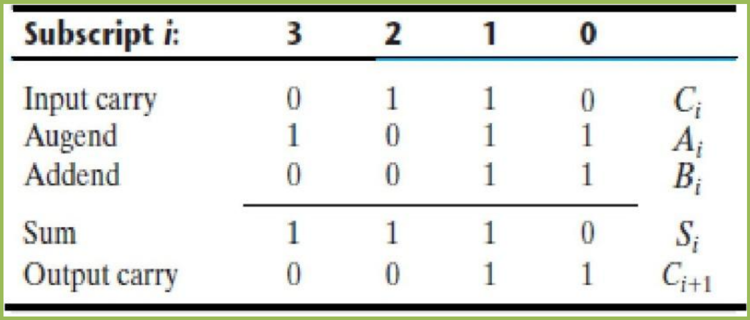
- A full adder is a combinational circuit that forms the arithmetic sum of three bits.

BINARY ADDER:-

- A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers.
- It can be constructed with full adders connected in cascade, with the output carry from each full adderconnected to the input carry of the next full adder in the chain.
- Addition of n-bit numbers requires a chain of n full adders or a chain of one-half adder and n-1 full adders. In the former case, the input carry to the least significant position is fixed at 0.
- The interconnection of four full-adder (FA) circuits to provide a four-bit binary ripple carry adder isshown in the figure.
- The augend bits of A and the addend bits of B are designated by subscript numbers from right to left,with subscript 0 denoting the least significant bit.

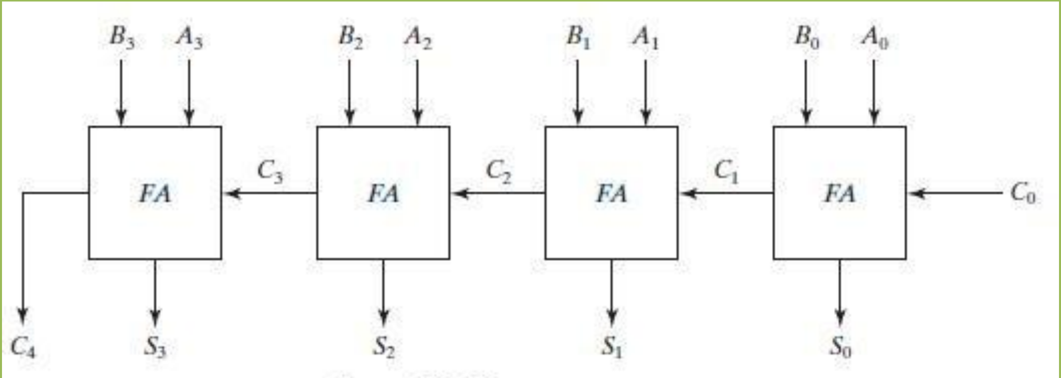
The carries are connected in a chain through the full adders. The input carry to the adder is C0, and itripples through the full adders to the output carry C4. The S outputs generate the required sum bits.

- An n -bit adder requires n full adders, with each output carry connected to the input carry of the nexthigher order full adder.
- Consider the two binary numbers A = 1011 and B = 0011. Their sum S = 1110 is formed with the four-bit adder as follows:



Subscript i:	3	2	1	0	
Input carry	0	1	1	0	C_i
Augend	1	0	1	1	A_i
Addend	0	0	1	1	B_i
Sum	1	1	1	0	S_i
Output carry	0	0	1	1	C_{i+1}

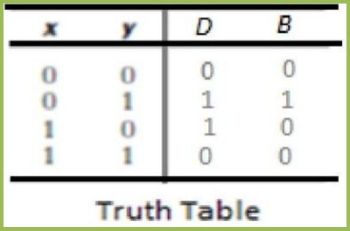
- The bits are added with full adders, starting from the least significant position (
- subscript 0), to form thesum bit and carry bit. The input carry C_0 in the least significant position must be 0.
- The value of C_{i+1} in a given significant position is the output carry of the full adder. This value is transferred into the input carry of the full adder that adds the bits one higher significant position to the left.
- The sum bits are thus generated starting from the rightmost position and are available as soon as the corresponding previous carry bit is generated. All the carries must be generated for the correct sum bits to appear at the outputs.



Four Bit Binary Adder

HALF SUBTRACTOR:-

- This circuit needs two binary inputs and two binary outputs.
- Symbols x and y are assigned to the two inputs and D (for difference) and B (for borrow) to the outputs.
- The truth table for the half subtractor is listed in the below table.



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

Truth Table

- The B output is 1 only when the inputs are 0 and 1. The D output represents the least significant bit ofthe subtraction.
- The subtraction operation is done by using the following rules as

0-0=0;

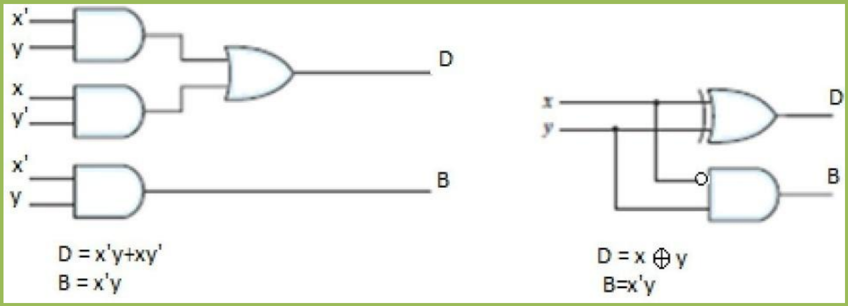
0-1=1 with borrow 1;

1-0=1;

1-1=0.

- The simplified Boolean functions for the two outputs can be obtained directly from the truth table. Thesimplified sum-of-products expressions are

D = x'y + xy' and B =x'y

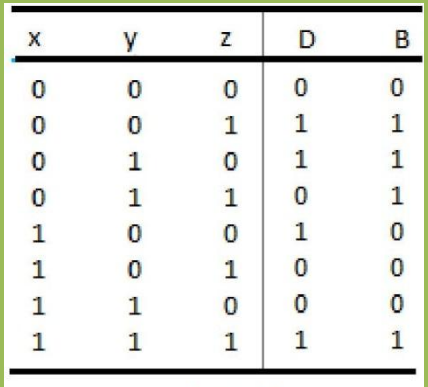


The logic diagram of the half adder implemented in sum of products is shown in

the figure can bealso implemented with an exclusive-OR and an AND gate with one inverted input.

FULL SUBTRACTOR:-

- A full subtractor is a combinational circuit that forms the arithmetic subtraction operation of three bits.
- It consists of three inputs and two outputs. Two of the input variables, denoted by x and y , represent the two significant bits to be subtracted. The third input, z , is



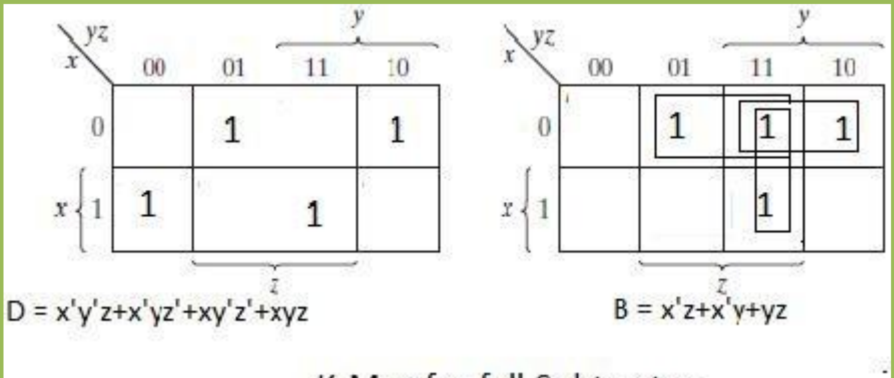
x	y	z	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Truth Table

subtracted from the result 0f the first subtraction.

- Two outputs are necessary because the arithmetic subtraction of three binary digits ranges in value from 0 to 3, and binary representation of 2 or 3 needs two bits. The two outputs are designated by the symbols D for difference and B for borrow.

- The binary variable D gives the value of the least significant bit of the difference. The binary variable B gives the output borrow formed during the subtraction process.



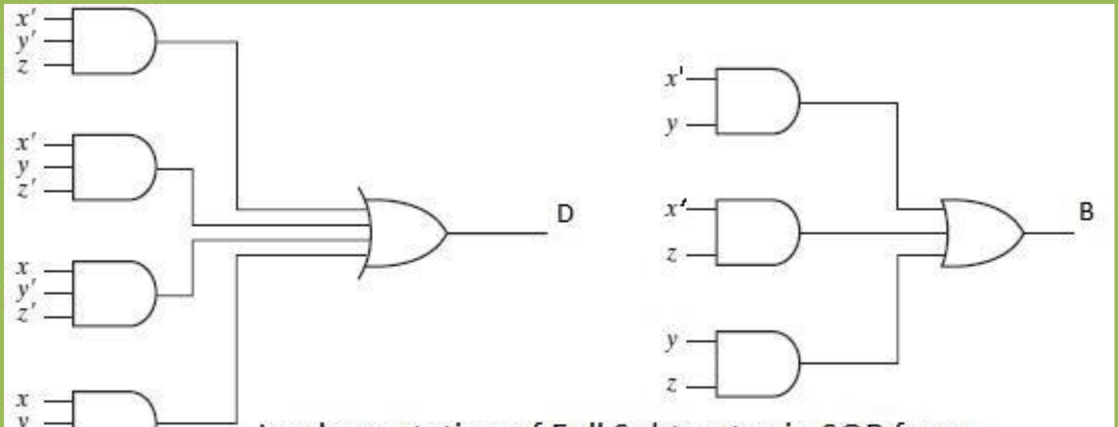
K-Map for full Subtractor

- The eight rows under the input variables designate all possible combinations of the three variables. The output variables are determined from the arithmetic subtraction of the input bits.
- The difference D becomes 1 when any one of the input is 1or all three inputs are equal to1 and theborrow B is 1 when the input combination is (0 0 1) or (0 1 0) or (0 1 1) or (1 1 1).

- The simplified expressions are

D = x'y'z + x'yz' + xy'z' + xyz B = x'z + x'y + yz

- The logic diagram for the full adder implemented in sum-of-products form is shown in figure.



Implementation of Full Subtractor in SOP form

MULTIPLEXER:-

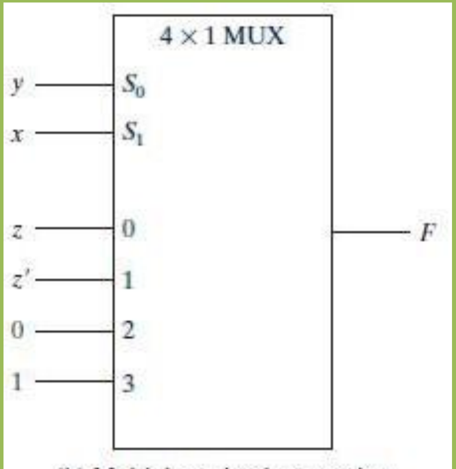
- A multiplexer is a combinational circuit that selects binary information from one of many input lines anddirects it to a single output line.
- The selection of a particular input line is controlled by a set of selection lines.
- Normally, there are 2^n input lines and n selection lines whose bit combinations determine which input isselected.
- A four-to-one-line multiplexer is shown in the below figure. Each of the four inputs, I_0 through I_3, isapplied to one input of an AND gate.
- Selection lines S_1 and S_0 are decoded to select a particular AND gate. The outputs of the AND gatesare applied to a single OR gate that provides the one-line output.

The function table lists the input that is passed to the output for each combination of

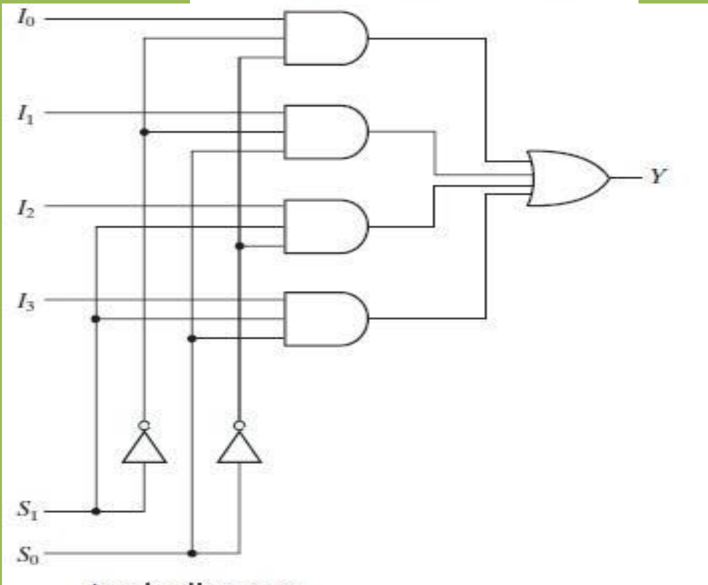
the binary selectionvalues.

- To demonstrate the operation of the circuit, consider the case when S_1S_0= 10.

- The AND gate associated with input I_2 has two of its inputs equal to 1 and the third input connected to I_2.
- The other three AND gates have at least one input equal to 0, which makes their outputs equal to 0. The output of the OR gate is now equal to the value of I_2, providing a path from the selected input to
  the output.
- A multiplexer is also called a data selector, since it selects one of many inputs and steers the binary information to the output line.



(b) Multiplexer implementation



S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Truth table

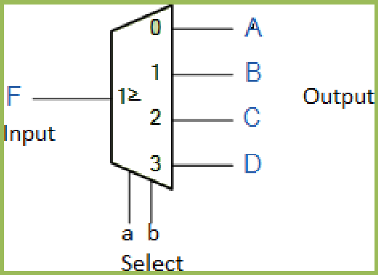
Logic diagram

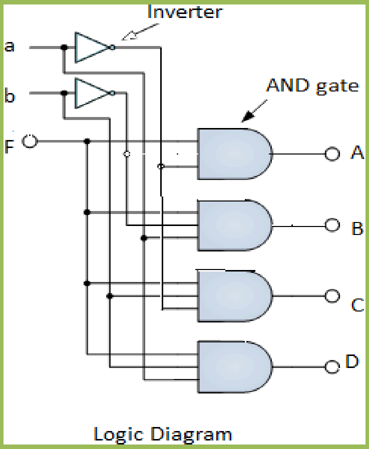
DEMULTIPLEXER:-

- The data distributor, known more commonly as a Demultiplexer or “Demux” for short, is the exactopposite of the Multiplexer.
- The demultiplexer takes one single input data line and then switches it to any one of a number of
  individual output lines one at a time. The demultiplexer converts a serial data signal at the input to aparallel data at its output lines as shown below.
- The Boolean expression for this 1-to-4 demultiplexer above with outputs A to D and data select lines a,b is given as:

F = (ab)’A + a’bB + ab’C + abD

- The function of the demultiplexer is to switch one common data input line to any one of the 4 output data lines A to D in our example above. As with the multiplexer the individual solid state switches are selected by the binary input address code on the output select pins “a” and “b” as shown.



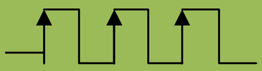


- Unlike multiplexers which convert data from a single data line to multiple lines and demultiplexers which convert multiple lines to a single data line, there are devices available which convert data to and from multiple lines and in the next tutorial about combinational logic devices.

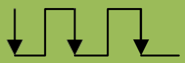
- Standard demultiplexer IC packages available are the TTL 74LS138 1 to 8-output demultiplexer, the TTL 74LS139 Dual 1-to-4 output demultiplexer or the CMOS CD4514 1-to-16 output demultiplexer.

FLIP-FLOP AND LATCH:-

- A flip-flop or latch is a circuit that has two stable states and can be used to store information.
- A flip-flop is a binary storage device capable of storing one bit of information. In a stable state, theoutput of a flip-flop is either 0 or 1.
- Latch is a non-clocked flip-flop and it is the building block for the flip-flop.
- A storage element in digital circuit can maintain a binary state indefinitely until directed by an inputsignal to switch state.
- Storage element that operate with signal level are called latches and those operate with clock transitionare called as flip-flops.
- The circuit can be made to change state by signals applied to one or more control inputs and will haveone or two outputs.
- A flip-flop is called so because its output either flips or flops meaning to switch back and forth.
- A flip-flop is also called a bi-stable multi-vibrator as it has two stable states. The input signals which command the flip-flop to change state are called excitations.
- Flip-flops are storage devices and can store 1 or 0.
- Flip-flops using the clock signal are called clocked flip-flops. Control signals are effective only if they areapplied in synchronization with the clock signal.
- Clock-signals may be positive-edge triggered or negative-edge triggered.
- Positive-edge triggered flip-flops are those in which state transitions take place only at positive- goingedge of the clock pulse.



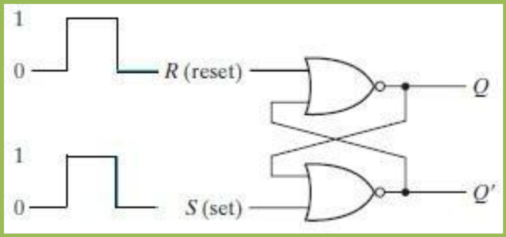
- Negative-edge triggered flip-flops are those in which state transition take place only at negative- goingedge of the clock pulse.



- Some common type of flip-flops include
  a) SR (set-reset) F-F
  b) D (data or delay) F-F
  c) T (toggle) F-F and
  d) JK F-F

SR latch:-

- The SR latch is a circuit with two cross-coupled NOR gates or two cross-coupled NAND gates.
- It has two outputs labeled Q and Q'. Two inputs are there labeled S for set and R foe reset.
- The latch has two useful states. When Q=0 and Q'=1 the condition is called reset state and when Q=1 and Q'=0 the condition is called set state.
- Normally Q and Q' are complement of each other.
- The figure represents a SR latch with two cross-coupled NOR gates. The circuit has NOR gates and as we know if any one of the input for a NOR gate is HIGH then its output will be LOW and if both the inputs are LOW then only the output will be HIGH.

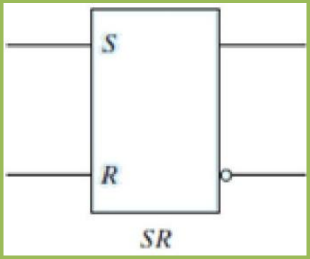


- Under normal conditions, both inputs of the latch remain at 0 unless the state has to be changed. The application of a momentary 1 to the S input causes the latch to go to the set state. The S input must go back to 0 before any other changes take place, in order to avoid the occurrence of an undefined next state that results from the forbidden input condition.
- The first condition (S = 1, R = 0) is the action that must be taken by input S to bring the circuit to the set
  state. Removing the active input from S leaves the circuit in the same state. After both inputs return to 0, it is then possible to shift to the reset state by momentary applying a 1 to the R input. The 1 can then be removed from R, whereupon the circuit remains in the reset state. When both inputs S and R are equal to 0, the latch can be in either the set or the reset state, depending on which input was most recently a 1.
- If a 1 is applied to both the S and R inputs of the latch, both outputs go to 0. This action produces an undefined next state, because the state that results from the input transitions depends on the order in which they return to 0. It also violates

the requirement that outputs be the complement of each other. In normal operation, this condition is avoided by making sure that 1's are not applied to both inputs simultaneously.

- Truth table for SR latch designed with NOR gates is shown below.

Inp ut		Outp ut				Comment
S	R	Q	Q'	QN ext	Q'N ext	
0	0	0	1	0	1	No change
0	0	1	0	1	0	
0	1	0	1	0	1	Reset
0	1	1	0	0	1	
1	0	0	1	1	0	Se t
1	0	1	0	1	0	
1	1	0	1	X	X	Prohibit ed state
1	1	1	0	X	X	



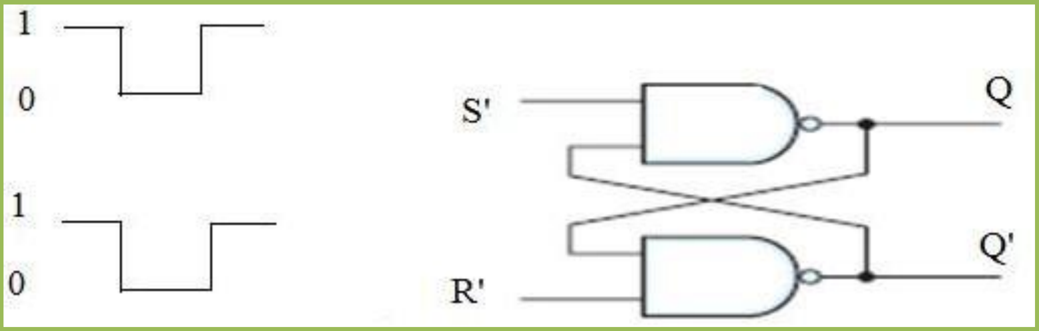
Symbol for SR NOR Latch

<u>Racing Condition</u>:-

In case of a SR latch when S=R=1 input is given both the output will try to become 0. This is called Racing condition.

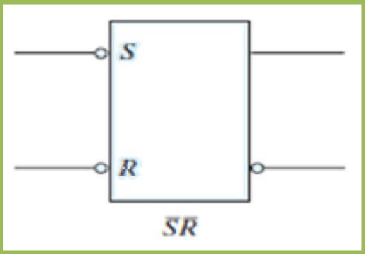
<u>SR latch using NAND gate:-</u>

- The below figure represents a SR latch with two cross-coupled NAND gates. The circuit has NAND gates and as we know if any one of the input for a NAND gate is LOW then its output will be HIGH and if both the inputs are HIGH then only the output will be LOW.
- It operates with both inputs normally at 1, unless the state of the latch has to be changed. The application of 0 to the S input causes output Q to go to 1, putting the latch in the set state.
-
-
- When the S input goes back to 1, the circuit remains in the set state. After both inputs go back to 1, we are allowed to change the state of the latch by placing a 0 in the R input. This action causes the circuit to go to the reset state and stay there even after both inputs return to 1.



- The condition that is forbidden for the NAND latch is both inputs being equal to 0 at the same time, an input combination that should be avoided.

- n comparing the NAND with the NOR latch, note that the input signals for the NAND require the complement of those values used for the NOR latch. Because the NAND latch requires a 0 signal to change its state, it is sometimes referred to as an S'R' latch. The primes (or, sometimes, bars over the letters) designate the fact that the inputs must be in their complement form to activate the circuit.

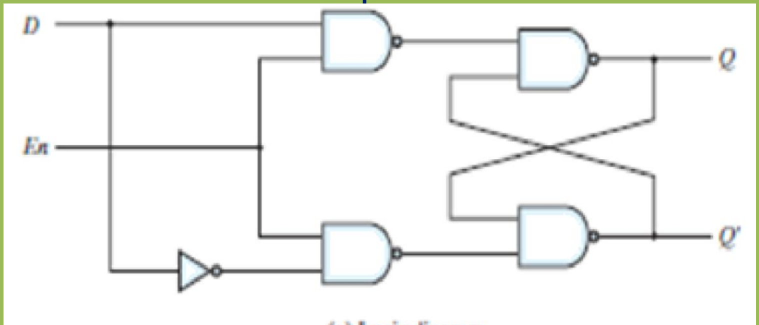


The above represents the symbol for inverted SR latch or SR latch using NAND gate. Truth table for SR latch using NAND gate or Inverted SR latch

S	R	Qnext	Q'next
0	0	Race	Race
0	1	0	1 (Reset)
1	0	1	0 (Set)
1	1	Q (No change)	Q' (No change)

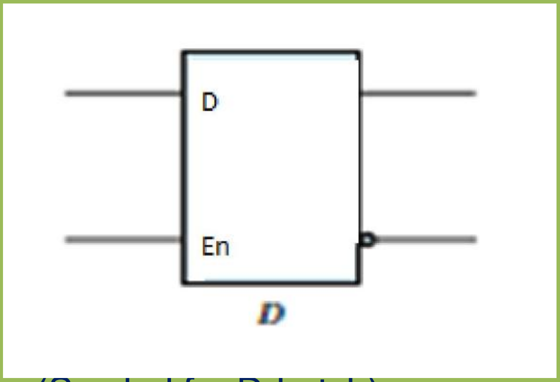
<u>D LATCH:-</u>

- One way to eliminate the undesirable condition of the indeterminate state in the SR latch is to ensurethat inputs S and R are never equal to 1 at the same time.



(a) Logic diagram

- This is done in the D latch. This latch has only two inputs: D (data) and En (enable).
- The D input goes directly to the S input, and its complement is applied to the R input.



(Symbol for D-Latch)

- As long as the enable input is at 0, the cross-coupled SR latch has both inputs at the 1 level and thecircuit can't change state regardless of the value of D.

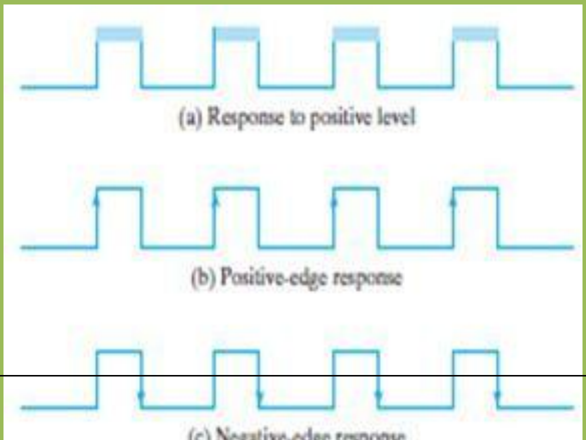
- The below represents the truth table for the D-latch.

En	D	Next State of Q
0	X	No change
1	0	Q=0;Reset State
1	1	Q=1;Set State

- The D input is sampled when En = 1. If D = 1, the Q output goes to 1, placing the circuit in the set state. If D = 0, output Q goes to 0, placing the circuit in the reset state. This situation provides a path from input D to the output, and for this reason, the circuit is often called a TRANSPARENT latch.

<u>TRIGGERING METHODS:-</u>

- The state of a latch or flip-flop is switched by a change in the control input. This momentary change is called a trigger, and the transition it causes is said to trigger the flip-flop.
- Flip-flop circuits are constructed in such a way as to make them operate properly when they are part of a sequential circuit that employs a common clock.
- The problem with the latch is that it responds to a change in the level of a clock pulse. For proper operation of a flip-flop it should be triggered only during a signal transition.
- This can be accomplished by eliminating the feedback path that is inherent in the operation of the sequential circuit using latches. A clock pulse goes through two transitions: from 0 to 1 and the return
  from 1 to 0.
- A ways that a latch can be modified to form a flip-flop is to produce a flip-flop that triggers only during a signal transition (from 0 to 1 or from 1 to 0) of the synchronizing signal (clock) and is disabled during the rest of the clock pulse.

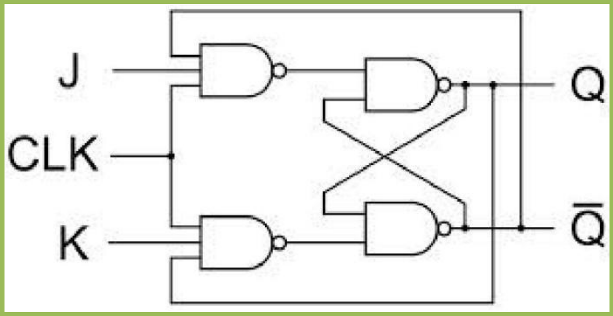

(a) Response to positive level
(b) Positive-edge response
(c) Negative-edge response

JK FLIP-FLOP:-

- The JK flip-flop can be constructed by using basic SR latch and a clock. In this case the outputs Q and Q' are returned back and connected to the inputs of NAND gates.

- This simple JK flip Flop is the most widely used of all the flip-flop designs and is considered to be auniversal flip-flop circuit.
- The sequential operation of the JK flip flop is exactly the same as for the previous SR flip-flop with thesame "Set" and "Reset" inputs.
- The difference this time is that the "JK flip flop" has no invalid or forbidden input states of the SR Latcheven when S and R are both at logic "1".

(The below diagram shows the circuit diagram of a JK flip-flop)



- The JK flip flop is basically a gated SR Flip-flop with the addition of a clock input circuitry that prevents the illegal or invalid output condition that can occur when both inputs S and R are equal to logic level "1".

The symbol for a JK flip flop is similar to that of an SR bistable latch except the

- Both the S and the R inputs of the SR bi-stable have now been replaced by two inputs called the J and K inputs, respectively after its inventor Jack and Kilby. Then this equates to: J = S and K = R.
- The two 2-input NAND gates of the gated SR bi-stable have now been replaced by two 3-input NAND gates with the third input of each gate connected to the outputs at Q and Q'.
- This cross coupling of the SR flip-flop allows the previously invalid condition of S = "1" and R = "1" state to be used to produce a "toggle action" as the two inputs are now interlocked.
- If the circuit is now "SET" the J input is inhibited by the "0" status of Q' through the lower NAND gate. If the circuit is "RESET" the K input is inhibited by the "0" status of Q through the upper NAND gate. As Q
  and Q' are always different we can use them to control the input.

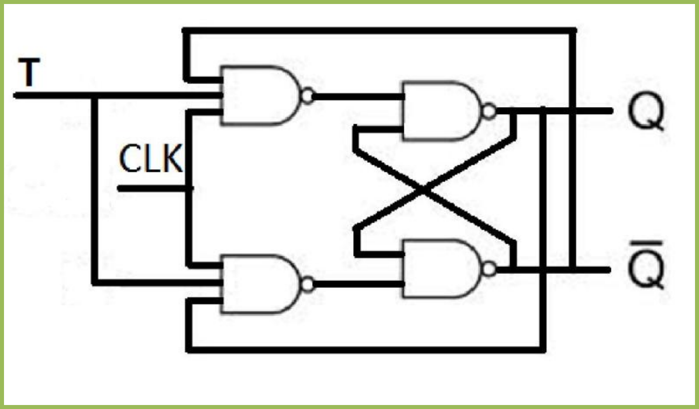
(Truth table for JK flip-flop)

Input		Output		Comment
J	K	Q	Qnext	
0	0	0	0	No change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Toggle
1	1	1	0	

- When both inputs J and K are equal to logic "1", the JK flip flop toggles.

T FLIP-FLOP:-

- Toggle flip-flop or commonly known as T flip-flop.
- This flip-flop has the similar operation as that of the JK flip-flop with both the inputs J and K are shorted
  i.e. both are given the common input.



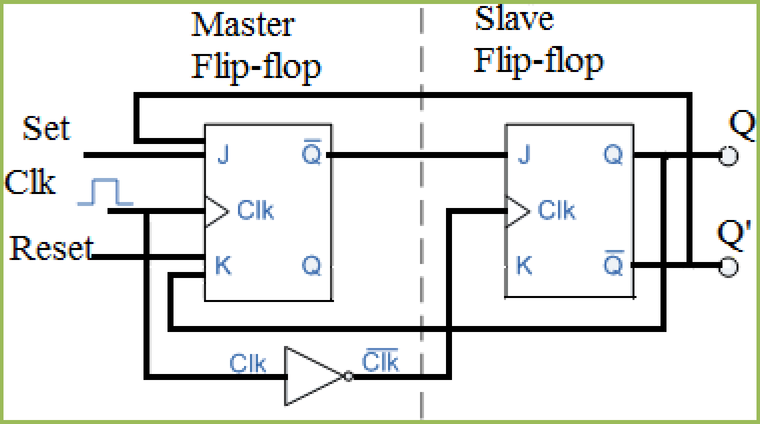
- Hence its truth table is same as that of JK flip-flop when J=K= 0 and J=K=1.So its truth table is asfollows.

T	Q	Qn ext	Comment
0	0	0	No change
	1	1	
1	0	1	Toggles
	1	0	

MASTER-SLAVE JK FLIP-FLOP:-

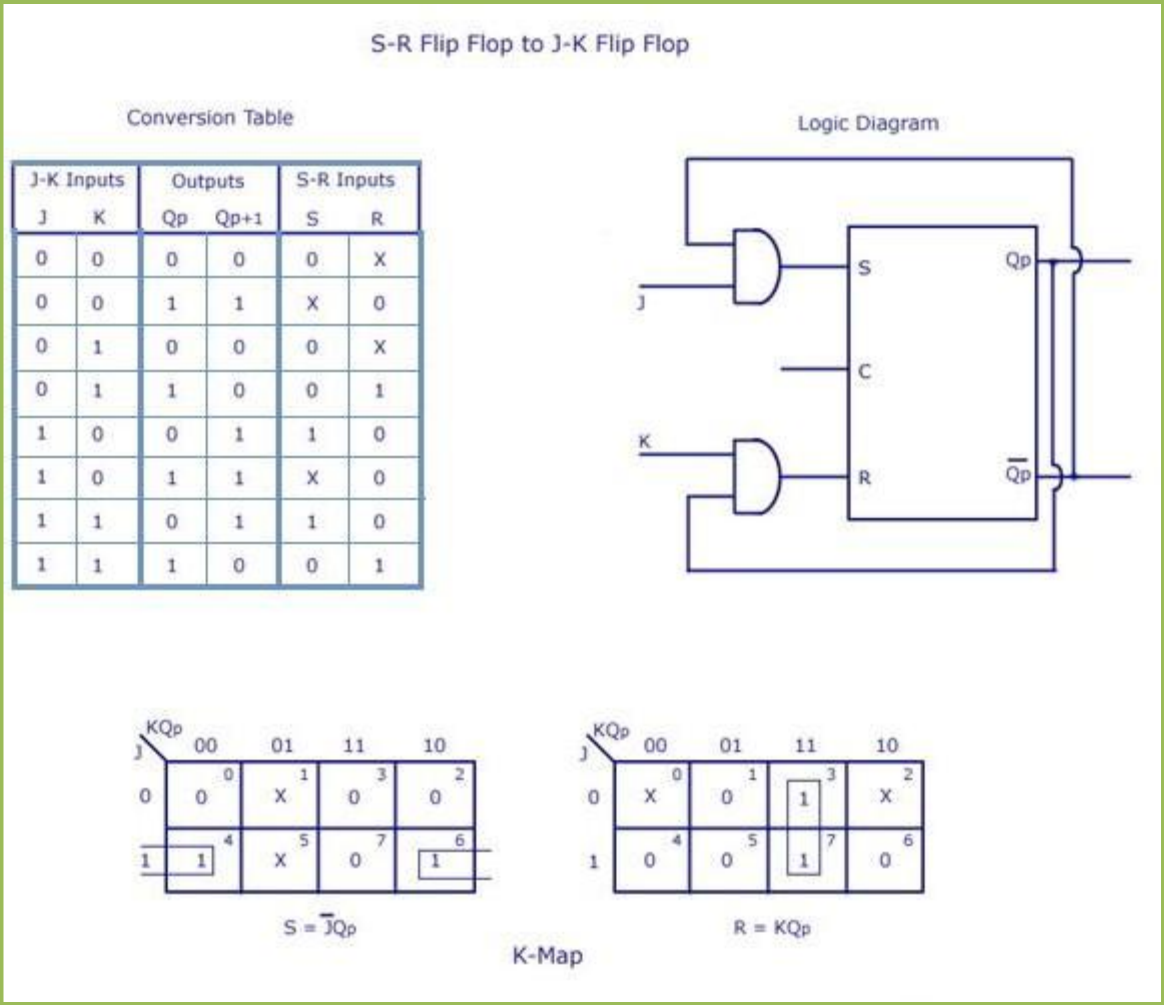
- The Master-Slave Flip-Flop is basically two gated SR flip-flops connected together in a seriesconfiguration with the slave having an inverted clock pulse.
- The outputs from Q and Q' from the "Slave" flip-flop are fed back to the inputs of the "Master" with theoutputs of the "Master" flip flop being connected to the two inputs of the "Slave" flip flop.
- This feedback configuration from the slave's output to the master's input gives the characteristic toggleof the JK flip flop as shown below.

The Master-Slave JK Flip Flop



- The input signals J and K are connected to the gated “master” SR flip flop which “locks” the input condition while the clock (Clk) input is “HIGH” at logic level “1”.
- As the clock input of the “slave” flip flop is the inverse (complement) of the “master” clock input, the“slave” SR flip flop does not toggle.
- The outputs from the “master” flip flop are only “seen” by the gated “slave” flip flop when the clock inputgoes “LOW” to logic level “0”.
- When the clock is “LOW”, the outputs from the “master” flip flop are latched and any additionalchanges to its inputs are ignored.
- The gated “slave” flip flop now responds to the state of its inputs passed over by the “master” section.
- Then on the “Low-to-High” transition of the clock pulse the inputs of the “master” flip flop are fed through to the gated inputs of the “slave” flip flop and on the “High-to-Low” transition the same inputs are reflected on the output of the “slave” making this type of flip flop edge or pulse-triggered.

JK Flip Flop to SR Flip Flop



J-K Inputs		Outputs		S-R Inputs	
J	K	Qp	Qp+1	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

J \ KQp	00	01	11	10
0	0	X	0	0
1	1	X	0	1

$S = J\overline{Q}p$

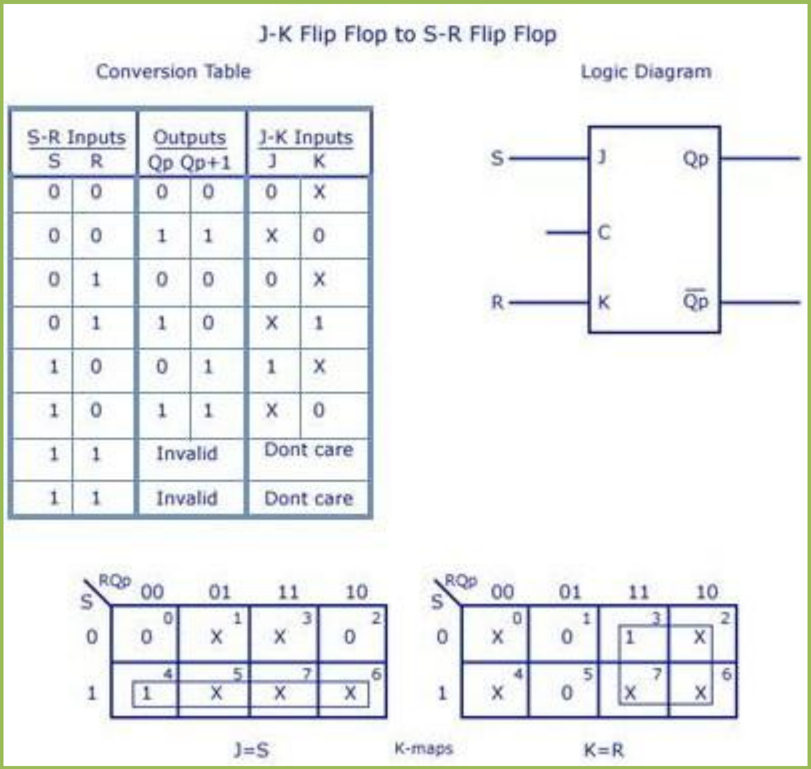
J \ KQp	00	01	11	10
0	X	0	1	X
1	0	0	1	0

$R = KQp$

- This will be the reverse process of the above explained conversion. S and R will be the external inputs

- to J and K. J and K will be the outputs of the combinational circuit. Thus, the values of J and K have to be obtained in terms of S, R and Qp.
- A conversion table is to be written using S, R, Qp, Qp+1, J and K.
- For two inputs, S and R, eight combinations are made. For each combination, the corresponding Qp+1 outputs are found out.
- The outputs for the combinations of S=1 and R=1 are not permitted for an SR flip flop. Thus the outputs are considered invalid and the J and K values are taken as "don't cares".

T



S-R Inputs		Outputs		J-K Inputs	
S	R	Qp	Qp+1	J	K
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	X	1
1	0	0	1	1	X
1	0	1	1	X	0
1	1	Invalid		Dont care	
1	1	Invalid		Dont care	

S \ RQp	00	01	11	10
0	0	X	X	0
1	1	X	X	X

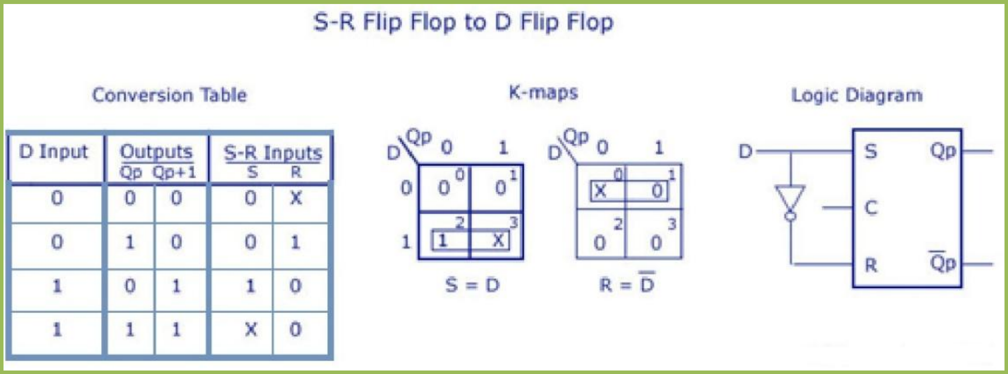
J=S

S \ RQp	00	01	11	10
0	X	0	1	X
1	X	0	X	X

K=R

<u>Flip Flop to D Flip Flop</u>

- S and R are the actual inputs of the flip flop and D is the external input of the flip flop.
- The four combinations, the logic diagram, conversion table, and the K-map for S and R in terms of Dand Qp are shown below.



S-R Flip Flop to D Flip Flop

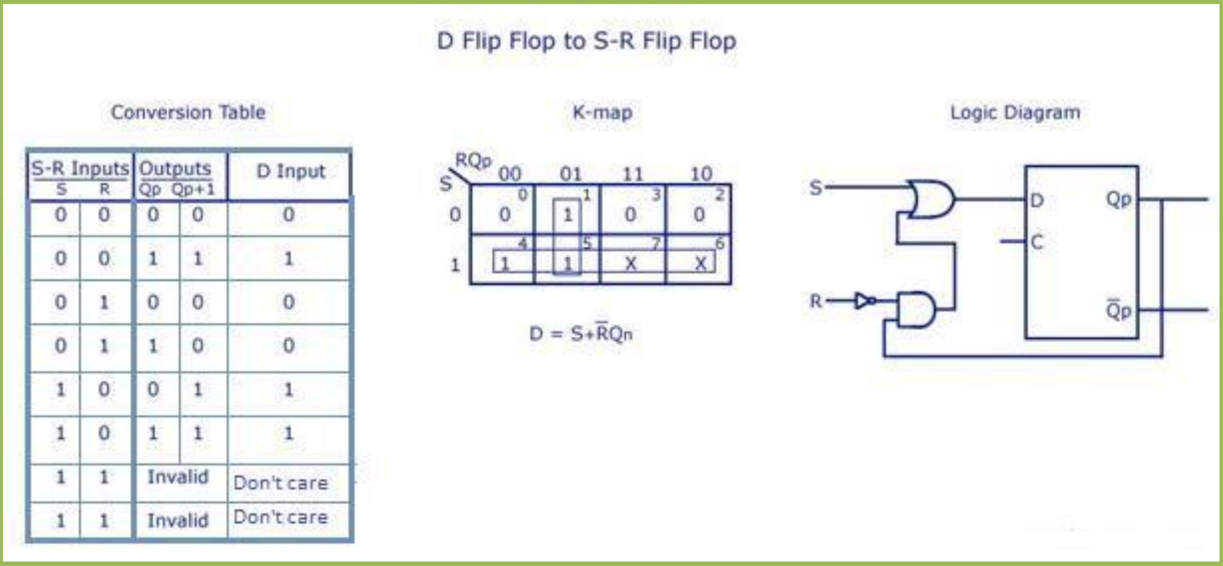
Conversion Table

D Input	Outputs		S-R Inputs	
	Qp	Qp+1	S	R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0

R

D Flip Flop to SR Flip Flop

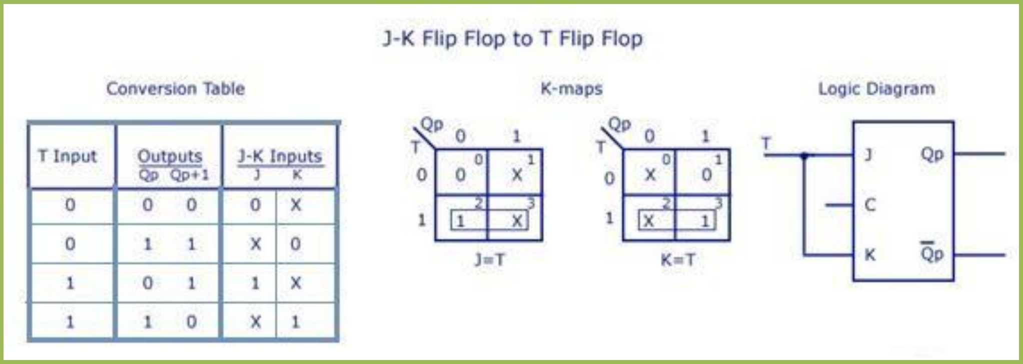
- D is the actual input of the flip flop and S and R are the external inputs. Eight possible combinations are achieved from the external inputs S, R and Qp.
- But, since the combination of S=1 and R=1 are invalid, the values of Qp+1 and D are considered as“don’t cares”.
- The logic diagram showing the conversion from D to SR, and the K-map for D in terms of S, R and Qpare shown below.



S-R Inputs		Outputs		D Input
S	R	Qp	Qp+1	
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	Invalid		Don't care
1	1	Invalid		Don't care

JK Flip Flop to T Flip Flop:-

- J and K are the actual inputs of the flip flop and T is taken as the external input for conversion
- Four combinations are produced with T and Qp. J and K are expressed in terms of T and Qp.
  - The conversion table, K-maps, and the logic diagram are given below.

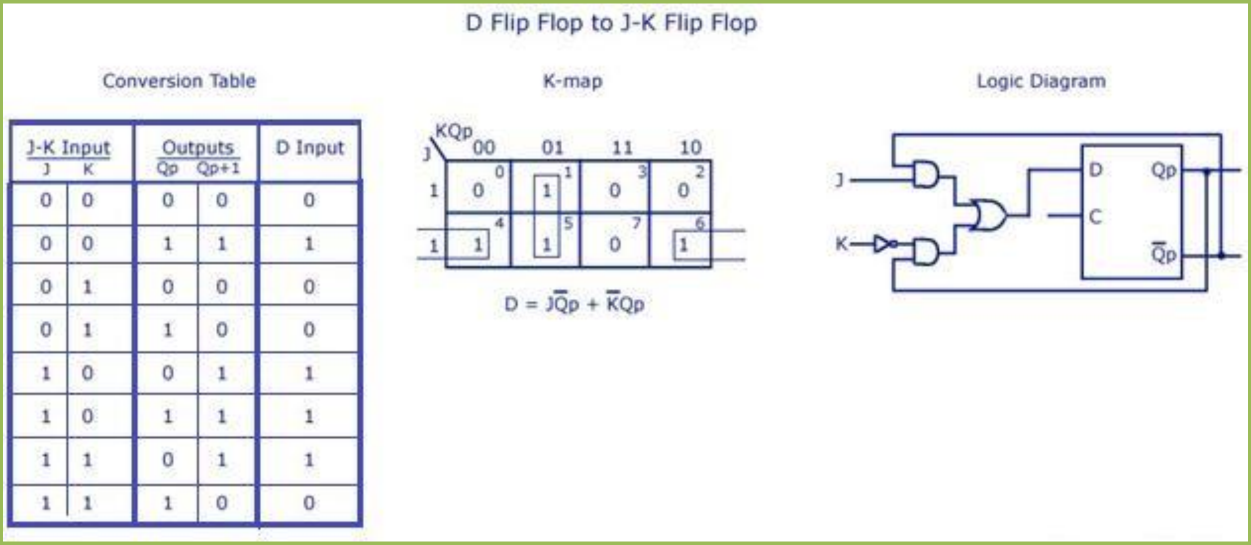

J-K Flip Flop to T Flip Flop

Conversion Table

T Input	Outputs Qp	Outputs Qp+1	J-K Inputs J	J-K Inputs K
0	0	0	0	X
0	1	1	X	0
1	0	1	1	X
1	1	0	X	1

D Flip Flop to JK Flip Flop:-

- In this conversion, D is the actual input to the flip flop and J and K are the external inputs.
- J, K and Qp make eight possible combinations, as shown in the conversion table below. D isexpressed in terms of J, K and Qp.
- The conversion table, the K-map for D in terms of J, K and Qp and the logic diagram showing theconversion from D to JK are given in the figure below.


D Flip Flop to J-K Flip Flop

Conversion Table

J-K Input J	J-K Input K	Outputs Qp	Outputs Qp+1	D Input
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

ONE MARK QUESTIONS

1. What is a Circuit?

a) Open-loop through which electrons can pass
b) Closed-loop through which electrons can pass
c) Closed-loop through which Neutrons can pass
d) None of the mentioned

2. Which of the following is a type of digital logic circuit?

a) Combinational logic circuits

b)Sequential logic circuits

c) Both Combinational & Sequential logic circuits

d) None of the mentioned

3. Which gates in Digital Circuits are required to convert a NOR-based SR latch to anSR flip-flop?
a) Two 2 input AND gates
b) Two 3 input AND gates
c) Two 2 input OR gates
d) Two 3 input OR gates

4. Which number system has a base 16

a)Hexadecimal

b)Octal

c)Binary

d)Decimal

5. What is a digital-to-analog converter?

a)It stores digital data on the computer.

b)It converts alternating current (AC) into direct current (DC).

c)It converts electrical power into mechanical power.

d)It takes the digital data from an audio CD and converts it to a useful form

6.How many bits are needed to store one BCD digit?

a)2 bits

b)4bits

c)3 bits

d)1 bit

7. Which of these sets of logic gates are known as universal gates?

a)XOR, NAND, OR

b)OR, NOT, XOR

c)NOR, NAND, XNOR

d)NOR, NAND

8. What is the binary subtraction of 101001 - 010110 =?

a)010011

b)100110

c)011001

d)010010

9. A classification of integrated circuits with complexities of 30 to 300 equivalent gates on a single chip is known as?

a)VLSI

b)SSI

c)LSI

d)MSI

1's complement of 1011001 is

a)0100111

b)0101100

c)0100110

d)0110110

11. An overflow is a

a)User input problem

b)Hardware problem

c)Software problem

d)Input-Output problem

15. The excess-3 code for 584 is given by

a)100010110111

b)100001110111

c)100010010110

d)100001010110

16. In Digital electronics (Boolean algebra), the OR operation is performed by which of the given properties

a)Distributive properties

b)Commutative properties

c)Associative properties

d)All of these

17. DeMorgan's Law states that

a)(A+B)' = A'*B

b)(AB)' = A' + B'

c)(AB)' = A' + B

d)(AB)' = A + B

K-map (Karnaugh map) is an abstract form of which diagram organized as a squares matrix.

a)Block diagram

b)Cycle diagram

c)Square diagram

d)Venn diagram

19. The AND operation is equivalent to

a)Union

b)Intersection

c)Division

d)Both option a and b

20. Super Computer can perform -----------------of instructions Per Second .

[**Ans: Hundreds of millions]**

21. How many binary bits are necessary to represent 748?

a) 9 b) 7 **c)10** d) 11

22. For subtraction of binary number, subtract the ------------------ .

a) Minuend from the subtrahend digit

b) Subtrahend digit from the Minuend

d) MSB from the LSB

d) None of the above

23. A Karnaugh map (K – map) is an abstract form of ---------- diagram, organized as a matrix of squares

a) Venn diagram b) Cycle Diagramc) Block Diagram d) Triangular

24.For every x, y in B

a)x + y = y + x , x. y = y. x

a)Commutative Law b) Complement c) Closure d) None of the above

25. How many unique symbols are used in the decimal number system?

a) One b) Nine **c) Ten** d) Unlimited

26. Which technology used in the evaluation of aptitude tests?

a) OCR **b) OMR** c) MICR d) MCR

27. **The main building blocks of combinational circuits are:**

a) Flip-flops
b) Counters
c) Logic gates
d) Memory units

28. **Which logic gate is used to perform the Boolean OR operation?**

a) AND gate
b) NOT gate

c) NOR gate
d) OR gate

29. Which of these number systems has a base of 16?

a) Decimal

b) Binary

c)Hexadecimal

d) Octal

30. The radix of an octal number system is:

a) 8 b) 2 c) 16 d) 10

Three Mark Questions:

1.List the number systems?

i) Decimal Number system

ii) Binary Number system

iii) Octal Number system

iv) Hexadecimal Number system

2. Define binary logic?

Binary logic consists of binary variables and logical operations. The variables vare designated by the alphabets such as A, B, C, x, y, z, etc., with each variable having only two distinct values: 1 and 0. There are three basic logic operations: AND, OR, and NOT.

3. What is a Logic gate?

Logic gates are the basic elements that make up a digital system. The electronic gate is a circuit that is able to operate on a number of binary inputs in order to perform a particular logical function.

4. What are the basic digital logic gates?

The three basic logic gates are:

1. AND gate

2. OR gate

3. NOT gat

5. State the distributive property of Boolean algebra.

The distributive property states that AND ing several variables and OR ing the result with a single variable is equivalent to OR ing the single variable with each of the several variables and then AND ing the sums. The distributive property is:

i). $A+BC = (A+B)(A+C)$ ii). $A(B+C) = AB + AC$

6.State the limitations of karnaugh map.

i) Generally it is limited to six variable map (i.e.) more then six variable involving expressions are not reduced.

ii) The map method is restricted in its capability since they are useful for simplifying only Boolean expression represented in standard form

7.Define Duality Theorem.

The Duality theorem states that starting with a Boolean relation we can derive another Boolean relation by:

i). Changing OR (operation) i.e., + (Plus) sign to an AND (operation) i.e., (dot) and Vice-versa.

ii). Complement any 0 or 1 appearing in the expression i.e., replacing contains 0 and 1 by 1 and 0 respectively.

8. Simplify the following expression Y = (A + B)(A + C')(B' + C')

Y = (A + B)(A + C')(B' + C')

= (AA' + AC +A'B +BC)(B' + C') [A.A' = 0]

= (AC + A'B + BC)(B' + C')

= AB'C + ACC' + A'BB' + A'BC' + BB'C + BCC‘

9. Show that (X + Y' + XY)(X + Y')(X'Y) = 0

(X + Y' + XY)(X + Y')(X'Y) = (X + Y' + X)(X + Y')(X' + Y) [A + A'B = A + B]

= (X + Y')(X + Y')(X'Y) [A + A = 1]

= (X + Y')(X'Y) [A.A = 1]

= X.X' + Y'.X'.Y

= 0 [A.A' = 0]

10. Convert the given expression in canonical SOP form Y = AC + AB + BC

Y = AC + AB + BC

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

=ABC + ABC' + AB'C + AB'C' + ABC + ABC' + ABC

=ABC + ABC' +AB'C + AB'C' [A + A =1]

11. What is a diode?

Diodes are used for permitting the current flow in a particular direction. They are made using semiconductor substances. The main five types of diodes are Zener Diode, Small Signal Diode, Light Emitting Diode, Small Signal Diode and Schottky Diode.

12. What are the main advantages of digital systems over analogue systems?

The main advantages of digital systems over analogue systems are:

The data transmission in digital systems occurs without signal degradation due to noise. Digital systems possess noise immunity, which enables efficient data processing. On the other hand, analogue systems are susceptible to wear and tear, which deteriorates the stored information.

Most digital systems have computer interfaces that allow for much easier control of the signals and information. Updating the system software will enable us to resolve any bugs in the data or signal. Such features are not present in the analogue systems.

13. **What are the main disadvantages of digital systems?**

Even though digital systems are considered to be noise-resistant and storage efficient, they also have a few disadvantages.

The energy consumption of digital systems is larger compared to analogue systems. This energy is used to process signals and calculations.

Digital communication systems require larger bandwidth than analogue systems. Digital systems are expensive in terms of components and maintenance.

Digital systems are vulnerable to errors if there is even slight misinterpretation in input data.

- **Practice Questions**
- 1) What is the difference between AND and NAND gates?
- 2) What is the difference between analogue and digital signals?
- 3) Which is the first digital computer?
- 4) What is a boolean function?
- 5) Which type of signal is vulnerable to noise