PNS SCHOOL OF ENGINEERING & TECHNOLOGY

Nishamani Vihar, Marshaghai, Kendrapara

LECTURE NOTES
ON
DIGITAL ELECTRONICS

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

3RD SEMESTER

PREPARED BY

MR. ADITYA NARAYAN JENA

LECTURER IN ELECTRONICS & TELECOMMUNICATION

BASIC OF DIGITAL ELECTRONICS

INTRODUCTION:-

- 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.

1.1 NUMBER SYSTEM:-

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

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

This will be interpreted as

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

where Y = value of the entire number

a_n = the value of the n^{th}

digitr = radix

TYPES OF NUMBER SYSTEM:-

There are four types of number systems. They are

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

➢ 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.

➢ 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.

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.

➢ 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.

Examples:

(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:

$(111.101)_2 = (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3})$
$= 4 + 2 + 1 + 0.5 + 0 + 0.125$
$= (7.625)_{10}$

(b) Binary to Octal conversion:-

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

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

Examples:

(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 $(5726.663)_8$

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

Example:

(i) Convert $(1011011011)_2$ into hexadecimal.

Solution:

Given Binary number	10	1101	1011
Group of 4 bits are	0010	1101	1011
Convert each group into hex =	2	D	B

The result is **$(2DB)_{16}$**

(ii) 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.

Example:

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

Solution:

Divide the given decimal number successively by 2 read the integer part remainder upwards to getequivalent binary number. Multiply the fraction part by 2. Keep the integer in the product as it is and multiplythe 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

Answer of $(52)_{10}$ is $(110100)_2$.

(ii) Convert $(105.15)_{10}$ into binary.

Solution:

```
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) Decimal to octal conversion:-

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.

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.

Example:

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

hexadecimal. Solution:

```
16 |                          0.675 x 16 = 10.8    A
2598
16 | 162  — 6    6            0.800 x 16 = 12.8    C
16 | 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.

Example:

Convert $(367.52)_8$ into binary.

Solution:

```
Given Octal number is                      6    7.    5    2

Convert each group octalto binary       110  111 .  101  010
```

Result of $(367.52)_8$ is $(011110111.101010)_2$

(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:

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

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 binarynumber 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 is 3 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 $(0011\ 1010\ 1001\ 1110.1011\ 0000\ 1101)_2$

(b) Hexadecimal to decimal conversion:-

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 $(41209.0572)_{10}$

(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	=	101 110 110 111 . 101 011 100
Convert 3 bits group to octal.	=	5 6 6 7 . 5 3 4

Result is $(5637.534)_8$

ARITHEMATIC OPERATION

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 :-

$$\begin{array}{r} 100101 \\ + \ \underline{1101111} \\ \underline{10010100} \end{array}$$

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 :-

$$\begin{array}{r} 1010.010 \\ - \ \underline{111.111} \\ \underline{0010.011} \end{array}$$

Result is $(0010.011)_2$

3. BINARY MULTIPLICATION:-

The binary multiplication rules are as follows0 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. BINARY DIVISION:-

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 $(111.1)_2$

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

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

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.

GRAY CODE:-

The gray code is a non-weighted code. It is not a BCD code. It is cyclic code because successive words in thisdiffer 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

followsG$_{\oplus}$ = B_n
Gn-1 = Bn Bn-1

.

.

.

.

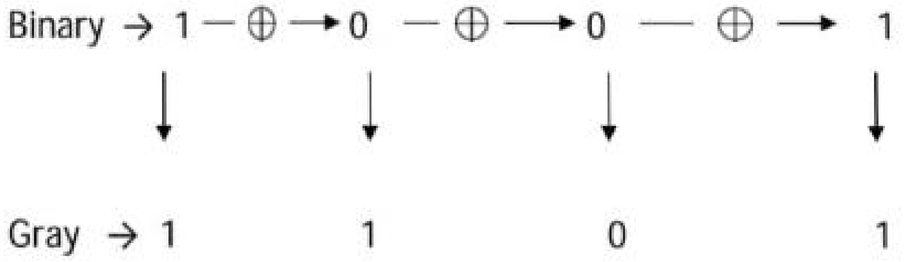
$G_1 = B_2{}^{\oplus} B_1$

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

For example

Convert the binary 1001 to the Gray code.

Solution :-`



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} = Bn G_{n-1}

.

.

.

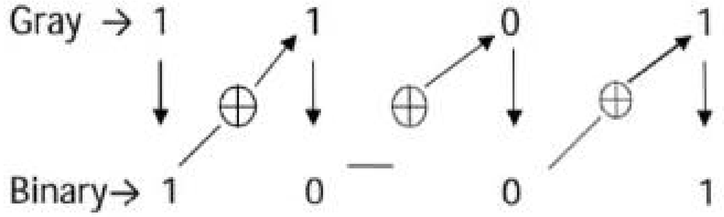
.

$B_1 = B_2 \oplus G$

For example :-

Convert the Gray code 1101 to the binary.

Solution :-



The binary code is 1001

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'scomplement 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 $(0110)_2$

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.

For example:-

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

Solution:-

```
 11010              11010                       =  26
- 1000     =>     + 01111  (1's complement)     = - 16
     0
          Carry →  101001                         + 10
                  +     1
                   01010   = +10
```

Result is +10

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:-

```
  1010100                1010100                        =  84
- 1010100       =>     + 0101100 (2's complement)       - 84
                         10000000 ( Ignore the carry)      0
                     =
                         0 (result = 0)
```

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

LOGIC GATES

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 andcomponents.
- Inputs and outputs of logic gates can occur only in 2 levels. These two levels are termed HIGH andLOW, 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 iscalled 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 thelower 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 thehigher of the two voltages level represents the logic 0.

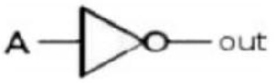
DIFFERENT TYPES OF LOGIC GATES:-

NOT GATE (INVERTER):-

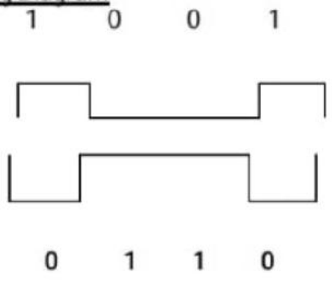
- A NOT gate, also called and inverter, has only one input and one output.
- It is a device whose output is always the complement of its input.
- The output of a NOT gate is the logic 1 state when its input is in logic 0 state and the logic 0 state whenits inputs is in logic 1 state.

IC No. :- 7404

Logic Symbol



Timing Diagram



Truth table

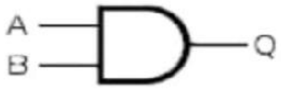
INPUT	OUTPUT
A	$\overline{A}$
0	1
1	0

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.

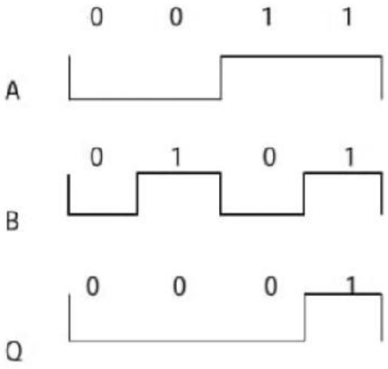
IC No.:- 7408

Logic Symbol



Truth Table

Timing Diagram

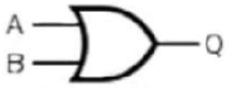


		OUTPUT
A	B	Q=A . B
0	0	0
0	1	0
1	0	0
1	1	1

OR GATE:-

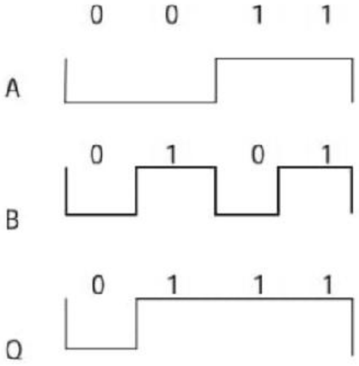
- An OR gate may have two or more inputs but only one output.
- The output is logic 1 state, even if one of its input is in logic 1 state.
- The output is logic 0 state, only when each one of its inputs is in logic state.

IC No.:- 7432



Logic Symbol
Truth Table

Timing Diagram



Truth table:

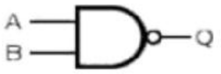
INPUT		OUTPUT
A	B	Q=A + B
0	0	0
0	1	1
1	0	1
1	1	1

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 gate7420 four
input NAND gate 7430
eight input NAND gate

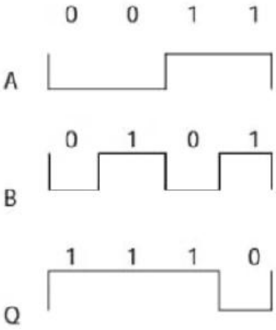
Logic Symbol



Truth Table

INPUT		OUTPUT
A	B	Q= ——
0	0	1
0	1	1
1	0	1
1	1	0

Timing Diagram



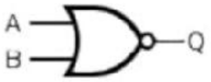
NOR GATE:-

- 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.

IC No.:- 7402 two input NOR
gate 7427 three input

NOR gate7425 four
input NOR gate

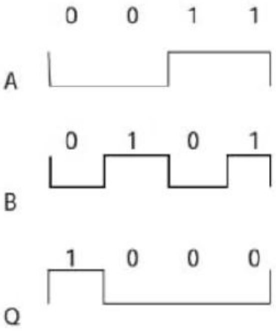
Logic Symbol



Truth Table

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

Timing Diagram

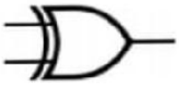


EXCLUSIVE – OR (X-OR) GATE:-

- 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



Truth Table

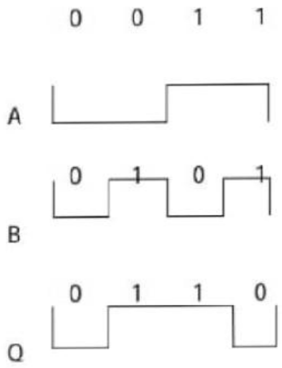
INPUT		OUTPUT
A	B	Q = A⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

INPUTS are A and B

OUTPUT is $Q = A \oplus B$

$= A' B + A B'$

Timing Diagram

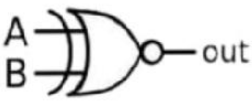


EXCLUSIVE – NOR (X-NOR) GATE:-

- 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

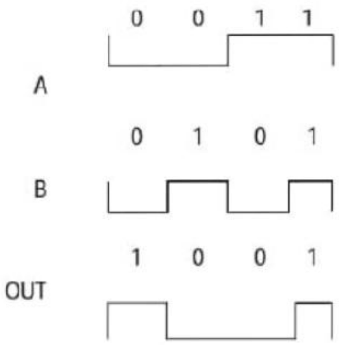


INPUT		OUTPUT
A	B	OUT =A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

OUT =A B + A'B'

= A XNOR B

Timing Diagram

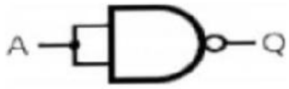


UNIVERSAL GATES:-

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:-

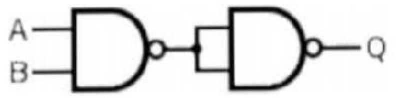
a) Inverter from NAND gate



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

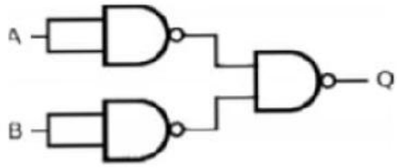
b) AND gate from NAND gate

Input s are A and BOutput Q = A.B



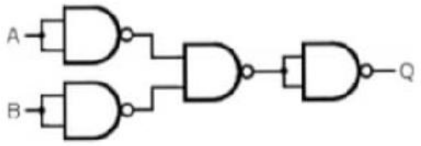
c) OR gate from NAND gate

Inputs are A and BOutput Q = A+B



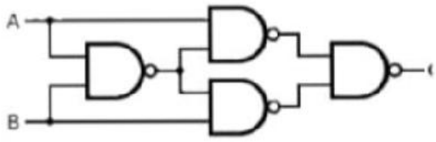
d) NOR gate from NAND gate

Inputs are A and BOutput Q = A+B



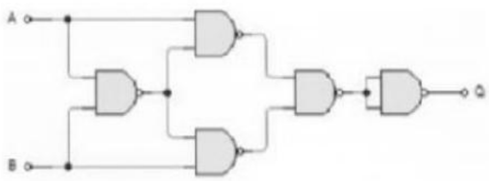
e) EX-OR gate from NAND gate

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



f) EX-NOR gate From NAND gate

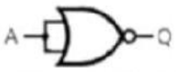
Inputs are A and B
Output Q = A B + A′ B′



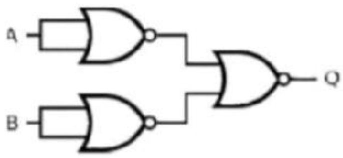
NOR GATE:-

a) Inverter from NOR gate

Input = A
Output Q = A′

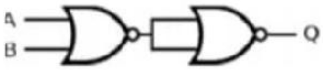


b) AND gate from NOR gateInput s are A and B Output Q = A.B



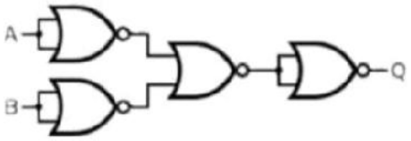
a) OR gate from NOR gate

Inputs are A and BOutput Q = A+B



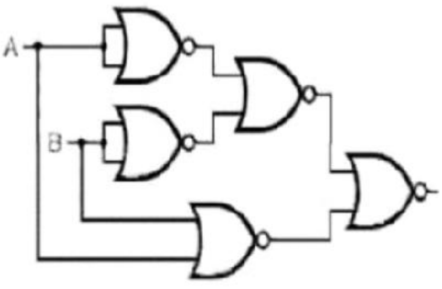
b) NAND gate from NOR gate

Inputs are A and BOutput Q = A.B



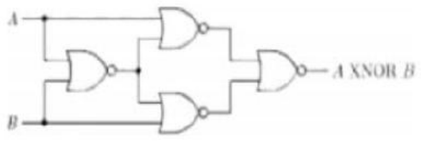
c) EX-OR gate from NOR gate

Inputs are A and_B Output Q = A′B + AB′



d) EX-NOR gate From NOR gate

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



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 ofelements (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 opcration
Axiom 1: 0 . 0 = 0
Axiom 2: 0 . 1 = 0
Axiom 3: 1 . 0 = 0
Axiom 2: 1 . 1 = 1

OR opcration
Axiom 5: 0 + 0
= 0
Axiom 6: 0 + 1 = 1
Axiom 7: 1 + 0 = 1
Axiom 8: 1 + 1 = 1

NOT opcration
Axiom 9: $\bar{1} = 0$
Axiom 10:0 = 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: $\underline{0} = 1$
Law 2: 1 = 0
Law 3: if A = 0, then A = ~~1~~
Law 4: if $\underline{A}$ = 1,thenA = 0
Law 5: A = 0 (double complementation law)

2. OR Laws:-

The four OR laws are as
follows Law 1: A + 0 =
0

Law 2: A + 1 = 1
Law 3: A + A = A Law
4: A +A = 1

3. AND Laws:-

The four AND laws are as
follows Law 1: A . 0 = 0
Law 2: A . 1 = 1
Law 3: A . A = A
Law 4: A .$\underline{A}$ = 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

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

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)

A	B	C	A+B	(A+B)+C
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

=

A	B	C	B+C	A+(B+C)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Law 2: (A .B) C = A (B .C)

A	B	C	AB	(AB)C
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

=

A	B	C	B.C	A(B.C)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

This law can be extended to any number of variables. For

Example
A(BCD) = (ABC)D = (AB) (CD)

6. Distributive Laws:-

The distributive laws allow factoring or multiplying out of expressions. There are two distributive laws.

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

A	B	C	B+C	A(B+C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

=

A	B	C	AB	AC	A+(B+C)
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

Law 2: A + BC = (A+B)

(A+C)Proof RHS = (A+B) (A+C)

= AA + AC + BA + BC

= A + AC + AB + BC

= A (1+ C + B) + BC

= A. 1 + BC (1 +C + B = 1 + B = 1)

= A + BC

= LHS

7. Redundant Literal Rule

(RLR):-Law 1: A + AB = A

+ B

$$\mathbf{A + \bar{A}B = (A + \bar{A})\,(A + B)}$$
$$\mathbf{= 1.\,(A + B)}$$
$$\mathbf{= A + B}$$

Law 2: $A\,(\bar{A} + B) = AB$

$= AA' + AB$

$= 0 + AB$

$= AB$

8. Idempotence Laws:- Idempotence means same value.

Law 1: A. A = A

If A = 0, then A. A = 0. 0 =0

= A If A = 1, then A. A = 1. 1

= 1 = A

This law states that AND of a variable with itself is equal to that variable only.

Law 2: A + A = A

If A = 0, then A + A = 0 + 0 = 0

= AIf A = 1, then A + A = 1 + 1

= 1 = A

This law states that OR of a variable with itself is equal to that variable only.

9. Absorption Laws:-

There are two laws:

Law 1: $A + A \cdot B = A$

$A + A \cdot B = A\,(1 + B) = A \cdot 1 = A$

A	B	AB	A+AB
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

2: $A\,(A + B) = A$

$A\,(A + B) = A \cdot A + A \cdot B = A + AB = A(1 + B) = A \cdot 1 = A$

Law

A	B	A+B	A(A+B)
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

10. De Morgan's Theorem:-

Law 1: (A + B)′ =A′· B′

A	B	A + B	$\overline{A + B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

=

A	B	$\overline{A}$	$\overline{B}$	$\overline{A}\ \overline{B}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

This law states that the complement of a sum of variables is equal to the product of their individualcomp~~le~~ments.

Law 2: (A· B)′ = A ′+ B′

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

=

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

This law states that the complement of a product of variables is equal to the sum of their individualcomplements.

Types of canonical expression:

1. Sum of Product (SOP)
2. Product of Sum (POS)

SUM - OF - PRODUCTS FORM:-

- This is also called disjunctive Canonical Form (DCF) or Expanded Sum of Products Form or CanonicalSum of Products Form.
- In this form, the function is the sum of a number of products terms where each product term contains allvariables 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 tothose combinations for which 'f ' assumes the value 1.

  For example

  $f(A, B, C) = AB + BC$

  $= AB\,(C + C') + BC\,(A + A')$

  $= A\,BC + ABC' + ABC + A'BC$
- The product term which contains all the variables of the functions either in complemented oruncomplemented 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 forwhich 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 min terms.

PRODUCT- OF - SUMS FORM:-

- This form is also called as Conjunctive Canonical Form (CCF) or Expanded Product - of – Sums Formor 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 $f(A, B, C) = (A + \underline{B} + C\cdot C') + (\underline{A} + \underline{B} + C\cdot C')$
-

  $= (A + B + C)\,(A + B + C')\,(A + B + C)\,(A + B + C')$
- The sum term which contains each of the 'n' variables in either complemented or uncomplemented formis called a maxterm.
- Maxterm is represented as $M_0, M_1, M_2,$ ....

  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.

1.SUM OF PRODUCT (SOP) FORM or MINTERM:

- The SOP expression usually takes the form of two or more variable ANDed together ORed with two or more other variable ANDed together.

Example→AB′+AC+A′BC
→AB+CD

STANDARD SOP FORM:

- A standard SOP expression is one in which all the variables is present in each product term in the expression.

- **Example →AB′CD+A′B′CD′+ABC′D′**

2.PRODUCT OF SUM (POS)FORM or MAXTERM:

- The POS expression generally takes the form of two or more ORed .Variable with in parentheses ANDed with two or more other variable within parentheses.

Example→(A+B).(C+D)
→(X+Y′)(Y+Z)
→(Y+Z′+X′)(XY+Z)

STANDARD POS FORM:

- A standard POS expression is one in which all the variables is present in each sum term in the expression.

- **Example→**(A′+B′+C′+D′)(A+B′+C+D)(A+B+C′+D)

GENERAL SOP TO STANDARD SOP:

Example 1:
AB+BC

Solution-AB.1+BC.1
=AB(C+C′)+BC(A+A′)
=ABC+ABC′+ABC+A′BC
=ABC+ABC′+A′BC **[ABC+ABC=ABC Using OR rules A+A=A]**

Example 2:
ABC′+AB+C
Solution:
=ABC′+AB(C+C′)+C(A+A′)(B+B′)
=ABC′+ABC+ABC′+C(AB+AB′+A′B+A′B′)
=ABC′+ABC+ABC′+ABC+AB′C+A′BC+A′B′C
=ABC+ABC′+AB′C+A′BC+A′B′C

GENERAL POS TO STANDARD POS:-

Example 1:
(A+B+C)(A+B)
=(A+B+C)(A+B+CC′)
=(A+B+C).(A+B+C)(A+B+C′)
=(A+B+C).(A+B+C′)

Example 2:
(A′+B+C).A
=(A′+B+C)(A+BB′+CC′)
=(A′+B+C)(A+B+C)(A+B+C′)(A+B′+C)(A+B′+C′)

RULES FOR STANDARD SOP TO STANDARD POS:

- Consider each variables as 1.
- Write the possible combinations.
- Write the left combinations.
- In the left combinations consider each variables as zero &write the sum terms.
- The product of the sum terms is the standard POS.

Example 1→ABC+A′BC+A′B′C+A′B′C′

- Consider each variables as 1 i.e. A=B=C=1
- Possible combinations i.e.111+011+001+000
- Left combinations are

010→A+B′+C
100 →A′+B+C
101→A′+B+C′
110→A′+B′+C
POS →(A+B′+C)(A′+B+C)(A′+B+C′)(A′+B′+C)→Standard POS

Example 2→A′B′C′+A′B′C+A′BC+ABC′+AB′C′

- Consider each variables as 1 i.e. A=B=C=1
- Possible combinations i.e.000+001+011+110+100
- Left combinations are

010→A+B′+C
101→A′+B+C′
111→A′+B′+C′
POS→ (A+B′+C) (A′+B+C′) (A′+B′+C′)→Standard POS

RULES FOR STANDARD POS TO STANDARD SOP:-

- Consider each variables as 0.
- Write the possible combinations.
- Write the left combination.
- In the left combinations consider each variable as 1and write the product terms.
- The sum of the product term is the standard SOP.

Example 1→ (A′+B+C)(A+B+C)(A+B+C′)(A+B′+C)(A+B′+C′)

- Consider each variable as 0 i.e.=B=C=0
- Possible combinations i.e. (1+0+0)(0+0+0)(0+0+1)(0+1+0)(0+1+1)
- Left combinations are

1 0 1→AB′C
1 1 0→ABC′
1 1 1→ABC
SOP→(AB′C)+(ABC′)+(ABC)→Standards SOP

Example 2→ (A+B+C′)(A′+B′+C′)(A′+B′+C):

- Consider each variable as 0 i.e. A=B=C=0
- Possible combinations i.e.(0+0+1)(1+1+1)(1+1+0)
- Left combinations are

0 0 0→A′B′C′
0 1 0→A′BC′
0 1 1→A′BC
1 0 0→AB′C′
1 0 1→AB′C
SOP=(A′B′C)+(A′BC′)+(A′BC)+(AB′C′)+(AB′C)→Standards SOP.

Q.1. Find the canonical SOP (minterm) for the following function.
Y (A.B) =A+B

Solution:
Y (A.B) =A+B
=A.1+B.1
=A (B+B′) +B (A+A′)

=AB+AB′+AB+A′B
=AB+AB′+A′B

Q.2. Y=A+B′C express the function in canonical SOP & canonical POS.

Solution:

Y=A+B′C

=A(B+B′)(C+C′)+B′C(A+A′)

=A(BC+BC′+B′C+B′C′)+AB′C+A′B′C

=ABC+ABC′+AB′C+AB′C′+AB′C′+A′B′C

=ABC+ABC′+AB′C+AB′C′+A′B′C

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

Therefore Y=∑m (1, 4, 5, 6, 7)

- canonical pos (maxterm)

Y=A+B′C

=(A+B′)(A+C)

=(A+B′+CC′)(A+C+BB′)

=(A+B′+C)(A+B′+C′)(A+B+C)(A+B′+C)

=(A+B+C)(A+B′+C)(A+B′+C′) **[(A+B′+C)+ (A+B′+C)= (A+B′+C)]**

$Y=m_0m_2m_3$

Therefore Y=∏m (0, 2, 3)

3. **Expand the function A+BC′+ABD′+ABCD to minterm & maxterm.**

Solution:-

Y=A+BC′+ABD′+ABCD

=A(B+B′)(C+C′)(D+D′)+BC(A+A′)(D+D′)+ABD′(C+C′)+ABCD

=A[(BC+BC′+B′C+B′C′)(D+D′)]+BC′(AD+AD′+A′D+A′D′)+ABCD′+ABC′D′+ABCD

=A[BCD+BC′D+B′CD+B′C′D+BCD′+BC′D′+B′CD′+B′C′D′]+ABC′D+ABC′D′+A′BC′D+A′B C′D′+ABCD′+ABC′D′+ABCD

=ABCD+ABC′D+AB′CD+AB′C′D+ABCD′+ABC′D′+AB′CD′+AB′C′D′+A′BC′D+A′BC′D′

SOP=∑m(4,5,8,9,10,11,12,13,14,15)

POS=∏m(0,1,2,3,6,7)

Q.4.Expand A(B′+A)B to maxterm & minterm

Solution:

In SOP (maxterm):

Y=A(B′+A)B

=(A+0)(B′+A)(B+0)

=(A+BB′)(B′+A)(B+AA′)

=(A+B)(A+B′)(A+B′)(A+B)(A′+B)

=(A+B)(A+B′)(A′+B′)

Y=∏m(0,1,2) ➔**maxterm(POS)**

- The maxterm **m3** is missing in the POS form, so the SOP form will contain only the minterm **m3.**

In SOP (minterm):

Solution:

Y=A(B′+A)B

=ABB′+AAB

=0+AB **[AND rules BB′=0 & AA=A]**

=AB

Y=∑m(3) ➔**minterm(SOP)**

Q.5. Expand the following expression to min terms & max terms.

x′ + x(x + y′) (y + z′)

Solution:

F= X′ + X (X + Y′)(Y + Z′)

= X′ + (XX +X Y′)(Y + Z′) [DISTRIBUTIVE LAW]

=X′ + (X +X Y′)(Y + Z′) [AND RULES XX=X]

= X′ + X(1 + Y′)(Y + Z′) [OR RULES 1+Y′=1]

=X′+X(Y+Z′)

= X′ + XY + XZ

= X′ (Y+Y′) (Z+Z′) +XY (Z+Z′) +XZ′ (Y+Y′)

=X′ (YZ+YZ′+Y′Z+Y′Z′) +XYZ+XYZ′+XYZ′+XY′Z′

=X′YZ+X′YZ′+X′Y′Z+X′Y′Z′+XYZ+XYZ′+XYZ′+XY′Z′

=X′YZ+X′YZ′+X′Y′Z+X′Y′Z′+XYZ+XYZ′+XY′Z′

Y= ∑M(0,1,2,3,4,6,7)➔MINTERM(SOP)

Y=(5)➔MAXTERM(POS)

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.

		B	
		0	1
A	0	$\bar{A}$ B	$\bar{A}$ B
	1	A $\overline{B}$	A B

Example:-

Map expression f= $\bar{A}$B +

ABSolution:-

The expression

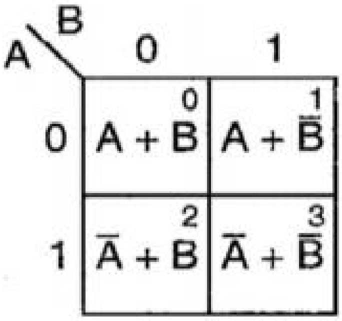
minterms isF = $m_1 + m_2$

= m(1, 2)

		B	
		0	1
A	0	0 0	1 1
	1	2 1	3 0

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.



A \ B	0	1
0	^{0} $A + B$	^{1} $A + \bar{B}$
1	^{2} $\bar{A} + B$	^{3} $\bar{A} + \bar{B}$

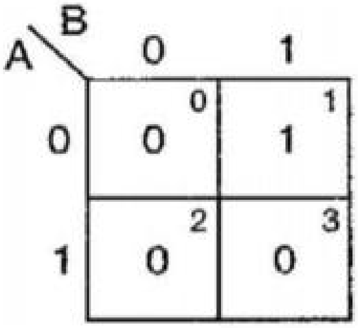
The maxterm of a two variable K-map

Example:-

Plot the expression f= (A + B)(A′ + B)(A′+ B′)

Solution:-

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

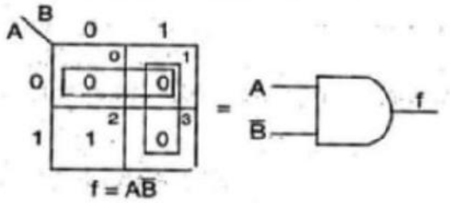


A \ B	0	1
0	0 ^{0}	1 ^{1}
1	0 ^{2}	0 ^{3}

Example:-

Reduce the expression f = (A + B) (A + B')(A' +B ') using mappingSolution:-

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

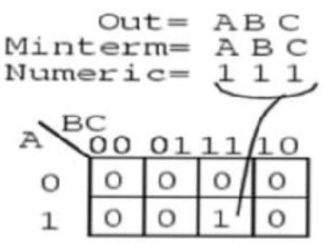


f = $A\overline{B}$

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

<u>Example 2</u>

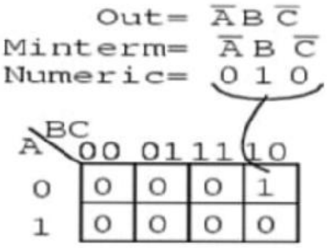
- The number of cells in 3 variable K-map is eight, since the number of variables is three.
- The following figure shows **3 variable K-Map**.
- **Examples 1:**


Out= ABC

Minterm= ABC

Numeric= 1 1 1

A \ BC	00	01	11	10
0	0	0	0	0
1	0	0	1	0

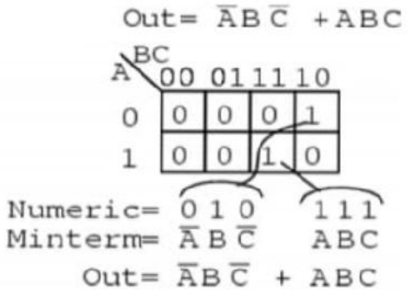

Out= $\overline{A}B\overline{C}$

Minterm= $\overline{A}B\overline{C}$

Numeric= 0 1 0

A \ BC	00	01	11	10
0	0	0	0	1
1	0	0	0	0

- Out= ABC   Out= $\overline{A}B\overline{C}$

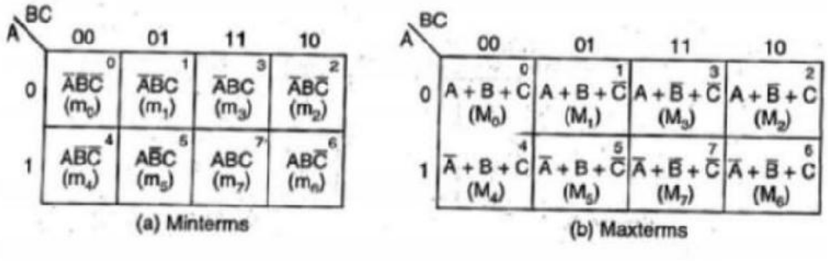

Out= $\overline{A}B\overline{C}$ +ABC

A \ BC	00	01	11	10
0	0	0	0	1
1	0	0	1	0

Numeric= 0 1 0   1 1 1

Minterm= $\overline{A}B\overline{C}$ ABC

Out= $\overline{A}B\overline{C}$ + ABC

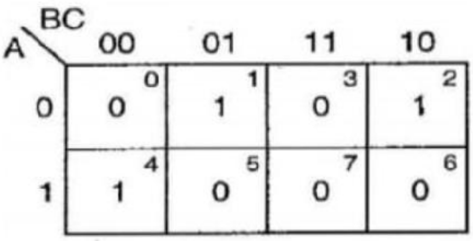


A \ BC	00	01	11	10
0	$\overline{A}\overline{B}\overline{C}$ (m_0)	$\overline{A}\overline{B}C$ (m_1)	$\overline{A}BC$ (m_3)	$\overline{A}B\overline{C}$ (m_2)
1	$A\overline{B}\overline{C}$ (m_4)	$A\overline{B}C$ (m_5)	ABC (m_7)	$AB\overline{C}$ (m_6)

(a) Minterms

A \ BC	00	01	11	10
0	$A+B+C$ (M_0)	$A+B+\overline{C}$ (M_1)	$A+\overline{B}+\overline{C}$ (M_3)	$A+\overline{B}+C$ (M_2)
1	$\overline{A}+B+C$ (M_4)	$\overline{A}+B+\overline{C}$ (M_5)	$\overline{A}+\overline{B}+\overline{C}$ (M_7)	$\overline{A}+\overline{B}+C$ (M_6)

(b) Maxterms

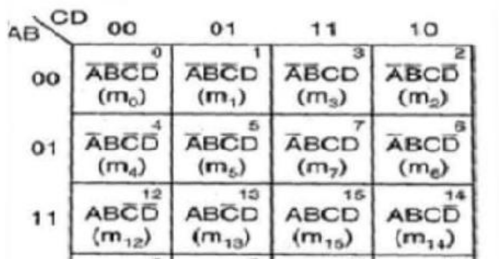


A \ BC	00	01	11	10
0	0	1	0	1
1	1	0	0	0

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.

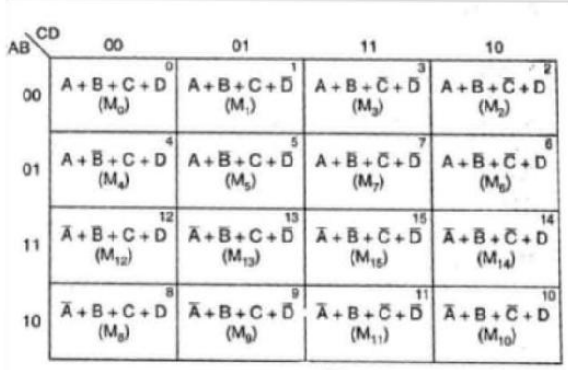
SOP FORM



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

POS 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:-

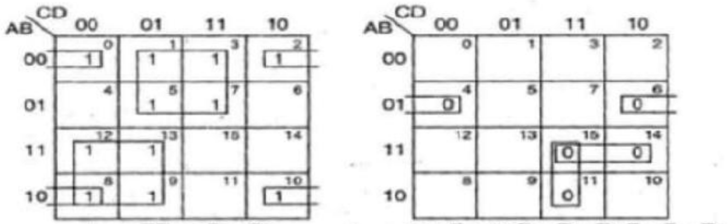
For reducing the Boolean expressions in SOP (POS) form the following steps are given below

- 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). Theyare 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)



$f_{min} = \bar{B}\bar{D} + A\bar{C} + \bar{A}D$

(a) SOP K-map

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

(b) POS K-map

The minimal SOP expression is f_{min}= BD + AC + AD

– – –

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

DON'T CARE COMBINATIONS:-

The combinations for which the values of the expression are not specified are called don't care combinations oroptional 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.

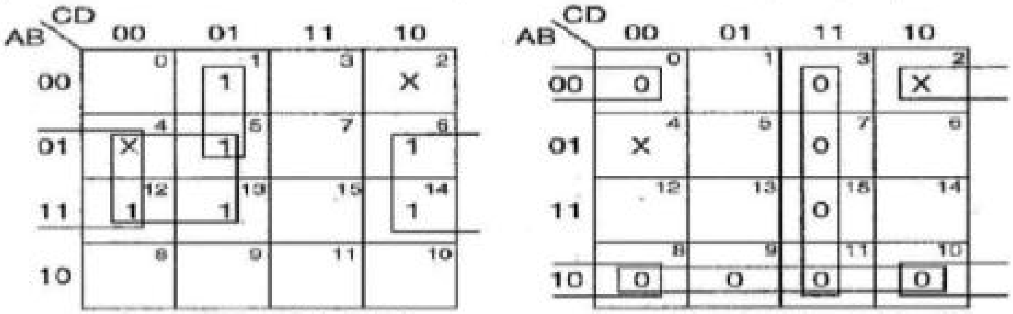
Example:-

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

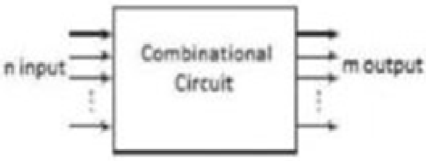
The minimal of SOP expression is f_{min} = BC + BD +ACD

– –

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

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.

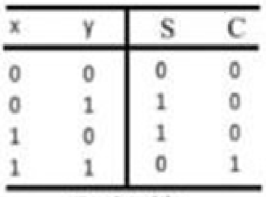


BINARY ADDER–SUBTRACTOR:-

- 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 the truth table.



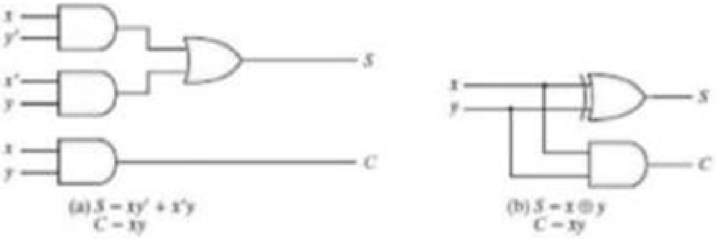
x	y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Truth Table

- The simplified sum-of-products expressions are

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

- 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.

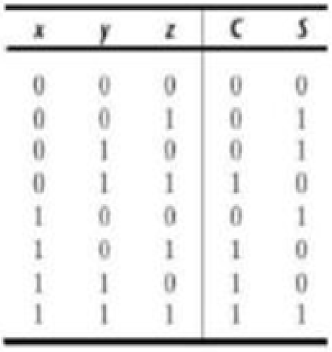


(a) $S = xy' + x'y$
$C = xy$

(b) $S = x \oplus y$
$C = xy$

FULL ADDER:-

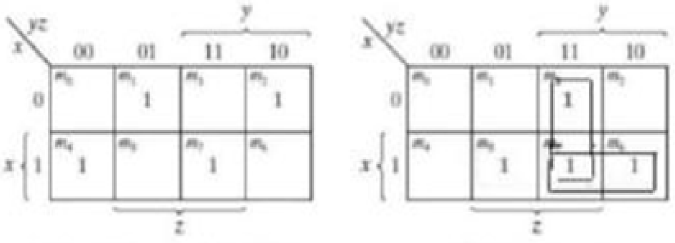
- A full adder is a combinational circuit that forms the arithmetic sum of three bits.
- It consists of three inputs and two outputs. Two of the input variables, denoted by x and y , represent the two significant bits to be added. The third input, z , represents the carry from the previous lower significant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



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

- 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.



x \ yz	00	01	11	10
0	m_0	m_1 1	m_3	m_2 1
1	m_4 1	m_5	m_7 1	m_6

x \ yz	00	01	11	10
0	m_0	m_1	m_3 1	m_2
1	m_4	m_5 1	m_7 1	m_6 1

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

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

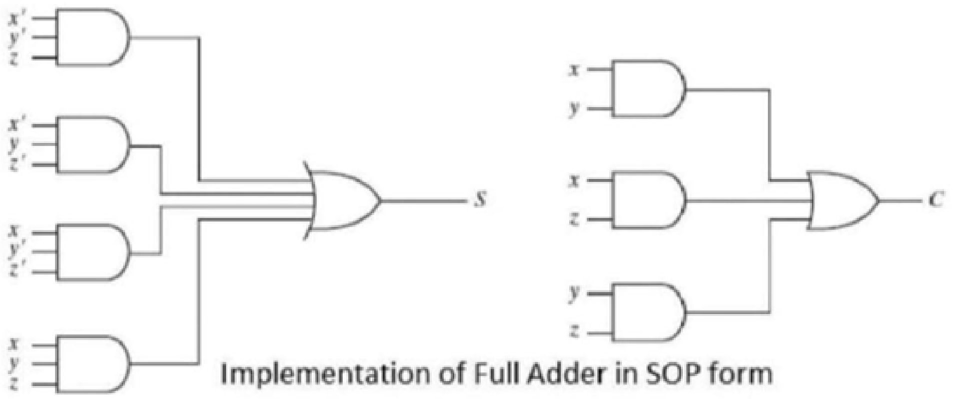
K-Map for full adder

- 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, the output 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. The C 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

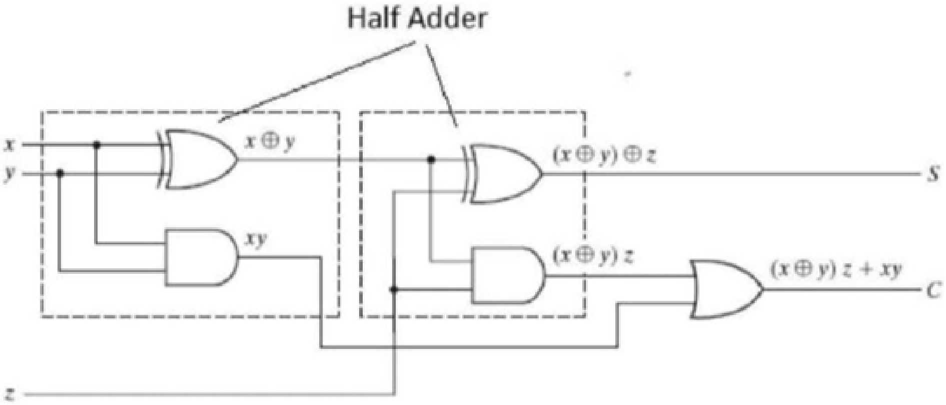
- The logic diagram for the full adder implemented in sum-of-products form is shown in figure.



Implementation of Full Adder in SOP form

Full adder using half adder

- 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

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

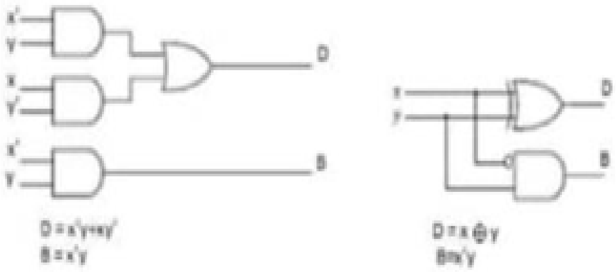
Truth Table

- The B output is 1 only when the inputs are 0 and 1. The D output represents the least significant bit of the 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. The simplified sum-of-products expressions are

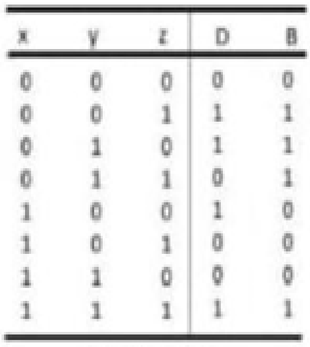
D = x'y + xy' and B =x'y



- The logic diagram of the half adder implemented in sum of products is shown in the figure. It can be also 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 subtracted from the result 0f the first subtraction.

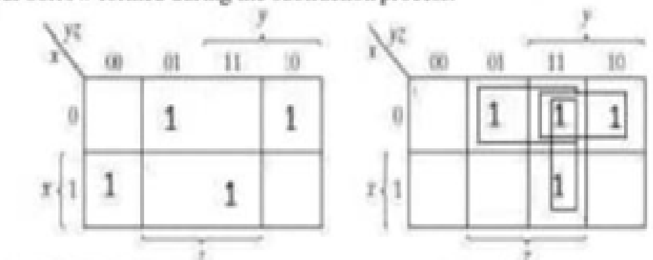


x	y	z	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Truth Table

- 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.



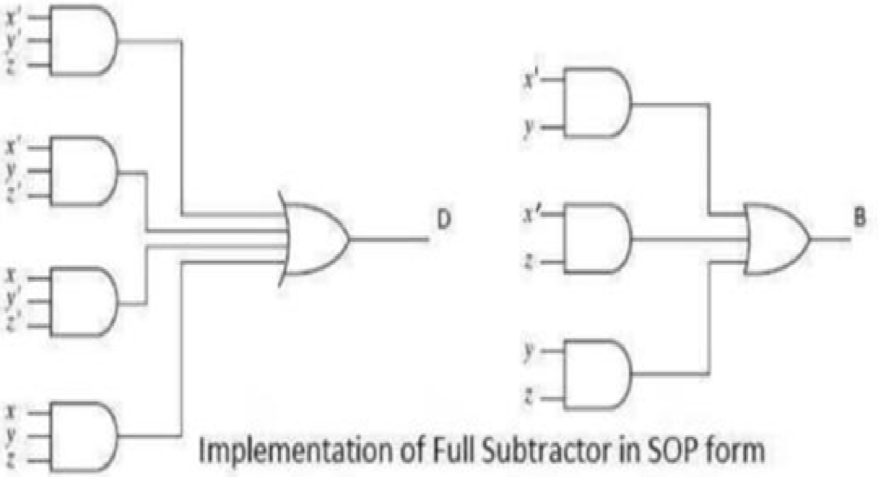
D = x'y'z+x'yz'+xy'z'+xyz

B = x'z+x'y+yz

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 the borrow 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

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



Implementation of Full Subtractor in SOP form

MAGNITUDE COMPARATOR:-

- A magnitude comparator is a combinational circuit that compares two numbers A and B and determines their relative magnitudes.
- The following description is about a 2-bit magnitude comparator circuit.
- The outcome of the comparison is specified by three binary variables that indicate whether A < B, A = B, or A > B.
- Consider two numbers, A and B, with two digits each. Now writing the coefficients of the numbers in descending order of significance:

$$A = A_1$$
$$A_0\ B =$$
$$B_1\ B_0$$

- The two numbers are equal if all pairs of significant digits are equal i.e. if and only if A1 = B1, and A0 = B0.
- When the numbers are binary, the digits are either 1 or 0, and the equality of each pair of bits can be expressed logically with an exclusive-NOR function as

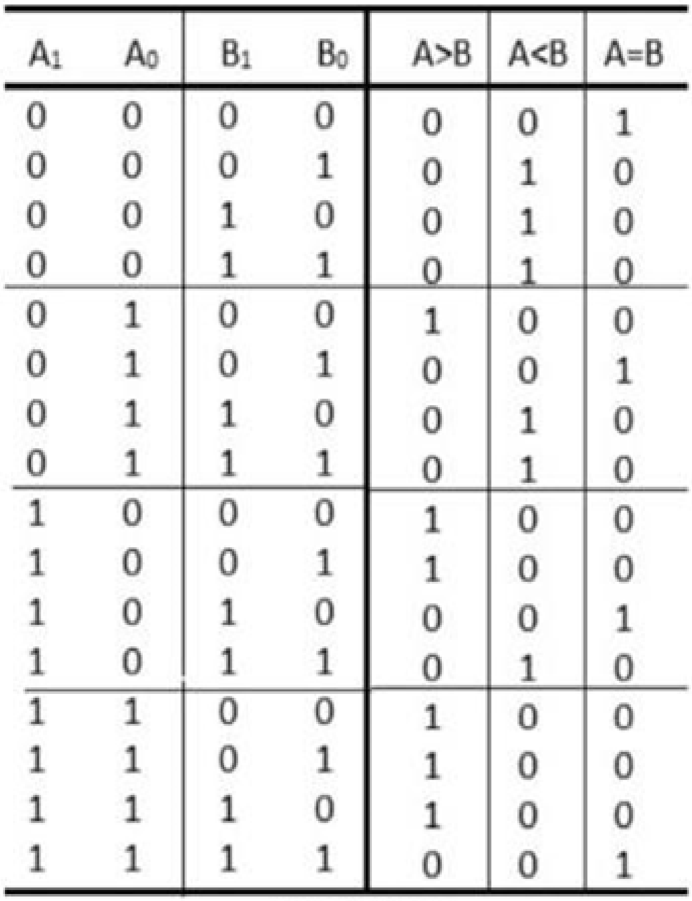
$$x1=A_1B_1+A_1'B_1'$$
$$\text{And } x0=A_0B_0+A_0'B_0'$$

- The equality of the two numbers A and B is displayed in a combinational circuit by an output binary variable that we designate by the symbol (A = B).
- This binary variable is equal to 1 if the input numbers, A and B , are equal, and is equal to 0 otherwise.
- For equality to exist, all xi variables must be equal to 1, a condition that dictates an AND operation of all variables:

$$(A = B) = x_1x_0$$

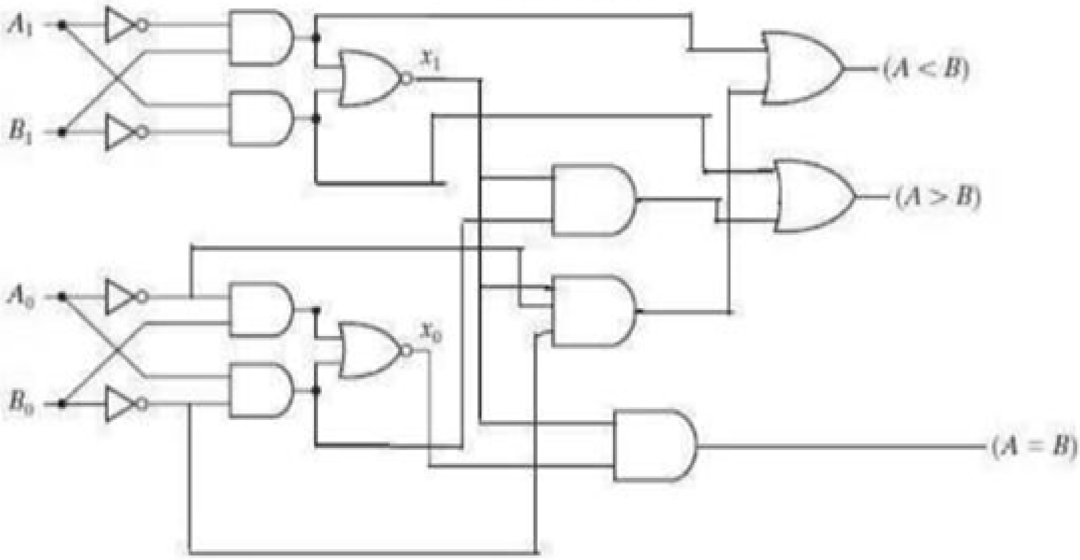
- The binary variable (A = B) is equal to 1 only if all pairs of digits of the two numbers are equal.
- To determine whether A is greater or less than B, we inspect the relative magnitudes of pairs of significant digits, starting from the most significant position. If the two digits of a pair are equal, we compare the next lower significant pair of digits. If the corresponding digit of A is 1 and that of B is 0, we conclude that A > B. If the corresponding digit of A is 0 and that of B is 1, we have A < B. The sequential comparison can be expressed logically by the two Boolean functions

$$(A > B) =$$
$$A_1B_1'+x_1A_0B'_0\ (A < B)$$
$$= A_1'\ B_1 +x_1A_0'B_0'$$



A_1	A_0	B_1	B_0	A>B	A<B	A=B
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	1
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	0	1	1	0	1	0
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	0	1

Truth Table

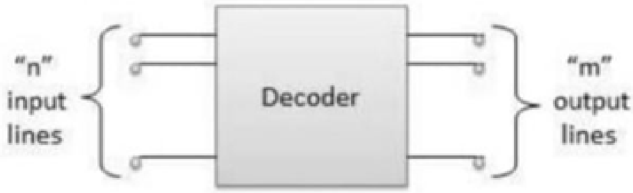


Logic Diagram of 2-bit Magnitude Comparator

Decoder

A decoder is a combinational circuit. It has n input and to a maximum m = 2n outputs. Decoder is identical to a demultiplexer without any data input. It performs operations which are exactly opposite to those of an encoder.

Block diagram



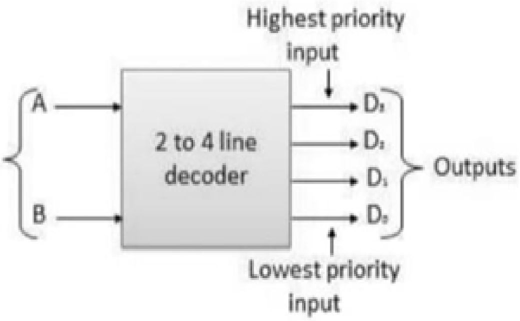
Examples of Decoders are following.

- Code converters
- BCD to seven segment decoders
- Nixie tube decoders
- Relay actuator

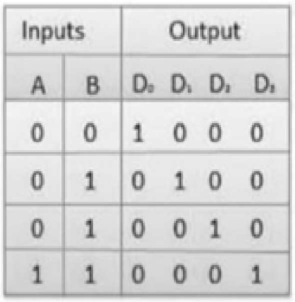
2 to 4 Line Decoder

The block diagram of 2 to 4 line decoder is shown in the fig. A and B are the two inputs where D through D are the four outputs. Truth table explains the operations of a decoder. It shows that each output is 1 for only a specific combination of inputs.

Block diagram

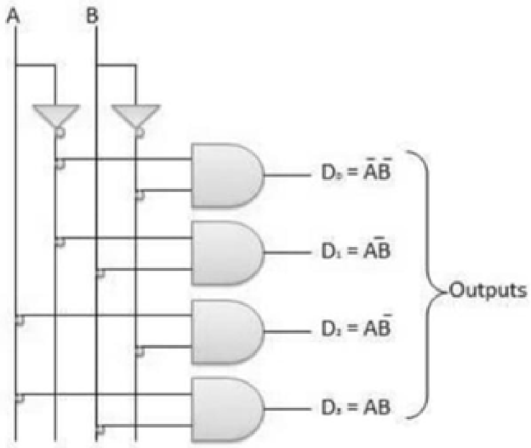


Truth Table



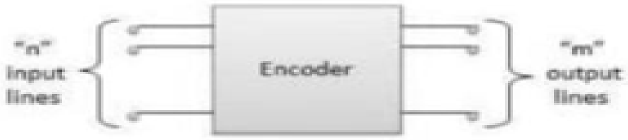
Inputs		Output			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
0	1	0	0	1	0
1	1	0	0	0	1

Logic Circuit



Encoder

Encoder is a combinational circuit which is designed to perform the inverse operation of the decoder. An encoder has n number of input lines and m number of output lines. An encoder produces an m bit binary code corresponding to the digital input number. The encoder accepts an n input digital word and converts it into an m bit another digital word. Block diagram



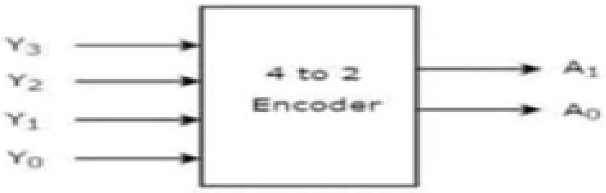
Examples of Encoders are following.

- Priority encoders
- Decimal to BCD encoder
- Octal to binary encoder
- Hexadecimal to binary encoder

4 to 2 Encoder

Let 4 to 2 Encoder has four inputs Y_3, Y_2, Y_1& Y_0 and two outputs A_1& A_0.

The **block diagram** of 4 to 2 Encoder is shown in the following figure.



At any time, only one of these 4 inputs can be '1' in order to get the respective binary code at the output. The **Truth table** of 4 to 2 encoder is shown below.

Inputs				Outputs	
Y_3	Y_2	Y_1	Y_0	A_1	A_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

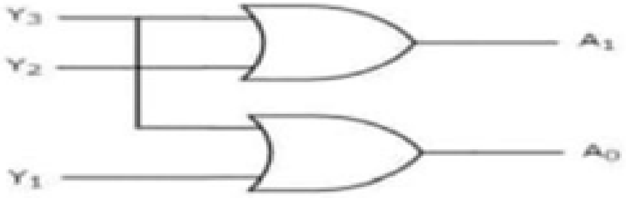
From Truth table, we can write the **Boolean functions** for each output as

$$A1 = Y3 + Y2$$

$$A0 = Y3 + Y1$$

We can implement the above two Boolean functions by using two input OR gates.

The **circuit diagram** of 4 to 2 encoder is shown in the following figure.

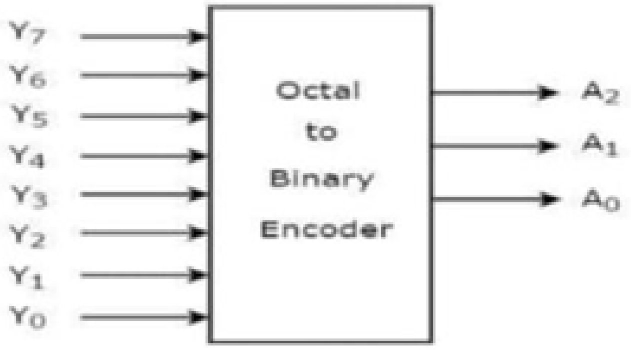


The above circuit diagram contains two OR gates. These OR gates encode the four inputs with two bits

Octal to Binary Encoder

Octal to binary Encoder has eight inputs, Y_7 to Y_0 and three outputs A_2, A_1& A_0. Octal to binary encoder is nothing but 8 to 3 encoder.

The **block diagram** of octal to binary Encoder is shown in the following figure.



At any time, only one of these eight inputs can be '1' in order to get the respective binary code. The **Truth table** of octal to binary encoder is shown below.

Inputs								Outputs		
Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

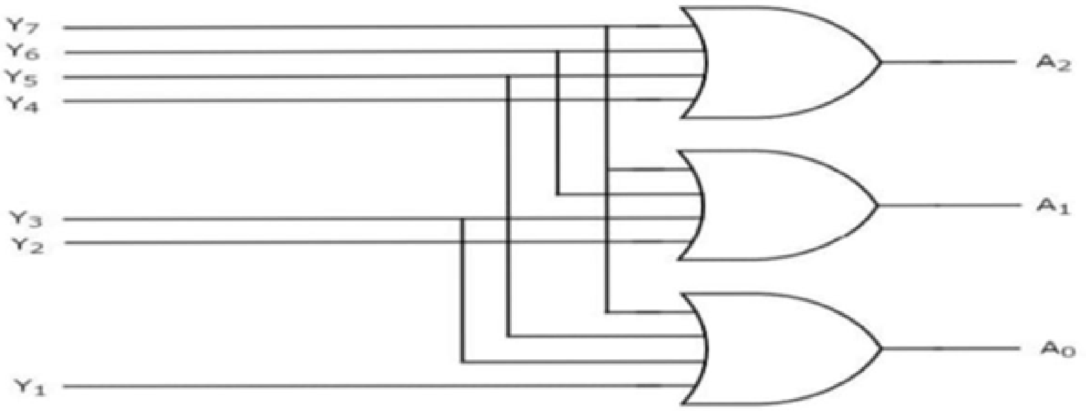
From Truth table, we can write the **Boolean functions** for each output as

$$A2 = Y7 + Y6 + Y5 + Y4$$

$$A1 = Y7 + Y6 + Y3 + Y2$$

$$A0 = Y7 + Y5 + Y3 + Y1$$

We can implement the above Boolean functions by using four input OR gates.



The above circuit diagram contains three 4-input OR gates. These OR gates encode the eight inputs with three bits.

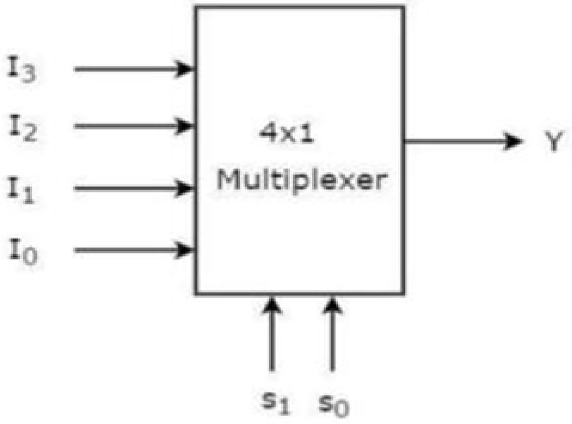
Multiplexer:-

Multiplexer is a combinational circuit that has maximum of 2^n data inputs, „n" selection lines and single output line. One of these data inputs will be connected to the output based on the values of selection lines.

Since there are „n" selection lines, there will be 2^n possible combinations of zeros and ones. So, each combination will select only one data input. Multiplexer is also called as **Mux**.

4x1 Multiplexer

4x1 Multiplexer has four data inputs I_3, I_2, I_1 & I_0, two selection lines s_1 & $s_0$and one output Y. The **block diagram** of 4x1 Multiplexer is shown in the following figure.



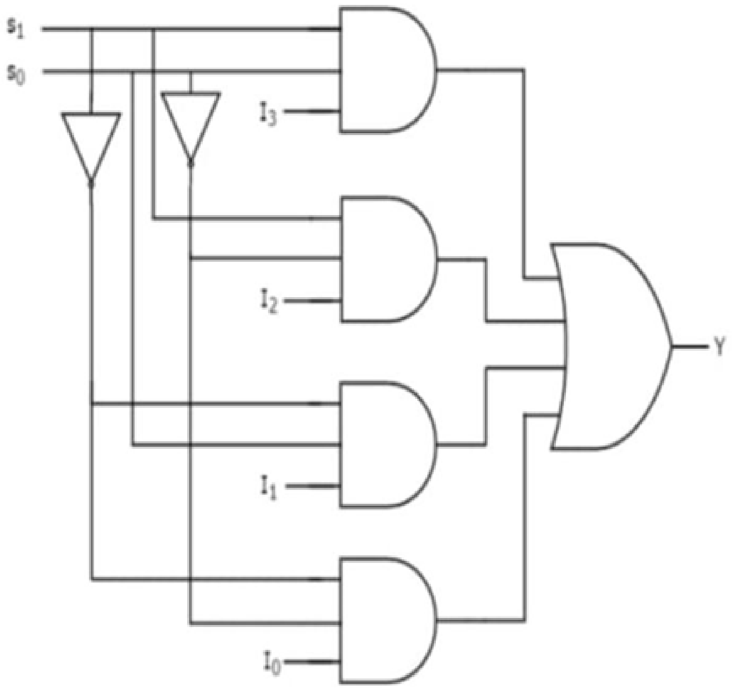
One of these 4 inputs will be connected to the output based on the combination of inputs present at these two selection lines. **Truth table** of 4x1 Multiplexer is shown below.

Selection Lines		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

From Truth table, we can directly write the **Boolean function** for output, Y as

Y=S1′S0′I0+S1′S0I1+S1S0′I2+S1S0I3Y=S1′S0′I0+S1′S0I1+S1S0′I2+S1S0I3

We can implement this Boolean function using Inverters, AND gates & OR gate. The **circuit diagram** of 4x1 multiplexer is shown in the following figure.



Applications of Multiplexer:

Multiplexer are used in various fields where multiple data need to be transmitted using a single line. Following are some of the applications of multiplexers -

1. **Communication system** – Communication system is a set of system that enable communication like transmission system, relay and tributary station, and communication network. The efficiency of communication system can be increased considerably using multiplexer. Multiplexer allow the process of transmitting different type of data such as audio, video at the same time using a single transmission line.
2. **Telephone network –** In telephone network, multiple audio signals are integrated on a single line for transmission with the help of multiplexers. In this way, multiple audio signals can be isolated and eventually, the desire audio signals reach the intended recipients.
3. **Computer memory** - Multiplexers are used to implement huge amount of memory into the computer, at the same time reduces the number of copper lines required to connect the memory to other parts of the computer circuit.
4. **Transmission from the computer system of a satellite** – Multiplexer can be used for the transmission of data signals from the computer system of a satellite or spacecraft to the ground system using the GPS (Global Positioning System) satellites.

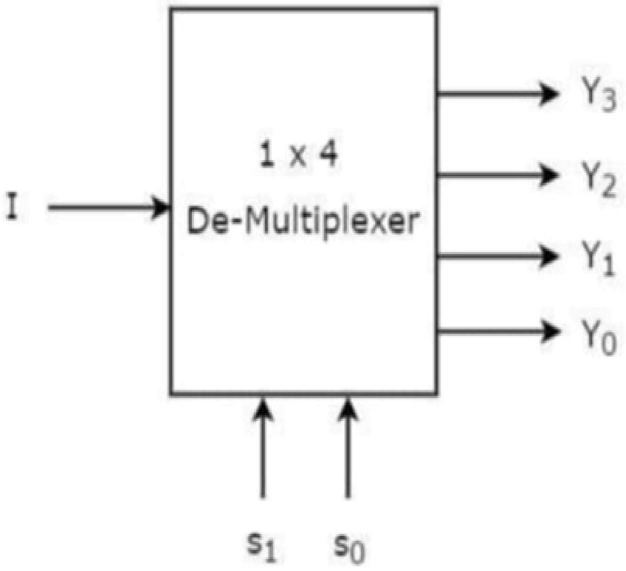
De-Multiplexer

De-Multiplexer is a combinational circuit that performs the reverse operation of Multiplexer. It has single input, „n" selection lines and maximum of 2^noutputs. The input will be connected to one of these outputs based on the values of selection lines.

Since there are „n" selection lines, there will be 2^n possible combinations of zeros and ones. So, each combination can select only one output. De-Multiplexer is also called as **De-Mux**.

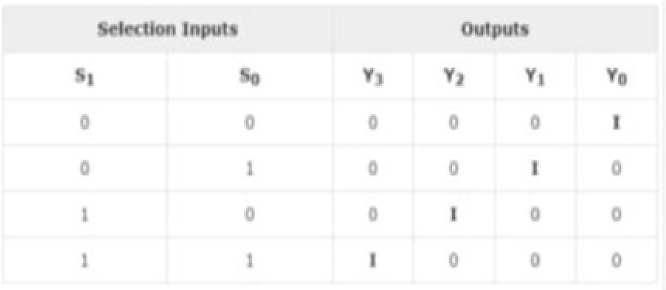
1x4 De-Multiplexer

1x4 De-Multiplexer has one input I, two selection lines, s_1 & s_0 and four outputs Y_3, Y_2, Y_1 &Y_0. The **block diagram** of 1x4 De-Multiplexer is shown in the following figure.



The single input „I" will be connected to one of the four outputs, Y_3 to Y_0 based on the values of selection lines s_1 & s0. The **Truth table** of 1x4 De-Multiplexer is shown below.

From the above Truth table, we can directly write the **Boolean functions** for each output as



Selection Inputs		Outputs			
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

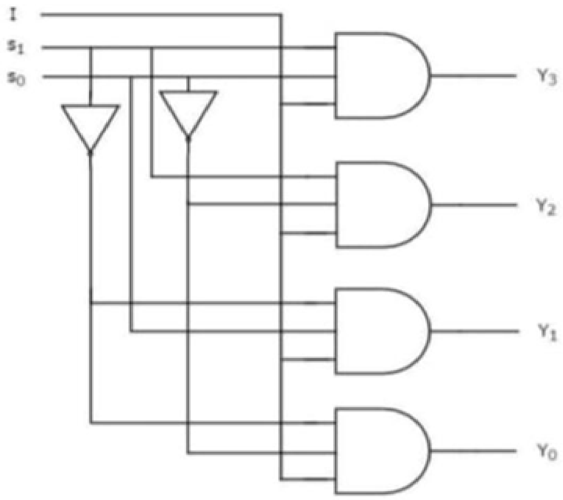
Y3=s1s0I

Y2=s1s0I′

Y1=s1′s0I

Y0=s1′s0I′

We can implement these Boolean functions using Inverters & 3-input AND gates. The **circuit diagram** of 1x4 De-Multiplexer is shown in the following figure.



We can easily understand the operation of the above circuit. Similarly, you can implement 1x8 De-Multiplexer and 1x16 De-Multiplexer by following the same procedure.

Applications of Demultiplexer:

1. Demultiplexer is used to connect a single source to multiple destinations. The main application area of demultiplexer is communication system where multiplexer are used. Most of the communication system are bidirectional i.e. they function in both ways (transmitting and receiving signals). Hence, for most of the applications, the multiplexer and demultiplexer work in sync. Demultiplexer is also used for reconstruction of parallel data and ALU circuits.
2. **Communication System** - Communication system use multiplexer to carry multiple data like audio, video and other form of data using a single line for transmission. This process makes the transmission easier. The demultiplexer receives the output signals of the multiplexer and converts them back to the original form of the data at the receiving end. The multiplexer and demultiplexer work together to carry out the process of transmission and reception of data in communication system.
3. **ALU (Arithmetic Logic Unit)** – In an ALU circuit, the output of ALU can be stored in multiple registers or storage units with the help of demultiplexer. The output of ALU is fed as the data input to the demultiplexer. Each output of demultiplexer is connected to multiple register which can be stored in the registers.
4. **Serial to parallel converter** - A serial to parallel converter is used for reconstructing parallel data from incoming serial data stream. In this technique, serial data from the incoming serial data stream is given as data input to the demultiplexer at the regular intervals. A counter is attaching to the control input of the demultiplexer. This counter directs the data signal to the output of the demultiplexer where these data signals are stored. When all data signals have been stored, the output of the demultiplexer can be retrieved and read out in parallel.

LECTURE NOTES
ON
SEQUENTIAL
LOGIC CIRCUITS(Unit-3)

Prepared by
Er.Aditya Narayan Jena
Dept. of Electronics & Telecommunication Engg.

PNS SCHOOL OF ENGG. & TECH.
Nishamani Vihar,Marshaghai,Kendrapara

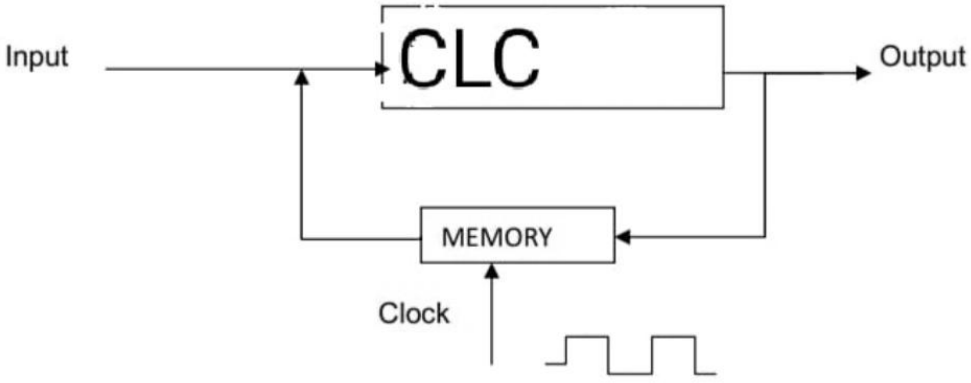
SEQUENTIAL LOGIC CIRCUIT

SEQUENTIAL CIRCUIT:-

- It is a circuit whose output depends upon the present input, previous output and the sequence in which the inputs are applied.

HOW THE SEQUENTIAL CIRCUIT IS DIFFERENT FROM COMBINATIONAL CIRCUIT? :-

- In combinational circuit output depends upon present input at any instant of time and do not use memory. Hence previous input does not have any effect on the circuit. But sequential circuit has memory and depends upon present input and previous output.
- Sequential circuits are slower than combinational circuits and these sequential circuits are harder to design.



[Block diagram of Sequential Logic Circuit]

- The data stored by the memory element at any given instant of time is called the present state of sequential circuit.

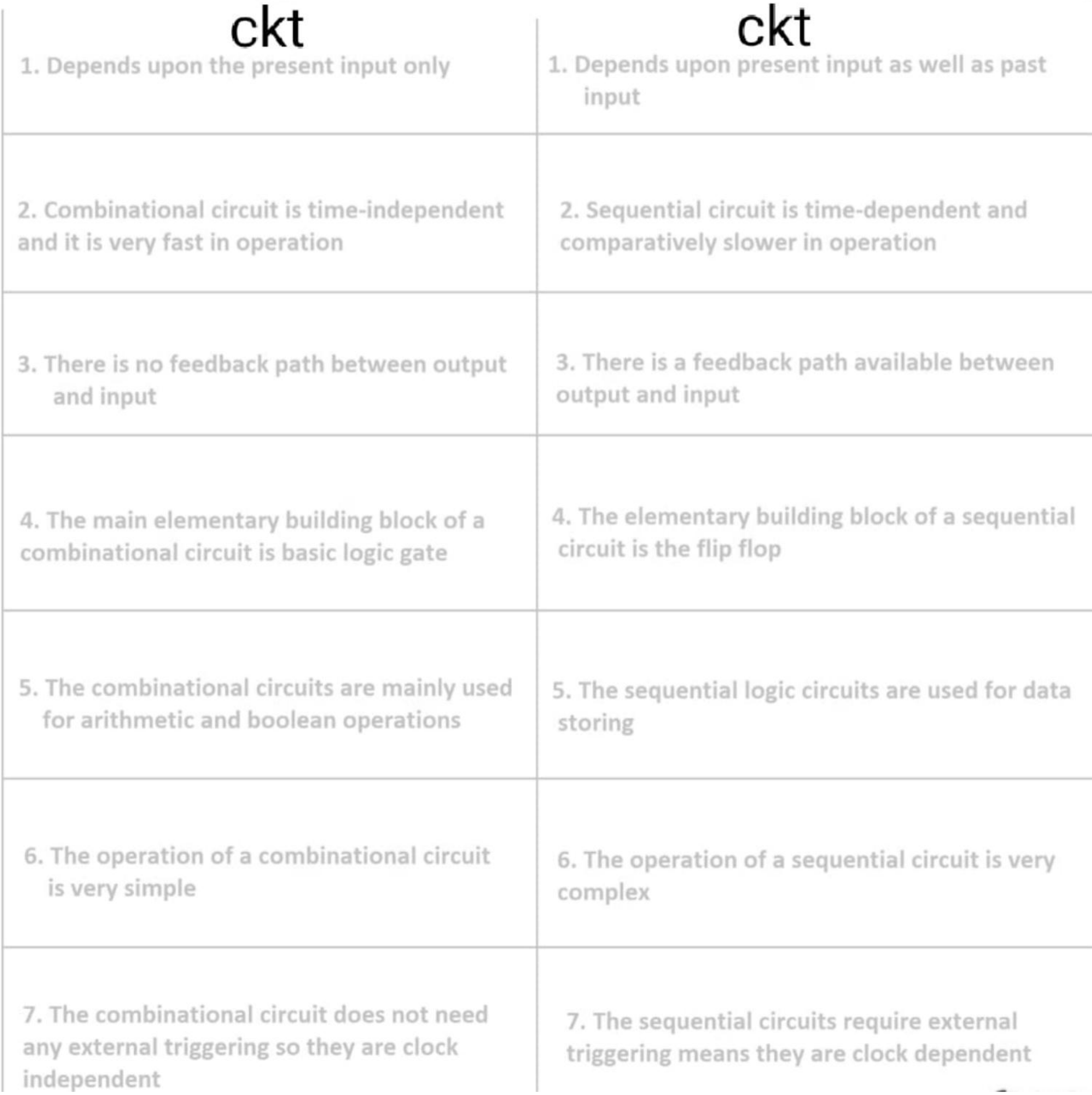
TYPES:-

Sequential logic circuits (SLC) are classified as

(i) Synchronous SLC
(ii) Asynchronous SLC

- The SLC that are controlled by clock are called synchronous SLC and those which are not controlled by a clock are asynchronous SLC.
- Clock:- A recurring pulse is called a clock.

Difference between

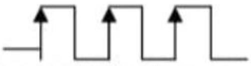


combinational logic ckt	sequential logic ckt
1. Depends upon the present input only	1. Depends upon present input as well as past input
2. Combinational circuit is time-independent and it is very fast in operation	2. Sequential circuit is time-dependent and comparatively slower in operation
3. There is no feedback path between output and input	3. There is a feedback path available between output and input
4. The main elementary building block of a combinational circuit is basic logic gate	4. The elementary building block of a sequential circuit is the flip flop
5. The combinational circuits are mainly used for arithmetic and boolean operations	5. The sequential logic circuits are used for data storing
6. The operation of a combinational circuit is very simple	6. The operation of a sequential circuit is very complex
7. The combinational circuit does not need any external triggering so they are clock independent	7. The sequential circuits require external triggering means they are clock dependent

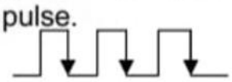
FLIP-FLOP AND LATCH:-

- 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, the output 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 input signal to switch state.
- Storage element that operate with signal level are called latches and those operate with clock transition are called as flip-flops.

- The circuit can be made to change state by signals applied to one or more control inputs and will have one 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 are applied 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- going edge of the clock pulse.



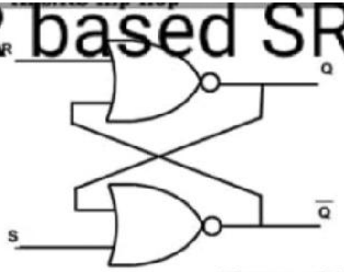
- Negative-edge triggered flip-flops are those in which state transition take place only at negative- going edge of the clock pulse.



Types of FFs:-

- 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

NOR based SR-FF:-



Inputs		Outputs	
R	S	Q	Status
0	0	Last State	No Change
1	0	0	Reset
0	1	1	Set
1	1	Forbidden	Race

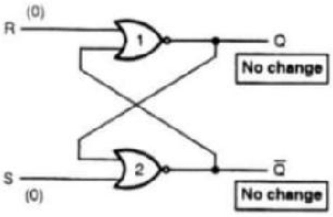
Figures : RS FlipFLop with NOR Gate.

In figure output of one NOR gates drives one of the input of the other NOR gate. The S and R inputs are used to set and reset the flip flop respectively.

Note : For the NOR gate , if any input of the NOR gate is '1' its output will be 0 irrespective of other inputs.

Operation of RS FlipFLop

Case 1: When R = 0 and S = 0 and Q = 0 , Q=1



When S = 0, R = 0 and Q=0 , Q=1 a '0' comes out from the upper NOR gate corresponding to Q = 0.

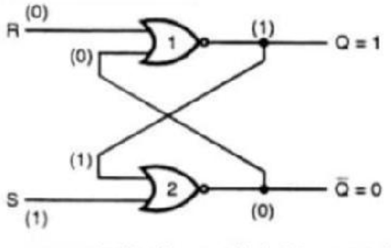
Now the lower NOR gate has both input '0' and hence a '1' comes out from the lower NOR gate corresponding to Q = 1.

Hence when R= 0 , S =0 Flip Flop remain in last state or No change State

Case 2: When R = 0 and S = 1 and Q = 0 , Q=1

When S = 1, R = 0 and Q=0 , Q=1 a '0' comes out from the upper NOR gate corresponding to Q = 0.

Now the lower NOR gate has one input '0' and other input as 1(Q=0 , S=1) ,hence a '0' comes out from the lower NOR gate corresponding to Q = 0.



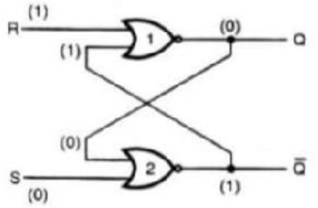
This Q = 0 is fed as input to upper NOR gate making R =0 and Q = 0 , this time a "1" come upper NOR gate.

Now the lower NOR gate has both input as 1(Q=1 , S=1) ,hence a '0' comes out from the lower NOR gate corresponding to Q = 0.

This process repeats till the output is fixed or settled .Hence at the end when R= 0 , S =0 Flip Flop out Q =1 and Q=0.

Hence when R= 0 , S =1 Flip Flop goes in Set state

Case 3: When R = 1 and S = 0 and Q = 1 , Q=0



When S = 0, R = 1 and Q=1, Q=0 a '0' comes out from the upper NOR gate corresponding to Q = 0.

Now the lower NOR gate has both input as '0' (Q=0 , S=0) ,hence a '1' comes out from the lower NOR gate corresponding to Q = 1.

This Q = 1 is fed as input to upper NOR gate making R =1 and Q = 1 , this time a "0" come upper NOR gate.

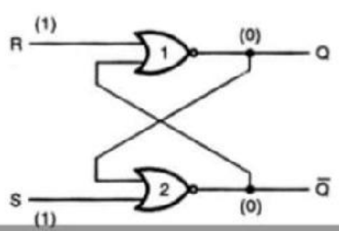
Now the lower NOR gate has both input as 0(Q=0 , S=0) ,hence a '1' comes out from the lower NOR gate corresponding to Q = 1.

This process repeats till the output is fixed or settled .Hence at the end when R=1 , S =0 Flip Flop out Q =0 and Q=1.

Hence when R=1 , S =0 Flip Flop goes in Reset state

Case 4: When R = 1 and S = 0 and Q = 1 , Q=0

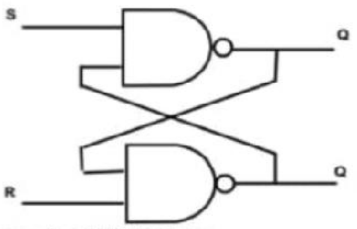
If S = 1 and R = 1 a '0' comes out of both NOR gates giving Q = Q = 1. This is condition is forbidden.



NAND Based SR-FF:-

Inputs		Outputs	
R	S	Q	Status
1	1	Last State	No Change
0	1	0	Reset
1	0	1	Set
0	0	Forbidden	Race

Truth Table

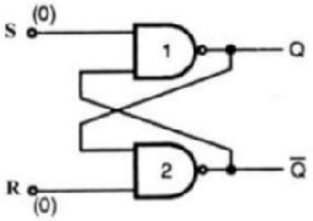


SR Flip-Flop using NAND Gate

In figure output of one NAND gates drives one of the input of the other NAND gate. The S and R inputs are used to set and reset the flip flop respectively.

Note : For the NAND gate , if any input of the NAND gate is '0' its output will be 1 irrespective of other inputs.

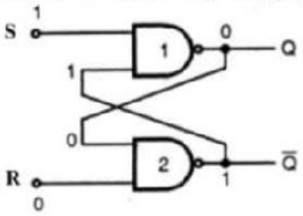
Case 1 : S=0 , R =0 Q =0 , Q=1 (Race Condition)



When S = 1, R = 1 and Q=0 , Q=1 a '1' comes out from the upper NAND gate corresponding to Q = 1.

Now the lower NAND gate has one input '0' and Other input as 1 and hence a '1' comes out from the lower NAND gate corresponding to Q = 1. Hence when R=0 , S =0 Flip Flop both outputs try to become one , this undefined or illegal or Forbidden state. This condition is called as RACE condition.

Case 1 : S=0 , R =1 Q =0 , Q=1 (SET Condition)

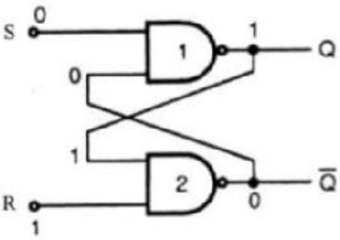


When S = 0, R = 1 and Q=0 , Q=1 a '1' comes out from the upper NAND gate corresponding to Q = 1.

Now the lower NAND gate has both input '1' ,hence a '0' comes out from the lower NAND gate corresponding to Q = 0.

This state remains as its , Hence when R=1 , S = 0 Flip Flop , output Q = 1 and Q = 0 , and the state is called as Set State

Case 2 : S=1 , R =0 Q =1 , Q=0 (RESET Condition)

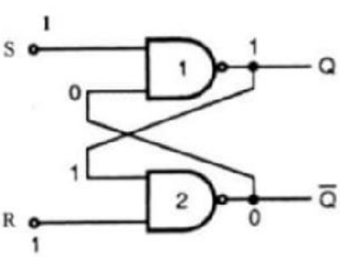


When S = 1, R = 0 and Q=1 , Q=0 a '1' comes out from the upper NAND gate corresponding to Q = 1.

Now the lower NAND gate has both input '1' ,hence a '0' comes out from the lower NAND gate corresponding to Q = 1.This 1 is fed as input to upper gate , now this time upper NAND gate both input as 1 , due to which output is 0.

Now this output is fed as input to lower NAND gate, whose both input are 0 , making output 1.This state remains as its , Hence when R=0 , S=1 Flip Flop , output Q = 0 and Q = 1, and the state is called as Reset State

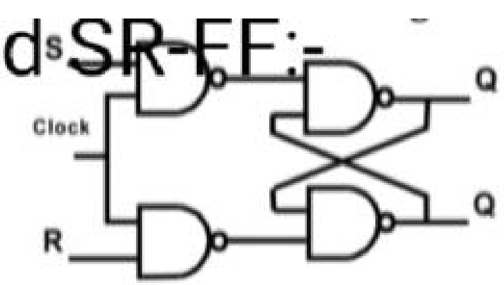
Case 4: When R = 1 and S = 1 and Q = 1 , Q = 0

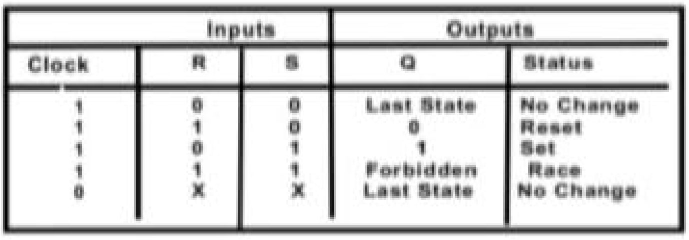


When S = 1, R = 1 and Q=1 , Q=0 a '1' comes out from the upper NAND gate corresponding to Q = 1.

Now the lower NAND gate has both input '1' and hence a '0' comes out from the lower NAND gate corresponding to Q = 0. Hence when R= 0 , S =0 Flip Flop remain in last state or No change State

Clocked SR-FF:-





Inputs			Outputs	
Clock	R	S	Q	Status
1	0	0	Last State	No Change
1	1	0	0	Reset
1	0	1	1	Set
1	1	1	Forbidden	Race
0	X	X	Last State	No Change

Figure Clocked RS Flip-Flop

It is often required to set or reset the memory cell in synchronism with a train of the pulse known as Clock. Such circuit is referred to as **clocked SR (set-reset flip-flop)**

The clock is a square wave signal because the clock drives both NAND and prevents S and R from controlling the latch.

Operation is as Follows

Case 1 : S =1 , R=0 (Set Condition)

- If S = 1 and R = 0 the output of gate A = 0 and B = 1. Now with clock = 1, S = 0, R = 1 and flip-flop set Q = 1 and Q= 0. I.e **Set Condition**

Case 2 : S =0 , R=1 (Reset Condition)

- If S = 0 and R = 1 the output of gate A = 1 and B = 0. Thus with clock = 1, S = 1, R = 0 and flip-flop set Q = 0 and Q = 1. I.e **Reset Condition**

Case 3 : S =1 , R=1 (Illegal/Forbidden Condition)

- With S = 1 and R = 1 the output of both gates will be 0 it is a forbidden condition state or a race condition. **I.e Illegal Condition or Forbidden State**

Case 4 : S =0 , R=0 (Last State Condition)

- When both S = 0, R = 0 and clock = 1 the output A & B gate = 1 which keep the flip-flop in last state. **I.e Last State**

D-flipFlop:-

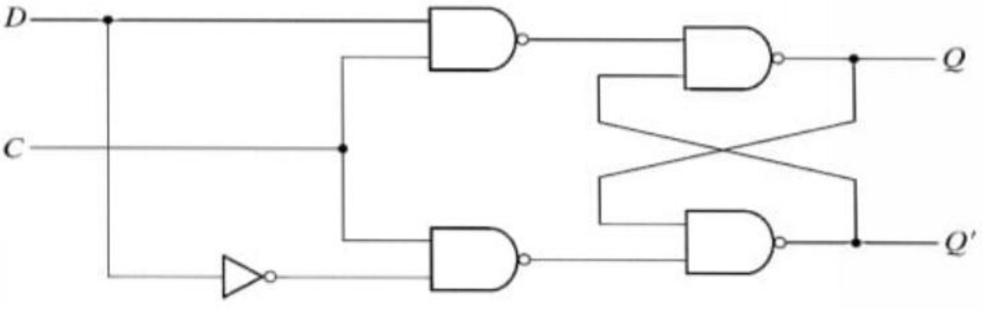
- D flip flop is actually a slight modification of the above explained clocked SR flip-flop. From the figure you can see that the D input is connected to the S input and the complement of the D input is connected to the R input.

- The D input is passed on to the flip flop when the value of CP is '1'.

- When CP is HIGH, the flip flop moves to the SET state. If it is '0', the flip flop switches to the CLEAR state.
- As long as the clock input C = 0, the SR latch has both inputs equal to 0 and it can't change its state regardless of the value of D
- When C is 1, the latch is placed in the set or reset state based on the value of D.

  If D = 1, the Q output goes to 1.

  If D = 0, the Q output goes to 0.



(a) Logic diagram

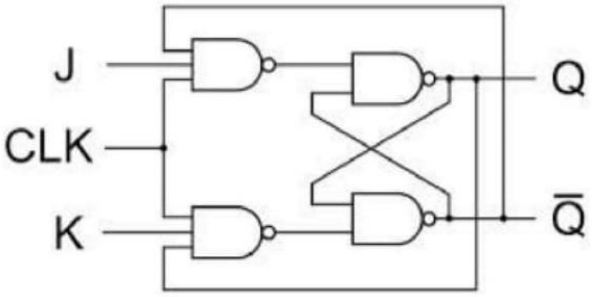
Truth Table:-

C	D	Q_{n+1}
0	0	NC
0	1	NC
1	0	0
1	1	1

JK FLIP-FLOP:-

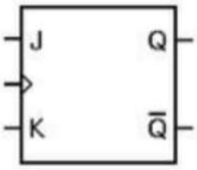
- The JK flip-flop can be constructed by using basic SR latch and a clock. In this case the outputs Q and Q' are returned back and connected to the inputs of NAND gates.
- This simple JK flip Flop is the most widely used of all the flip-flop designs and is considered to be a universal flip-flop circuit.
- The sequential operation of the JK flip flop is exactly the same as for the previous SR flip-flop with the same "Set" and "Reset" inputs.
- The difference this time is that the "JK flip flop" has no invalid or forbidden input states of the SR Latch even 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".
- Due to this additional clocked input, a JK flip-flop has four possible input combinations, "logic 1", "logic 0", "no change" and "toggle".

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



(The above diagram shows the symbol of a JK flip-flop.)

- 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.

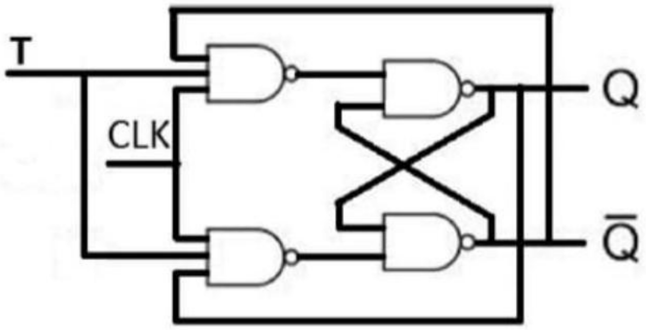
Truth Table of JK-FF:-

Input		Output		Comment
J	K	Q	Q_{next}	
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:-

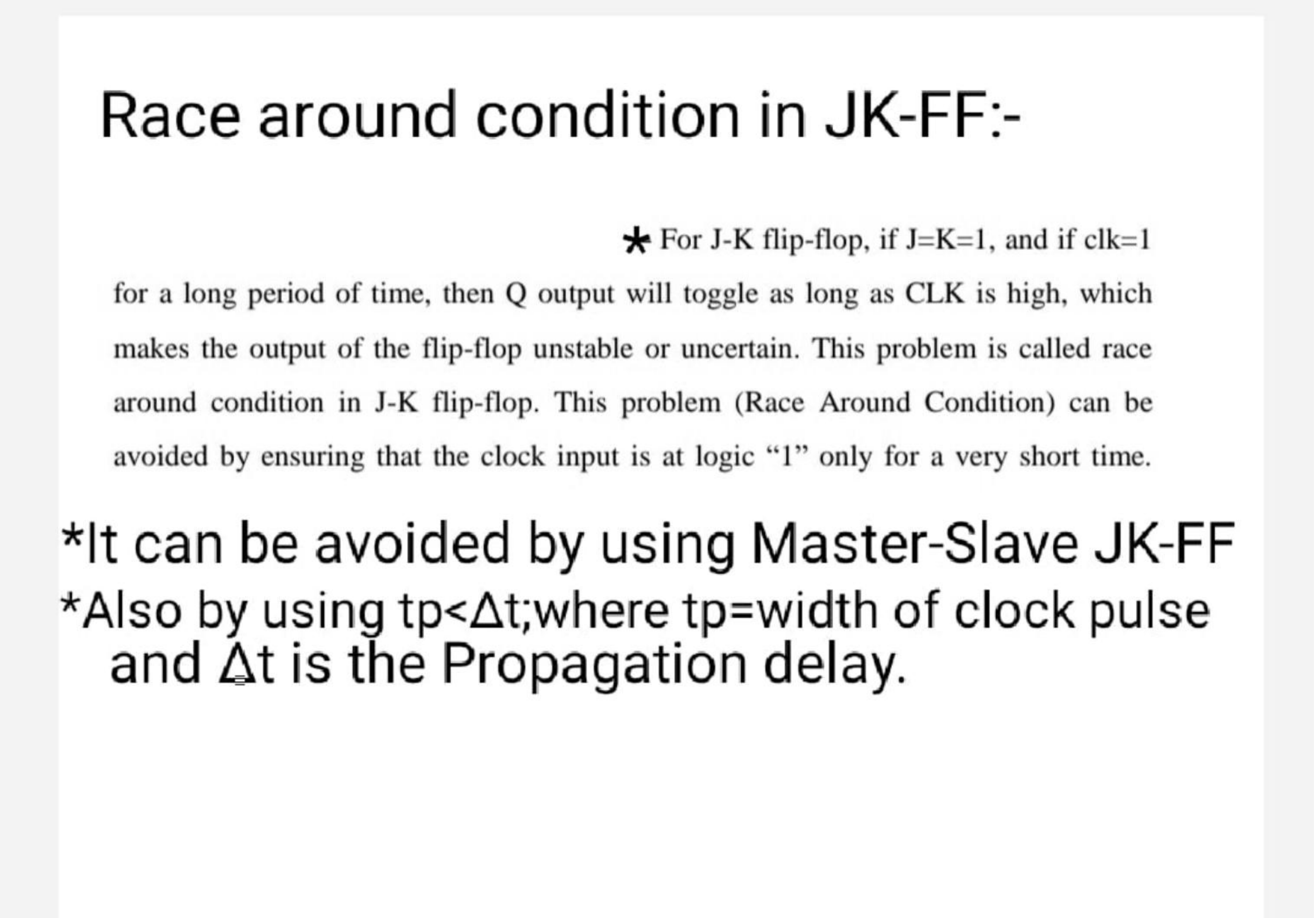
- Toggle flip-flop or commonly known as T flip-flop.
- This flip-flop has the similar operation as that of the JK flip-flop with both the inputs J and K are shorted i.e. both are given the common input.



- 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 as follows.

Truth Table:-

T	Q	Q_{next}	Comment
0	0	0	No change
	1	1	
1	0	1	Toggles
	1	0	



Race around condition in JK-FF:-

* For J-K flip-flop, if J=K=1, and if clk=1 for a long period of time, then Q output will toggle as long as CLK is high, which makes the output of the flip-flop unstable or uncertain. This problem is called race around condition in J-K flip-flop. This problem (Race Around Condition) can be avoided by ensuring that the clock input is at logic “1” only for a very short time.

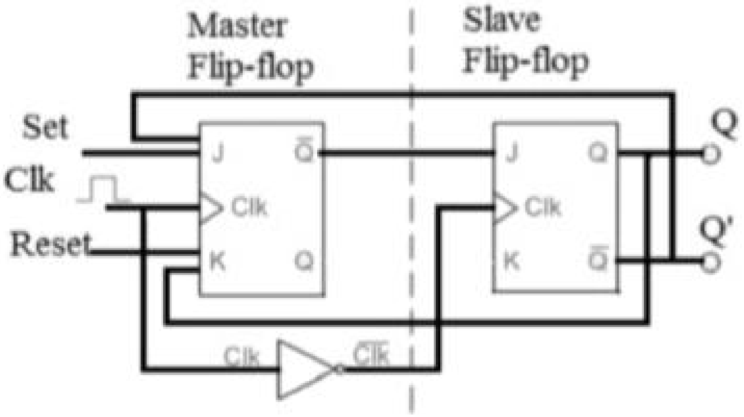
*It can be avoided by using Master-Slave JK-FF

*Also by using tp<Δt;where tp=width of clock pulse and Δt is the Propagation delay.

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.

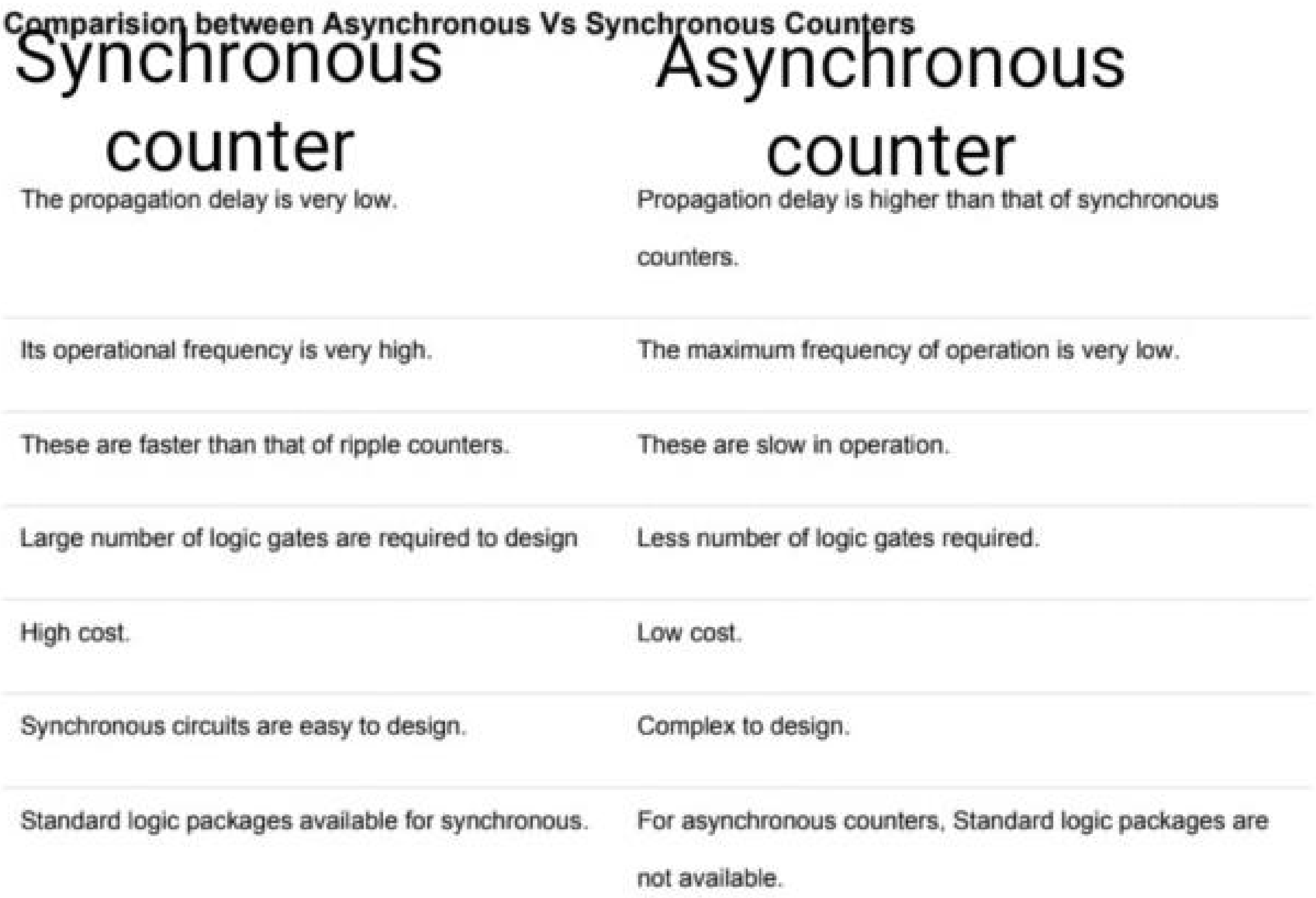
The Master-Slave JK Flip Flop



- The input signals J and K are connected to the gated "master" SR flip flop which "locks" the inputcondition 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.
- Then, the circuit accepts input data when the clock signal is "HIGH", and passes the data to the output on the falling-edge of the clock signal.
- In other words, the Master-Slave JK Flip flop is a "Synchronous" device as it only passes data with the timing of the clock signal.

COUNTER

- A counter is a sequential logic circuit which counts binary numbers.
- There are two types of counters:
  - Asynchronous counter or Ripple counter
  - Synchronous Counters
- In a **Ripple Counters**, First FF is triggered by a clk pulse and remaining FF are triggered by normal or complement output of previous FF.
- In **Synchronous Counters**, all the FFs are triggered by a common clk pulse.
- A counter that follows the binary number sequence is called a *binary counter* (n-bit counter count from 0 to 2^n-1)



Comparision between Asynchronous Vs Synchronous Counters

Synchronous counter	Asynchronous counter
The propagation delay is very low.	Propagation delay is higher than that of synchronous counters.
Its operational frequency is very high.	The maximum frequency of operation is very low.
These are faster than that of ripple counters.	These are slow in operation.
Large number of logic gates are required to design	Less number of logic gates required.
High cost.	Low cost.
Synchronous circuits are easy to design.	Complex to design.
Standard logic packages available for synchronous.	For asynchronous counters, Standard logic packages are not available.

Applications of counters :-

Counter found their applications in many digital electronic devices. Some of their applications are listed below.

1- Frequency counters

2- Digital clocks

3- Analog to digital convertors.

4- With some changes in their design, counters can be used as frequency divider circuits. The frequency divider circuit is that which divides the input frequency exactly by '2'.

5- In time measurement. That means calculating time in timers such as electronic devices like ovens and washing machines.

6- design digital triangular wave generator by using counters.

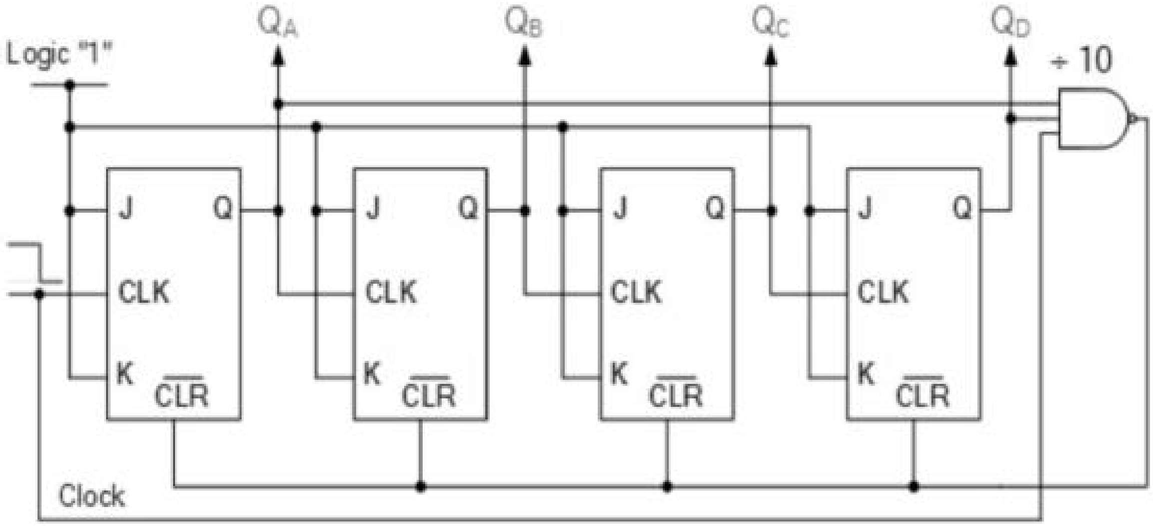
Modulus of a Counter:-

*The no. of states a counter can count is called its Modulus.

For eg;A MOD-5 Counter can count only 5 states.

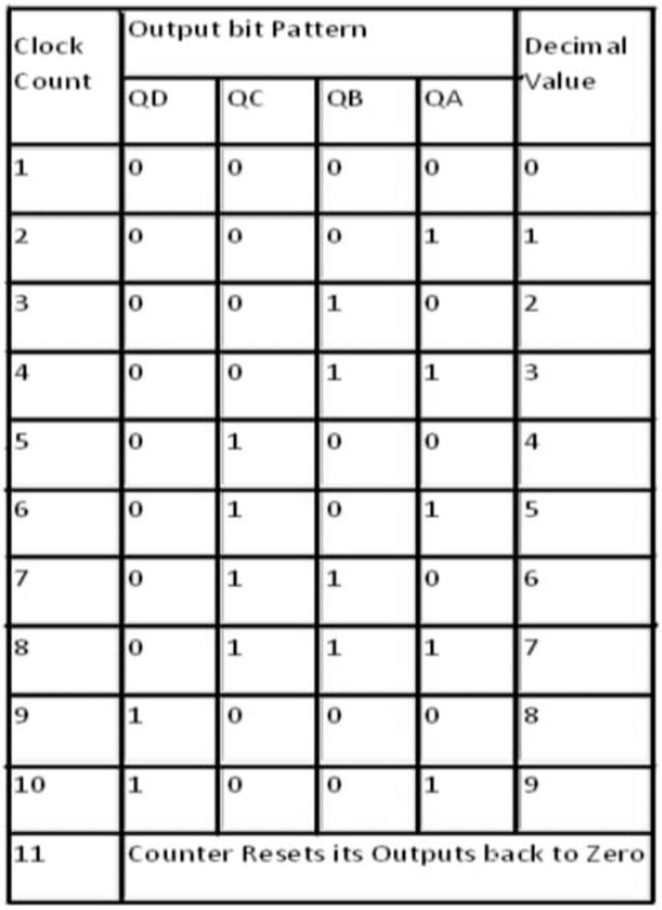
- Modulus counters are used in digital computers.
- A binary modulo-8 counter with three flip-flops, i.e., three stages, will produce an output pulse, i.e., display an output one-digit, after eight input pulses have been counted, i.e., entered or applied. This assumes that the counter started in the zero-condition.

Asynchronous Decade Counter



- A decade counter can count from BCD "0" to BCD "9".
- A decade counter requires resetting to zero when the output count reaches the decimal value of 10, ie. when DCBA = 1010 and this condition is fed back to the reset input.
- A counter with a count sequence from binary "0000" (BCD = "0") through to "1001" (BCD = "9") is generally referred to as a BCD binary-coded-decimal counter because its ten state sequence is that of a BCD code but binary decade counters are more common.
- This type of asynchronous counter counts upwards on each leading edge of the input clock signal starting from 0000 until it reaches an output 1001 (decimal 9).
- Both outputs Q_A and Q_D are now equal to logic "1" and the output from the NAND gate changes state from logic "1" to a logic "0" level and whose output is also connected to the CLEAR (CLR) inputs of all the J-K Flip-flops.
- This signal causes all of the Q outputs to be reset back to binary 0000 on the count of 10. Once QA and QD are both equal to logic "0" the output of the NAND gate returns back to a logic level "1" and the counter restarts again from 0000. We now have a decade or Modulo-10 counter.

Decade Counter Truth Table



Clock Count	Output bit Pattern				Decimal Value
	QD	QC	QB	QA	
1	0	0	0	0	0
2	0	0	0	1	1
3	0	0	1	0	2
4	0	0	1	1	3
5	0	1	0	0	4
6	0	1	0	1	5
7	0	1	1	0	6
8	0	1	1	1	7
9	1	0	0	0	8
10	1	0	0	1	9
11	Counter Resets its Outputs back to Zero				

4-bit Ripple Counter:-

Ans. 4-bit binary asynchronous or serial or ripple counter

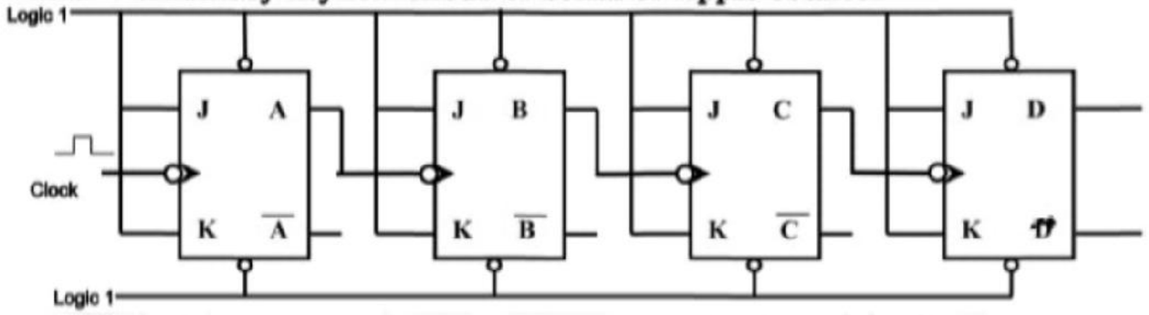


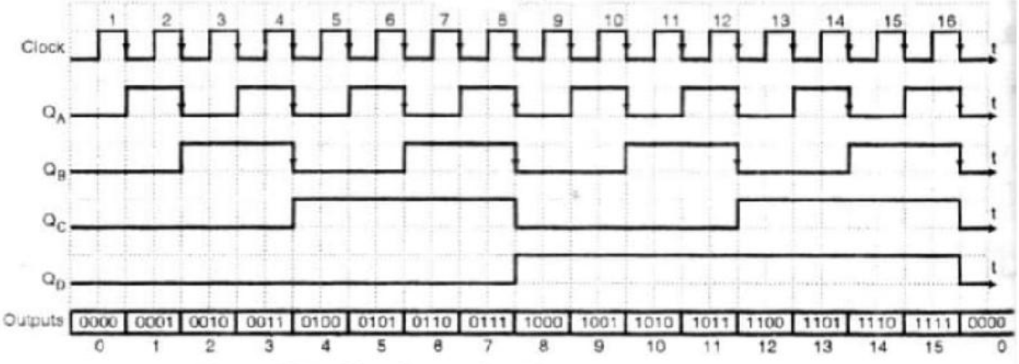
Fig. Shows a 4bit binary asynchronous counter.

It uses four negative edge triggered JK flip-flop. All the flip-flop will operate in the toggle mode because there J and K input are tied to Vcc. The clock pulse are applied to the flip-flop A. The output of flip-flop A drives clock input of flip-flop B, the output of flip-flop B drives clock input of flip-flop C and the output of flip-flop C drives clock input of flip-flop D. Since all flip-flop are negative edge triggered flip-flop they require a transition of 1 to 0 at their clock input to toggle or change the state.

Clock	D	C	B	A	Count
0(Intially)	0	0	0	0	1
1	0	0	0	1	2
2	0	0	1	0	3
3	0	0	1	1	4
4	0	1	0	0	5
5	0	1	0	1	6
6	0	1	1	0	7
7	0	1	1	1	8
8	1	0	0	0	9
9	1	0	0	1	10
10	1	0	1	0	11
11	1	0	1	1	12
12	1	1	0	0	13
13	1	1	0	1	14
14	1	1	1	0	15
15	1	1	1	1	16
16	0	0	0	0	17(0)

Truth Table for 4 bit Aysnchronous Counter

Timing diagram:-



Timing diagram

Operation of Counter

Initially all the flip-flop are cleared by using a common low clear signal. Therefore

DCBA = 0000

On the first clock pulse A flip-flop will toggle from 0 to 1 this will not trigger B flip-flop because it requires a change in 1 to 0 in A. Therefore B remain in last state and since B does change its state also C and D remain in last state. Hence on the first clock pulse we get output as,

DCBA = 0001

On the second clock pulse A flip-flop again toggles from 1 to 0. This now triggers B flip-flop, Now B FlipFlop toggles from 0 to 1. This will not affect C flip-flop because C flip-flop requires a change of 1 to 0 in B flip-flop. Therefore C remains 0 and so is D flip-flop. Hence on 2nd clock pulse we get

DCBA = 0010.

On the 3rd clock pulse A flip-flop changes from 0 to 1, B flip-flop remains at 1 and C and D flip-flop remain at 0 therefore,

DCBA = 0011.

On 4th clock pulse A flip-flop changes from 1 to 0 therefore B flip-flop now changes from 1 to 0. Now C flip-flop is triggered which will change from 0 to 1 but this will not effect D because it requires a change from 1 to 0 in C flip-flop. Hence D remains 0. Therefore or 4th clock pulse we get,

DCBA = 0100.

Thus it is observed that A flip-flop toggles with every clock pulse it receives. B flip-flop toggles whenever A flip-flop changes from 1 to 0.

C flip-flop toggles whenever B flip-flop changes from 1 to 0 and D flip-flop toggles whenever C changes from 1 to 0.

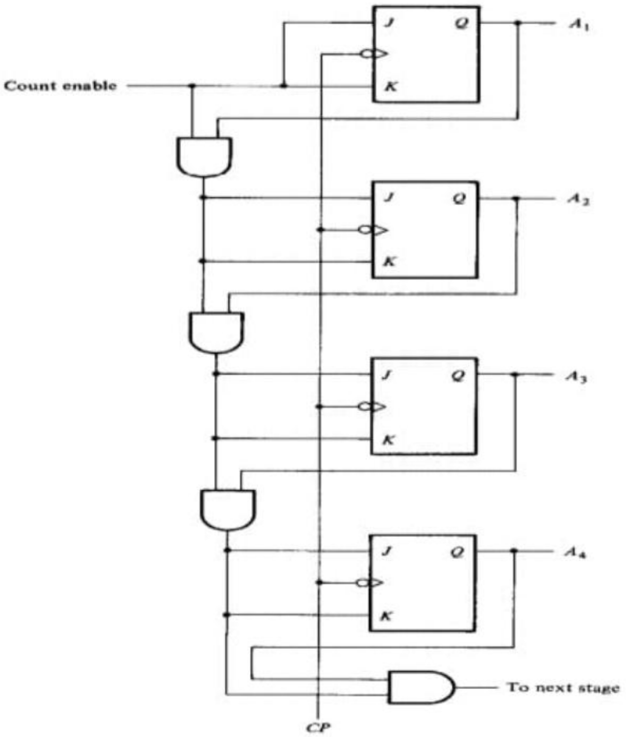
Hence on 15th clock pulse we get DCBA = 1111. On the next clock pulse A flip-flop changes from 1 to 0, B flip-flop change from 1 to 0. Therefore C flip-flop changes from 1 to 0. Hence D changes from 1 to 0, therefore all flip-flop are cleared again & we get,

DCBA = 0000

Thus this counter can count from 0 to 15 i.e. totally 16 count (or states). The number of discrete states through which the counter can progress on the application of pulse is given by 2^n where n= number of flip-flop used into the counter. If we connect 5 flip-flop the counter will progress through 00000 to 11111 i.e. 32 counts (0 to 31).

Synchronous counter

- A 4-bit synchronous counter using JK flip-flops is shown in the figure.
- In synchronous counters, the clock inputs of all the flip-flops are connected together and are triggered by the input pulses. Thus, all the flip-flops change state simultaneously (in parallel).



- The circuit below is a 4-bit synchronous counter.
- The J and K inputs of FF0 are connected to HIGH. FF1 has its J and K inputs connected to the output of FF0, and the J and K inputs of FF2 are connected to the output of an AND gate that is fed by the outputs of FF0 and FF1.
- A simple way of implementing the logic for each bit of an ascending counter (which is what is depicted in the image to the right) is for each bit to toggle when all of the less significant bits are at a logic high state.
- For example, bit 1 toggles when bit 0 is logic high; bit 2 toggles when both bit 1 and bit 0 are logic high; bit 3 toggles when bit 2, bit 1 and bit 0 are all high; and so on.

- Synchronous counters can also be implemented with hardware finite state machines, which are more complex but allow for smoother, more stable transitions.

UNIT-4

REGISTERS,MEMORIES & PLD

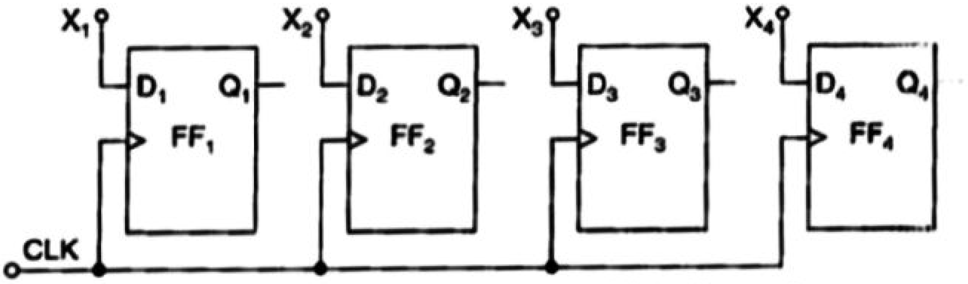
REGISTERS

INTRODUCTION:-

- The sequential circuits known as register are very important logical block in most of the digital systems.
- Registers are used for storage and transfer of binary information in a digital system.
- A register is mostly used for the purpose of storing and shifting binary data entered into it from an external source and has no characteristics internal sequence of states.
- The storage capacity of a register is defined as the number of bits of digital data, it can store or retain.
- These registers are normally used for temporary storage of data.

BUFFER REGISTER:-

- These are the simplest registers and are used for simply storing a binary word.
- These may be controlled by Controlled Buffer Register.
- D flip – flops are used for constructing a buffer register or other flip- flop can be used.
- The figure shown below is a 4- bit buffer register.



Logic diagram of a 4-bit buffer register.

- The binary word to be stored is applied to the data terminals.
- When the clock pulse is applied, the output word becomes the same as the word applied at the input terminals, i.e. the input word is loaded into the register by the application of clock pulse.
- When the positive clock edge arrives, the stored word becomes:

  Q4 Q3 Q2 Q1= X4 X3 X2 X1

  or Q = X .

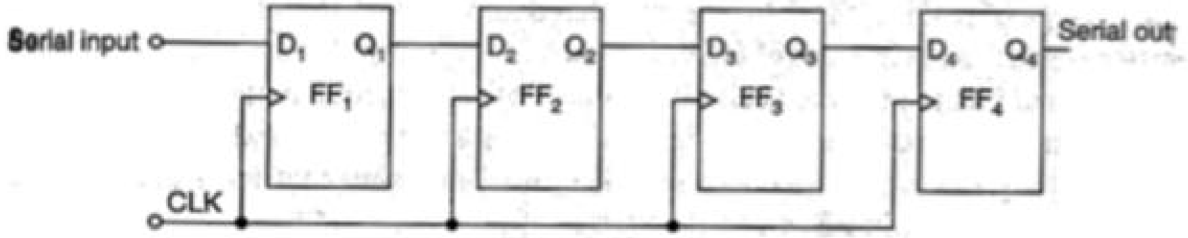
  This circuit is too primitive to be of any use.

Types of Shift Registers:-

- A number of FFs connected together such that data may be shifted into and shifted out of them is called a shift register.
- Data may be shifted into or out of the register either in serial form or in parallel form.
- There are four basic types of shift registers
  1. Serial in, serial out
  2. Serial in, parallel out
  3. Parallel in, serial out
  4. Parallel in , parallel out

SERIAL IN, SERIAL OUT SHIFT REGISTER:-

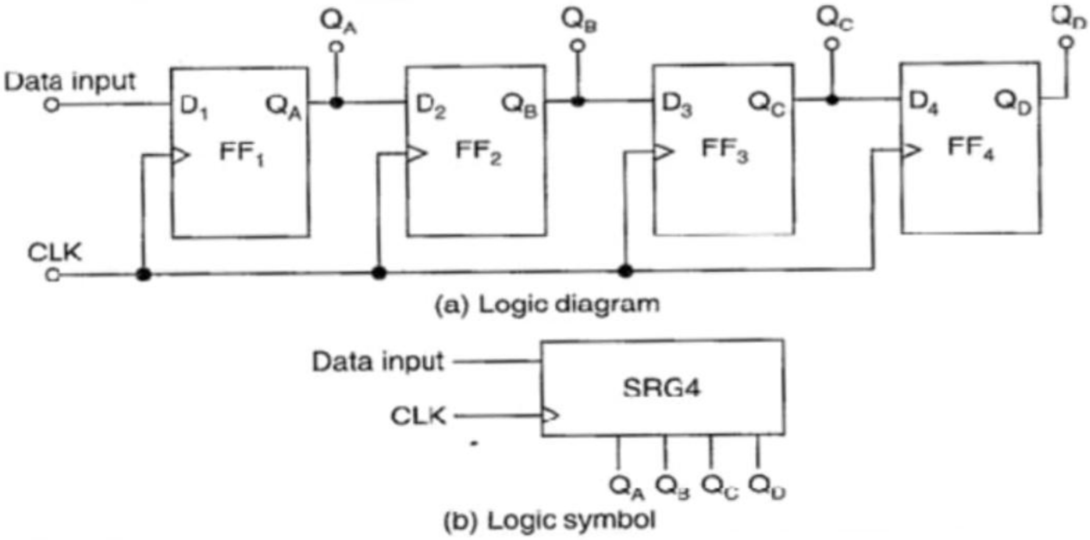
- This type of shift register accepts data serially, i.e., one bit at a time and also outputs data serially.
- The logic diagram of a four bit serial in, serial out shift register is shown in below figure:
- In 4 stages i.e. with 4 FFs, the register can store upto 4 bits of data.
- Serial data is applied at the D input of the first FF. The Q output of the first FF is connected to the D input of the second FF, the output of the second FF is connected to the D input of the third FF and the Q output of the third FF is connected to the D input of the fourth FF. The data is outputted from the Q terminal of the last FF.
- When a serial data is transferred to a register, each new bit is clocked into the first FF at the positive going edge of each clock pulse.
- The bit that is previously stored by the first FF is transferred to the second FF.
- The bit that is stored by the second FF is transferred to the third FF, and so on.
- The bit that was stored by the last FF is shifted out.
- A shift register can also be constructed using J-K FFs or S-R FFs as shown in the figure below.



4-bit serial-in, serial-out, shift-right, shift register.

SERIAL IN, PARALLEL OUT SHIFT REGISTER:-

- In this type of register, the data bits are entered into the register serially, but the data stored in the register serially, but the stored in the register is shifted out in the parallel form.
- When the data bits are stored once, each bits appears on its respective output line and all bits are available simultaneously, rather than bit – by – bit basis as in the serial output.
- The serial in, parallel out shift register can be used as a serial in, serial out shift register if the output is taken from the Q terminal of the last FF.
- The logic diagram and logic symbol of a 4 bit serial in, parallel out shift register is given below.



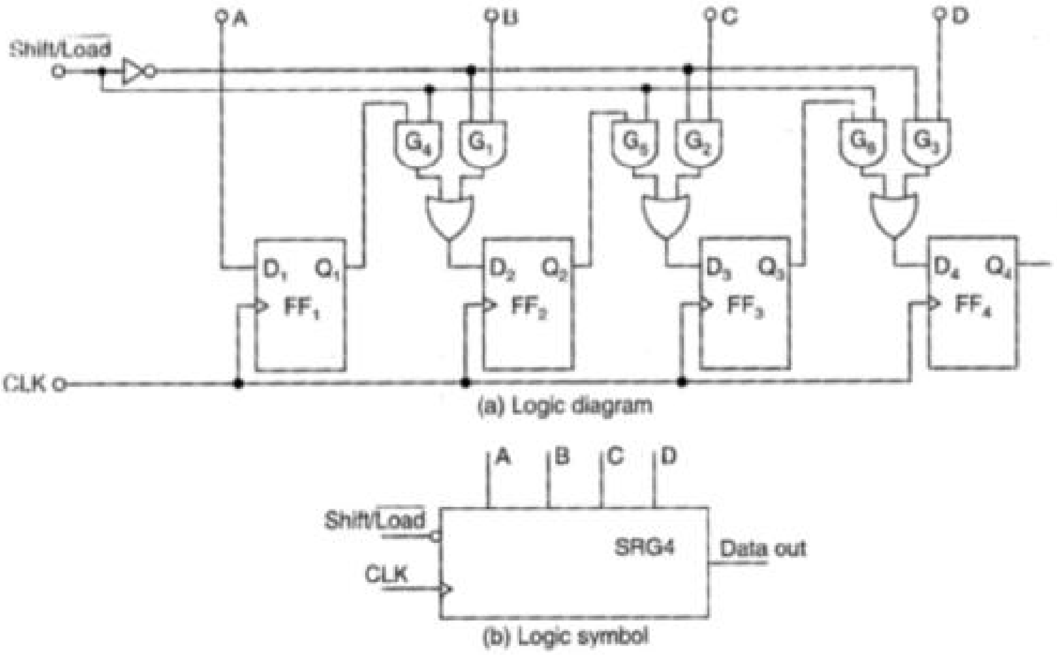
(a) Logic diagram

(b) Logic symbol

A 4- bit serial in, parallel out shift register

PARALLEL IN, SERIAL OUT SHIFT REGISTER:-

- For parallel in, serial out shift register the data bits are entered simultaneously into their respective stages on parallel lines, rather than on bit by bit basis on one line as with serial data inputs, but the data bits are transferred out of the register serially, i.e., on a bit by bit basis over a single line.
- The logic diagram and logic symbol of 4 bit parallel in, serial out shift register using D FFs is shown below.
- There are four data lines A, B, C and D through which the data is entered into the register in parallel form.
- The signal Shift /$\overline{\text{LOAD}}$ allows
  1. The data to be entered in parallel form into the register and
  2. The data to be shifted out serially from terminal Q_4.
- When Shift /$\overline{\text{LOAD}}$ line is HIGH, gates G1, G2, and G3 are disabled, but gates G4, G5 and G6 are enabled allowing the data bits to shift right from one stage to next.
- When Shift /LOAD line is LOW, gates G4, G5 and G6 are disabled, whereas gates G1, G2 and G3 are enabled allowing the data input to appear at the D inputs of the respective FFs.
- When clock pulse is applied, these data bits are shifted to the Q output terminals of the FFs and therefore the data is inputted in one step.
- The OR gate allows either the normal shifting operation or the parallel data entry depending on which AND gates are enabled by the level on the Shift /LOAD input.



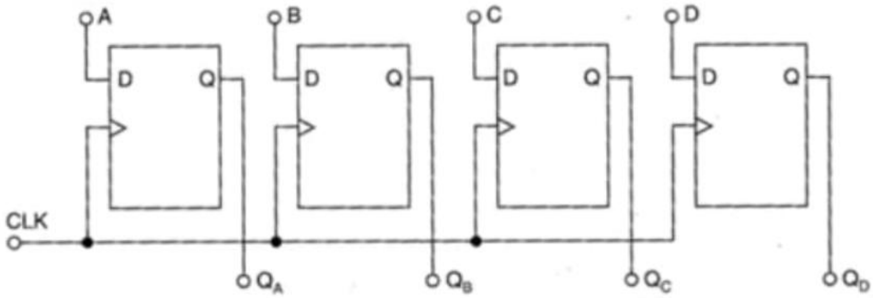
(a) Logic diagram

(b) Logic symbol

A 4- bit parallel in, serial out shift register

PARALLEL IN, PARALLEL OUT SHIFT REGISTER:-

- In a parallel in, parallel out shift register, the data entered into the register in parallel form and also the data taken out of the register in parallel form. Immediately following the simultaneous entry of all data bits appear on the parallel outputs.
- The figure shown below is a 4 bit parallel in parallel out shift register using D FFs.
- Data applied to the D input terminals of the FFs.
- When a clock pulse is applied at the positive edge of that pulse, the D inputs are shifted into the Q outputs of the FFs.
- The register now stores the data.
- The stored data is available instantaneously for shifting out in parallel form.



Logic diagram of a 4 – bit parallel in, parallel out shift register

APPLICATIONS OF SHIFT REGISTERS:-

1. **Time delays:**
   - In digital systems, it is necessary to delay the transfer of data until the operation of the other data have been completed, or to synchronize the arrival of data at a subsystem where it is processed with other data.
   - A shift register can be used to delay the arrival of serial data by a specific number of clock pulses, since the number of stages corresponds to the number of clock pulses required to shift each bit completely through the register.
   - The total time delay can be controlled by adjusting the clock frequency and by the number of stages in the register.
   - In practice, the clock frequency is fixed and the total delay can be adjusted only by controlling the number of stages through which the data is passed.
2. **Serial / Parallel data conversion:**
   - Transfer of data in parallel form is much faster than that in serial form.
   - Similarly the processing of data is much faster when all the data bits are available simultaneously. Thus in digital systems in which speed is important so to operate on data parallel form is used.
   - When large data is to be transmitted over long distances, transmitting data on parallel lines is costly and impracticable.
   - It is convenient and economical to transmit data in serial form, since serial data transmission requires only one line.

   - Shift registers are used for converting serial data to parallel form, so that a serial input can be processed by a parallel system and for converting parallel data to serial form, so that parallel data can be transmitted serially.
   - A serial in, parallel out shift register can be used to perform serial-to parallel conversion, and a parallel in, serial out shift register can be used to perform parallel- to –serial conversion.
   - A universal shift register can be used to perform both the serial- to – parallel and parallel-to-serial data conversion.
   - A bidirectional shift register can be used to reverse the order of data.

Memory Classification/ Memory Hierarchy

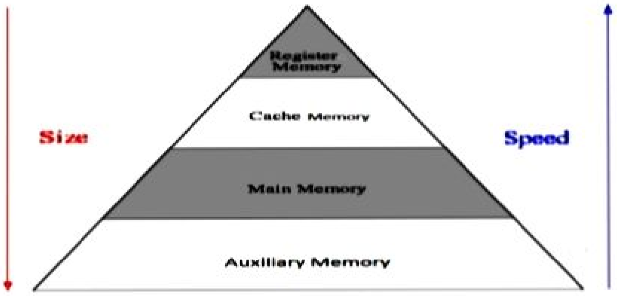
Memory in computer is used for following purpose:

- Stores programs and data during execution.
- Stores programs results of execution.
- Stores programs and data for future reference.

The memory is classified in hierarchical structure as follows:

I. Register Memory.
II. Cache memory.
III. Main memory/ Primary memory.
IV. Auxiliary memory/ Secondary memory.

In hierarchical structure of memory the speed of access increases up the pyramid and the size increases down the pyramid.



I. **Register memory:**
    - Register memories are high speed registers in the CPU, work as a memory for temporary storage of instruction and data.
    - CPU uses registers for the processing of data, the number of registers in a CPU and the size of each register affect the power and speed of a CPU.
    - The more the number of registers and bigger the size of each register, the better it is.

II. **Cache Memory:**
    - Cache memory is a fast memory which is placed in between the CPU and the main memory.
    - Cache memory consists of binary cells and the size of this memory is less than main memory.
    - When the CPU needs an instruction or data during processing, it first looks in the cache memory.
    - If the information is present in the cache, it is called a *cache hit*, and the data or instruction is retrieved from the cache.
    - If the information is not present in cache memory, then it is called a *cache miss* and the information is then retrieved from main memory.

III. **Main Memory/Primary Memory:**

- Data and programs are stored in this memory during execution.
- After execution the result is stored in this memory.
- It is volatile in nature. This means whenever power goes out the content of this memory will be lost.
- The access speed of this memory is higher than the auxiliary memory.
- Main memory consists of RAM & ROM.

RAM (RANDOM ACCESS MEMORY):-

- The access time of this memory to read data from the first location or the last location is same.
- Both read and write operation is possible in this memory.
- This is volatile in nature.
- RAM is of two types:-
  - **a)** SRAM(Static RAM)
  - **b)** DRAM (Dynamic RAM)

a) SRAM (Static RAM)

- SRAM (static RAM) is random access memory (RAM) that retains data bits in its memory as long as power is being supplied.
- SRAM does not have to be periodically refreshed.
- Static RAM provides faster access to data and is more expensive than DRAM.
- SRAM is used for a computer's cache memory.

b) DRAM (Dynamic RAM)

- Dynamic random access memory (DRAM) is the most common kind of random access memory for personal computers and workstations.
- DRAM stores each bit in a storage cell consisting of a capacitor and a transistor.
- Capacitors tend to lose their charge rather quickly; thus, it need for refreshed.

ROM (READ ONLY MEMORY):

- In this memory only read operation is possible.
- The content of this memory is written by the manufacturer.
- It is non-volatile in nature.

a) PROM (PROGRAMMABLE READ ONLY MEMORY):-

- In PROM, user can programme the ROM only once.

b) EPROM (ERASEABLE PROGRAMMABLE READ ONLY MEMORY:-

- In this content can be erased and programmed.
- Erasing is possible by use of ultra-violet rays.

c) EEPROM (ELECTRICALLY ERASEABLE PROGRAMMABLE READ ONLY MEMORY):-

- In this memory erasing the content is possible by applying the high voltage electricity.

IV. **Auxiliary Memory/Secondary Memory:**

- This memory consists of magnetic and optical materials.
- This memory capacity of this memory is high and access speed of this memory is very less.
- It is non-volatile in nature.

EX:- Magnetic disc, Magnetic tapes etc.

Programmable Logic Devices (PLDs)

Introduction:

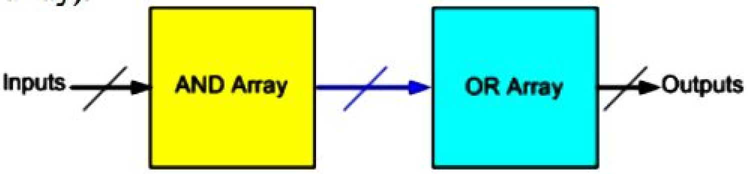
An IC that contains large numbers of gates, flip-flops, etc. that can be *configured by the user* to perform different functions is called a ***Programmable Logic Device (PLD)***.

The internal logic gates and/or connections of PLDs can be changed/configured by a programming process.

One of the simplest programming technologies is to use fuses. In the original state of the device, all the fuses are intact.

Programming the device involves blowing those fuses along the paths that must be removed in order to obtain the particular configuration of the desired logic function.

PLDs are typically built with an *array* of AND gates (AND-array) and an *array* of OR gates (OR-array).



Advantages of PLDs:

Problems of using standard ICs:

Problems of using standard ICs in logic design are that they require hundreds or thousands of these ICs, considerable amount of circuit board space, a great deal of time and cost in inserting, soldering, and testing. Also require keeping a significant inventory of ICs.

Advantages of using PLDs:

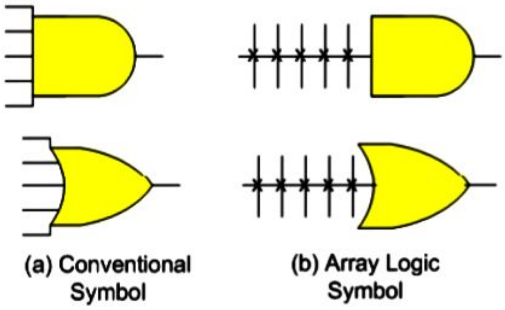
Advantages of using PLDs are less board space, faster, lower power requirements (i.e., smaller power supplies), less costly assembly processes, higher reliability (fewer ICs and circuit connections means easier troubleshooting), and availability of design software.

There are three fundamental types of standard PLDs: ***PROM, PAL,*** and ***PLA***.

A fourth type of PLD, which is discussed later, is the ***Complex Programmable Logic Device (CPLD),*** e.g., ***Field Programmable Gate Array (FPGA).***
A typical PLD may have hundreds to millions of gates.

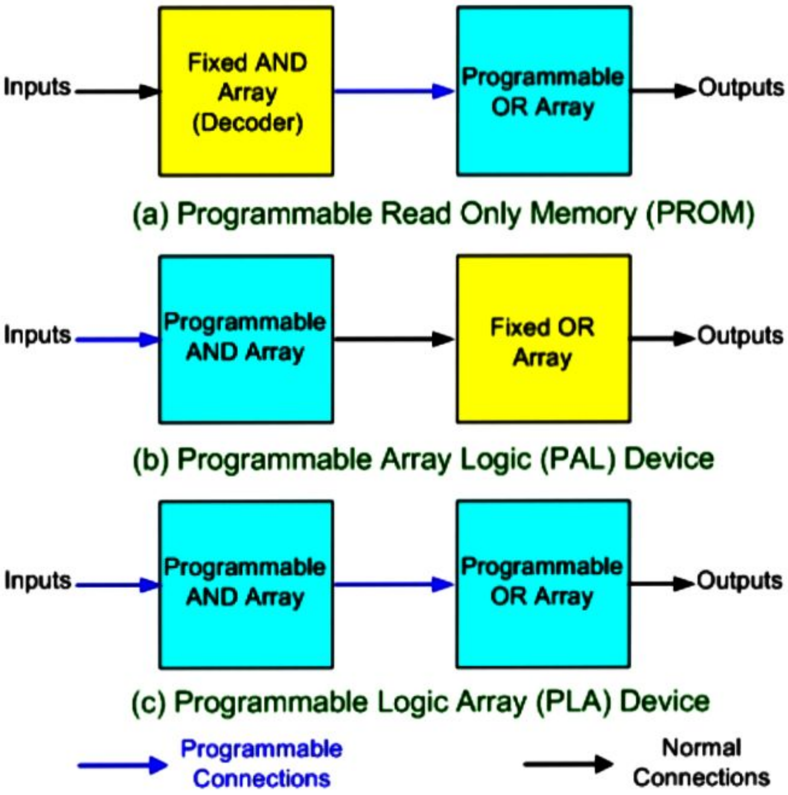
In order to show the internal logic diagram for such technologies in a concise form, it is necessary to have special symbols for array logic.

Figure shows the conventional and array logic symbols for a multiple input AND and a multiple input OR gate.



Three Fundamental Types of PLDs:

The three fundamental types of PLDs differ in the placement of programmable connections in the AND-OR arrays. Figure shows the locations of the programmable connections for the three types.



- The ***PROM (Programmable Read Only Memory)*** has a fixed AND array (constructed as a decoder) and programmable connections for the output OR gates array. The PROM implements Boolean functions in sum-of-minterms form.

- The ***PAL (Programmable Array Logic)*** device has a programmable AND array and fixed connections for the OR array.

- The ***PLA (Programmable Logic Array)*** has programmable connections for both AND and OR arrays. So it is the most flexible type of PLD.

UNIT-5: D/A and A/D Converter

Weighted Register Network

The most significant bit (MSB) resistance is one-eighth of the least significant bit (LSB) resistance. R_L is much larger than 8R. The voltages V_A, V_B, V_C and V_D can be either equal to V (for logic 1) or 0 (for logical 0). Thus there are 2^4 = 16 input combinations from 0000 to 1111. The output voltage V_0, given by Millman's theorem is

V_0 =

When input is 0001, V_A = V_B= V_C= 0 and V_D = V and output is V/15. If input is 0010, V_A = V_B=V_D =0 and V_C=V giving an output of 2V/15. If input is 0011, V_A = V_B=0 and V_C=V_D = V giving an output of 3v/15. Thus, the output voltage varies from 0 to V in steps of V/15.

Binary Ladder Network

The weighted resistor network requires a range of resistor values. The binary ladder network requires only two resistance values. From node 1, the resistance to the digital source is 2R and resistance to ground is also 2R. From node 2, the resistance to digital source is 2R and resistance to ground =

R + (2R) (2R) / (2R+2R) = 2R

Thus, from each of the nodes 1,2,3,4, the resistance to source and ground is 2R each. A digital input 0001 means that D is connected to V and A, B, C are grounded. The output voltage V_0 is V/16. Thus as input varies from 0000 to 1111, the output varies from V/16 to V in steps of V/16.

A complete digital-to-analog converter circuit consists of a number of ladder networks (to deal with more bits of data), operational amplifier, gates etc.

Performance Characteristics of D/A converters

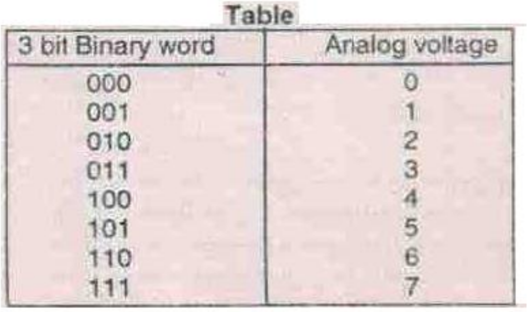
The performance characteristics of D/A converters are resolution, accuracy, linear errors, monotonicity, setting time and temperature sensitivity.

(a) **Resolution:** It is the reciprocal of the number of discrete steps in the D/A output. Evidently resolution depends on the number of bits. The percentage resolution is [1/ (2^N-1)] * 100 where N is the number of bits. The percentage resolution for different values of N is given in table.

(b) **Accuracy:** It is a measure of the difference between actual output and expected output. It is expressed as a percentage of the maximum output voltage. If the maximum output voltage (or full scale deflection) is 5 V and accuracy is ±0.1%, then the maximum error is $\frac{0.1}{100}$ * 5 = 0.005 V or 5 mV. Ideally the accuracy should be better than ±0.5 of LSB. In an 8 bit converter, LSB is 1/256 or 0.39% of full scale. The accuracy should be better than 0.2%.

(c) **Setting Time:** When the input signal changes, it is desirable that analog output signal should immediately show the new output value. However in actual practice, the D/A converter takes some time to settle at the new position of the output voltage. Setting time is defined as the time taken by the D/A converter to settle with ±1/2 LSB of its final value when a change in input digital signal occurs. The final time taken to settle down to new value is due to the transients and oscillations in the output voltage. Figure shows the definition of setting time.

Table



3 bit Binary word	Analog voltage
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Fig 1

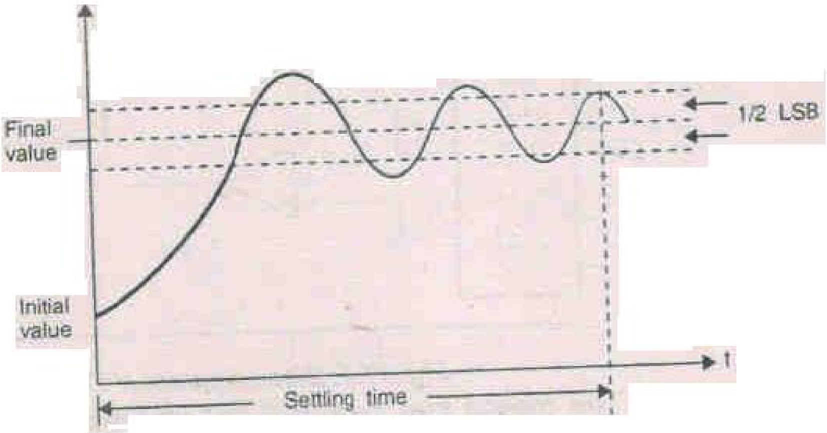


Fig. Settling time of D/A converter

Quantization error:

An analog to digital converter changes analog signal into digital signal. It is important to note that in D/A converter the number of input is fixed. In 4 bit D/a converter there are 16 possible inputs and in 6 bit D/A converter there are 64 possible inputs. However, in A/D converter the analog input voltage can have any value in the specified range but the digital output can have only 2^N discrete levels (for N bit converter). This means that there is a certain range of input voltage which correspond to every discrete output level.

Consider a 4 bit A/D converter having a resolution of 1 count per 100 mV. Fig (b) shows the analog input and digital output. It is seen that for input voltage range of 50 mV to 150 mV, the output is same i.e. 0001, for input voltage range of 150 mV to 250 mV, the output is the same, i.e. 0010. Thus we have one digital output for each 100 mV input range. If the digital signal of 0010 is fed to a D/A converter, it will show an output of 200 V whereas the original input voltage was between 150 V and 250 v. This error is called quntisation error and in this case this quntisation error can be ±50 mV and is equal to ±1/2 LSB.

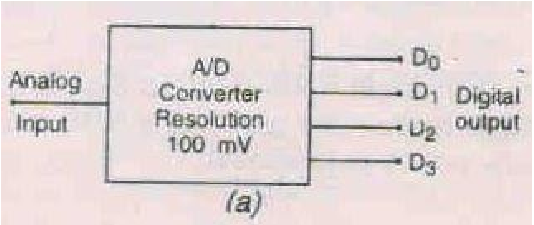


Fig (a) A/D Converter

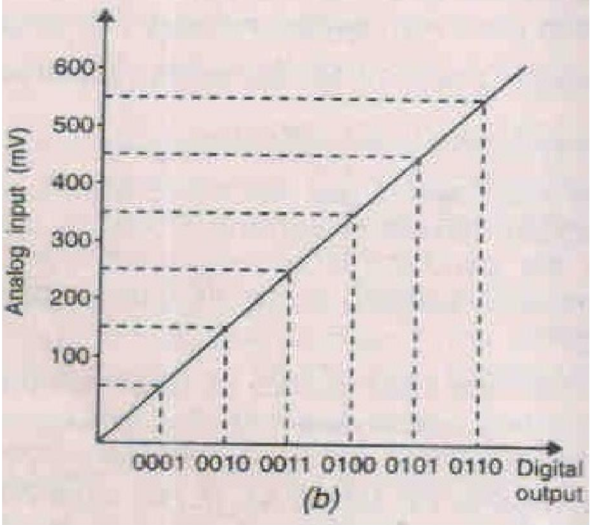


Fig (b) Quantisation error

Stair Step A/D Converter / Ramp A/D converter:

This converter is also called digital ramp or the counter type A/D converter. Figure shows the configuration for 8 bit converter. As seen in figure it uses a D/A converter and a binary counter to produce the digital number corresponding to analog input. The main components are comparator, AND gate, D/A converter, divide by 256 counter and latches. The analog input is given to non-inverting terminal of comparator. The D/A converter provides stair step reference voltage.

Let he counter be in reset state and output of D/A converter be zero. An analog input is given to non-inverting terminal of comparator. Since the reference input is 0, the comparator gives High output and enables the AND gate. The clock pulses cause advancing of counter through its binary states and stair step reference voltage is produced from D/A converter. As the counter keeps advancing, successively higher stair step output voltage is produced. When this stair step voltage reaches the level of analog input voltage, the comparator output goes Low and disables the AND gate. The clock pulses are cut off and counter stops. The state of counter at this point is equal to the number of steps in reference voltage at which comparison occurs. The binary number corresponding to this number of steps is the value of the analog input voltage. The control logic causes this binary number to be loaded into the latches and counter is reset.

This converter is rather slow in action because the counter has to pass through the maximum number of states before a conversion takes place. For 8 bit device this means 256 counter states.

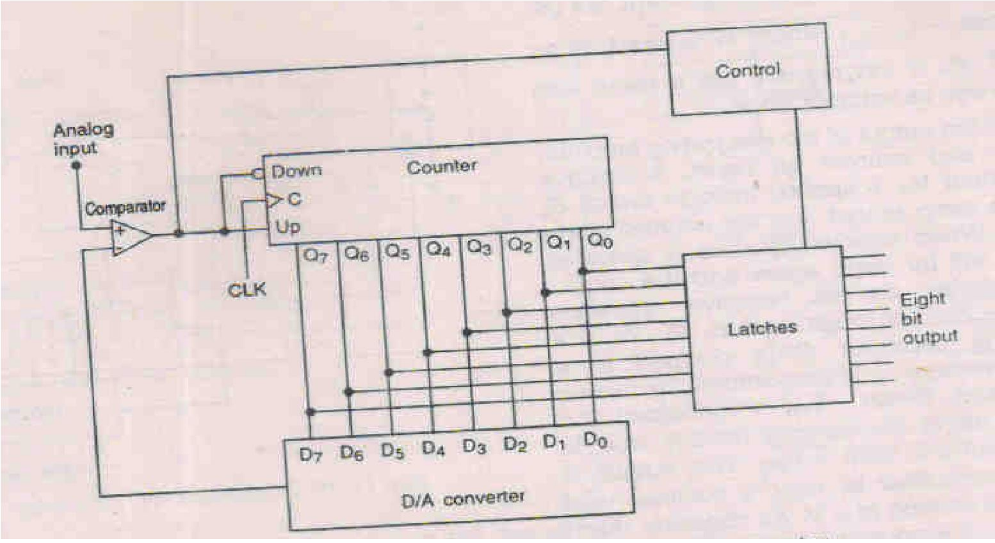


Fig (a) 8 bit up-down counter type A/D converter

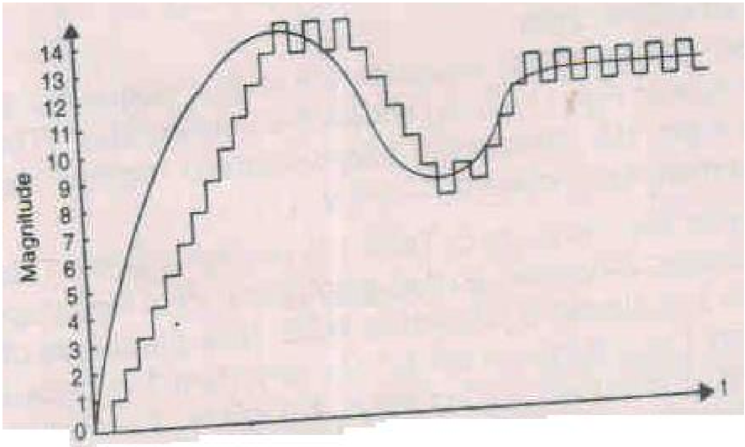


Fig (b) Tracking action of updown counter type
A/D Converter

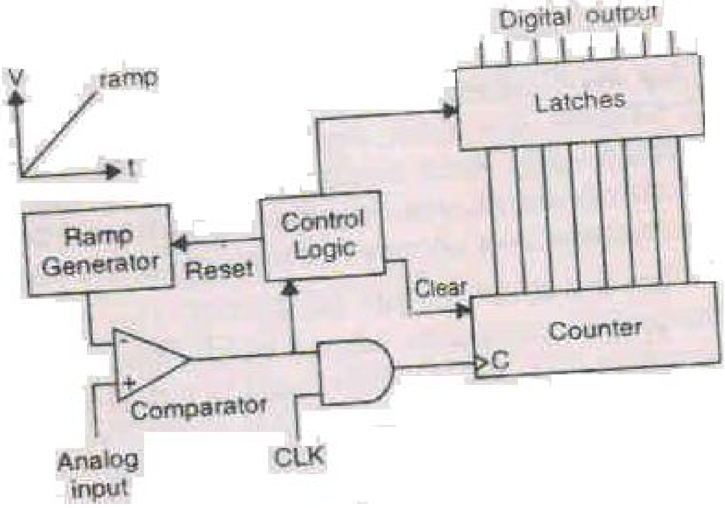


Fig (c) Single slope A/D converter

Dual slope A/D converter:

The single slope A/D converter is suscetible to noise. The dual slope converter is free from this problem. It uses an op-amp used as integreting amplifier for ramp generator. It is dual slope device because it uses a fixed slope ramp as well as variable slope ramp. Fig. Shows the configuration.

It is seen that the integreting op-amp uses a capacitor in the feedback path.

Output voltage of integreting op-amp = - $\frac{1}{C}\int i\,dt$ = - $\frac{1}{RC}\int V_{in}$ dt

Thus the output voltage is integral of analog input voltage. If V_{in} is constant, we get an output - $V_{in}\frac{t}{RC}$ which is a fixed slope ramp. If V_{in} is varing we get a ramp with fixed as well as variable slope.

Let the output of the integreting amplifier be zero and counter be reset. A positive analog input V_{in} is applied through switch S, we get a ramp output and the counter starts working. When counter reaches a specified count, it will be reset again and the control logic switches on the negative reference voltage - V_{ref} (through switch S). At this instant the capacitor C is charged to a negative voltage - V proportional to analog input voltage. When - V_{ref} is connected the capacitor starts discharging linearly due to constant current from - V_{ref}. The output of integreting amplifier is now a positive fixed slope ramp starting at – V. As capacitor discharges, the counter advances from the reset state. When the output of integretor becomes zero, the comparator output becomes Low and disables the clock signal to the AND gate. The counter is therefore stopped and the binary counter is latched. This completes one conversion cycle. The binary count is propor tional to analog input V_{in}.

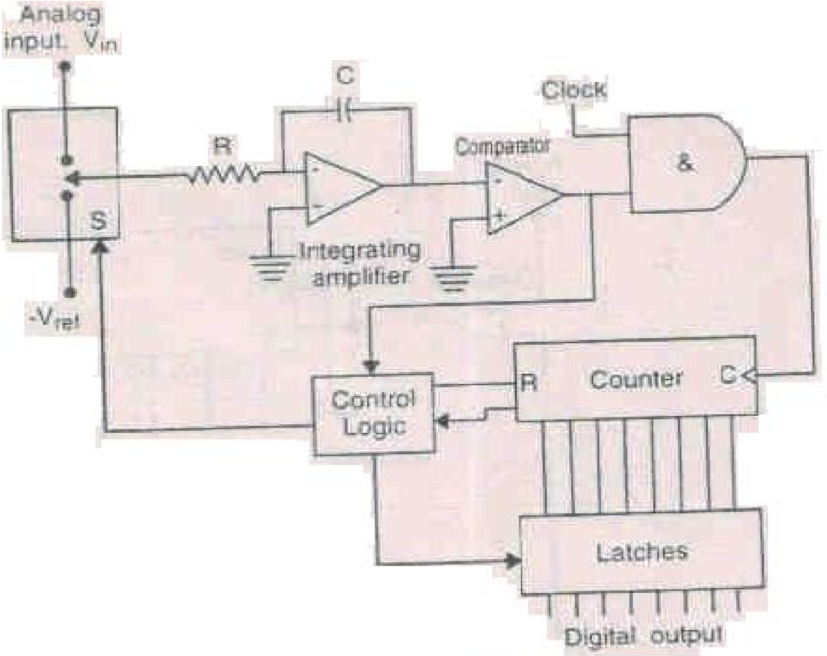


Fig. Dual slope A/D converter

Successive Approximation A/D Converter:

This is the most widely used A/D converter. As the name suggests the digital output tends towards analog input through successive approximations. Fig. Shows the configuration. The main components are op-amp comparator, control logic, SA (successive approximation) register and D/A converter. As shown it is a six bit device using a maximum reference of 64 V.

Let the analog input be 26.1 v. The SA register is first set to zero. Then 1 is placed in MSB. This is fed to D/A converter whose output goes to comparator. Since the analog input (26.1 V) is less than D/A output (i.e. 32 V) the MSB is set to zero. Then 1 is placed in bit next to MSB. Now the output of D/A is 16 V. Since analog input is more than 16 V, this 1 is retained in this bit position. Next 1 is placed in third bit position. Now the D/A output is 24 V which is less than analog input. Therefore this 1 bit is retained and 1 is placed in the next bit. Now the D/A output is 28 V, which is more than analog input. Therefore this 1 bit is set to zero and 1 is placed in 5th bit position producing a D/A output of 26 V. It is less than analog input. Therefore this 1 bit is retained. Now 1 is placed in LSB producing a D/A output of 27 V which is more than analog input. Therefore LSB is set to zero and the converter gives an output of 26 V.

The successive approximation method of A/D converter is very fast and takes only about 250 ns/ bit.

Performance Characteristics of A/D converters:

The performance characteristics of A/D converters are resolution, accuracy, A/D gain and drift and A/D speed.

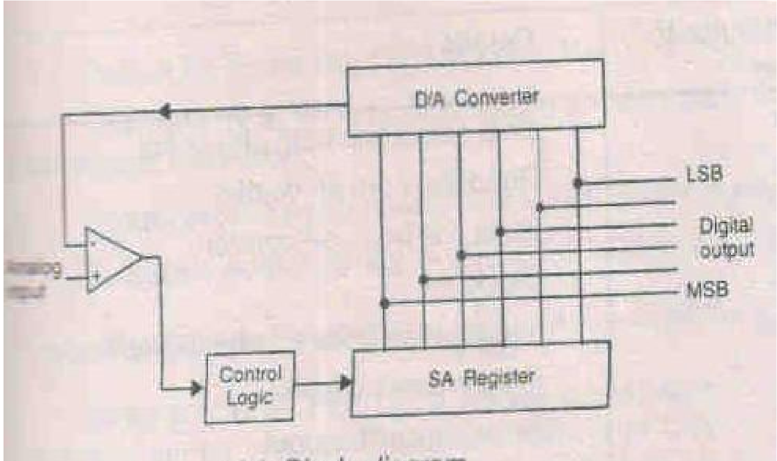
(a) Resolution: A/D rsolution is the change in voltage input necessary for a one bit change in output. It can also be expressed as percent.

(b) A/D Accuracy: The accuracy of A/D conversion is limited by the ±1/2 LSB due to quantisation error and the other errors of the system. It is defined as the maximum deviation of digital output from the ideal linear reference line. Ideally it aproaches ±1/2 LSB.

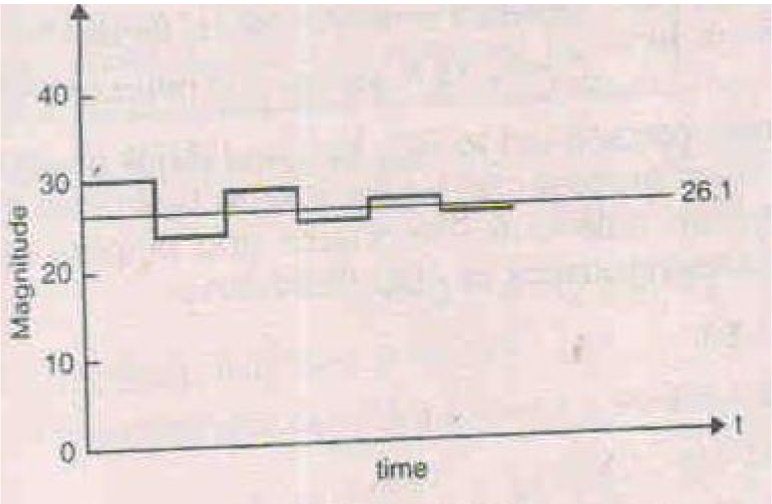
(c) A/D gain and Drift: A/D gain is the voltage output is devided by the voltage input at the linearity reference line. It can usually be zeroed out.

Drift means change in circuit parameters with time. Drift errors of upto ±1/2 LSB will cause a maximum errors of one LSB between the first and the last transition. Very low drift is quite difficult to achieve and increases cost of the device.

(d) A/D speed: It can be defined in two ways, i.e. either the time necessary to do one conversion or the line between successive conversion at the highest rate possible. Speed depends on the settling time of components and the speed of the logic.



(a) Block diagram



(b) Waveforms

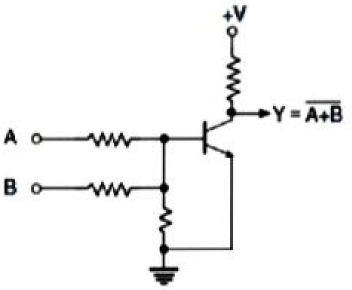
UNIT-6: ***LOGIC FAMILIES***

- A circuit configuration or approach used to produce a type of digital integrated circuit is called Logic Family.
- By using logic families we can generate different logic functions, when fabricated in the form of an IC with the same approach, or in other words belonging to the same logic family, will have identical electrical characteristics.
- The set of digital ICs belonging to the same logic family are electrically compatible with each other.
- Some common Characteristics of the Same Logic Family include Supply voltage range, speed of response, power dissipation, input and output logic levels, current sourcing and sinking capability, fan-out, noise margin, etc.
- Choosing digital ICs from the same logic family guarantees that these ICs are compatible with respect to each other and that the system as a whole performs the intended logic function.

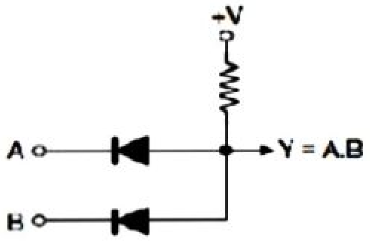
TYPES OF LOGIC FAMILY:-

- The entire range of digital ICs is fabricated using either bipolar devices or MOS devices or a combination of the two.
- Bipolar families include:-

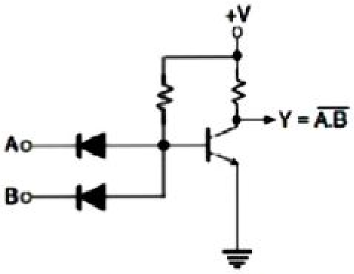
  Diode logic (DL)
  Resistor-Transistor logic (RTL)
  Diode-transistor logic (DTL)
  Transistor- Transistor logic (TTL)
  Emitter Coupled Logic (ECL),
  (also known as Current Mode Logic(CML))
  Integrated Injection logic (I2L)
- The Bi-MOS logic family uses both bipolar and MOS devices.



Resistor -Transistor logic (RTL)



Diode Logic (DL)



Diode-Transistor logic (DTL)

- Above are some example of DL, RTL and DTL.
- MOS families include:-

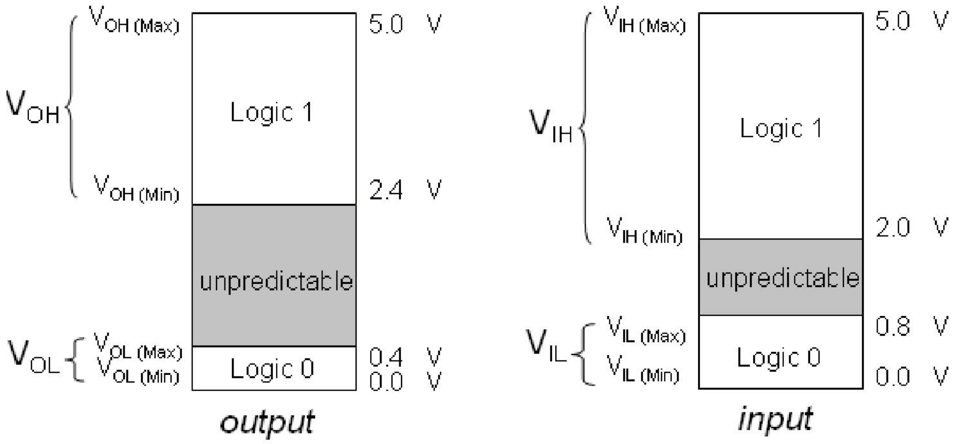
  ThePMOS family (using P-channel MOSFETs)
  The NMOS family (using N-channel MOSFETs)
  The CMOS family (using both N- and P-channel devices)

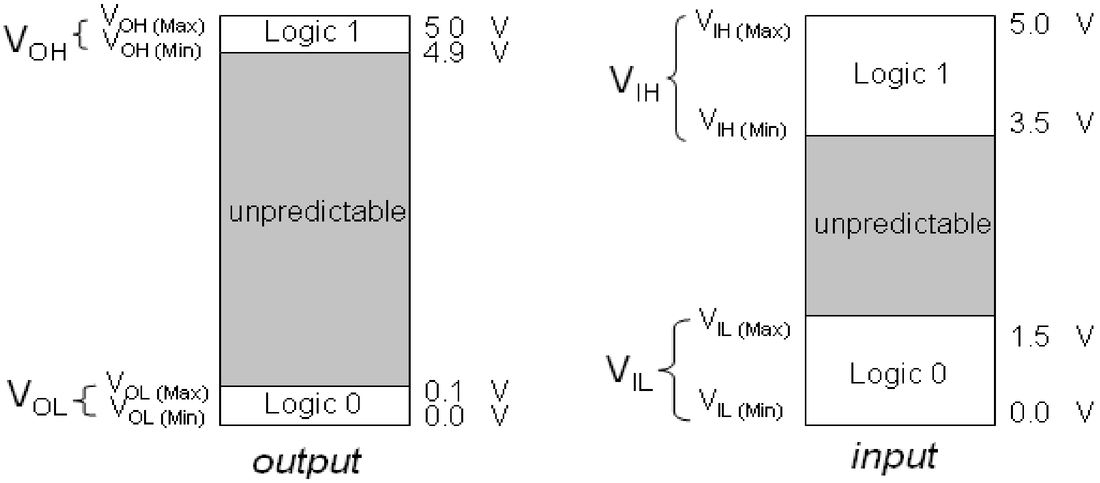
SOME OPERATIONAL PROPERTIES OF LOGIC FAMILY:-

DC Supply Voltage:-

- The nominal value of the dc supply voltage for TTL (transisitor-transistor logic) and CMOS (complementary metal-oxide semiconductor) devices is +5V. Although ommitted from logic diagrams for simplicity, this voltage is connected to Vcc or VDD pin of an IC package and ground is connected to the GND pin.

TTL Logic Levels





CMOS Logic Levels

Noise Immunity:-

- Noise is the unwanted voltage that is induced in electrical circuits and can present a threat to the poor operation of the circuit. In order not to be adversely effected by noise, a logic circuit must have a certain amount of 'noise immunity'.
- This is the ability to tolerate a certain amount of unwanted voltage fluctuation on its inputs without changing its output state is called Noise Immunity.

Noise Margin:-

- A measure of a circuit's noise immunity is called 'noise margin' which is expressed in volts.
- There are two values of noise margin specified for a given logic circuit: the HIGH (V_{NH}) and LOW (V_{NL}) noise margins.

These are defined by following equations :

$V_{NH} = V_{OH}$ (Min) - V_{IH} (Min) $V_{NL} = V_{IL}$ (Max) - V_{OL} (Max)

Power Dissipation:-

- A logic gate draws ICCH current from the supply when the gate is in the HIGH output state, draws ICCL current from the supply in the LOW output state.
- Average power is

  PD = VCC ICC where ICC = (ICCH + ICCL) / 2

Propagation Delay time:-

- When a signal passes (propagates) through a logic circuit, it always experiences a time delay as shown below. A change in the output level always occurs a short time, called 'propagation delay time' , later than the change in the input level that caused it.

Fan Out of Gates:-

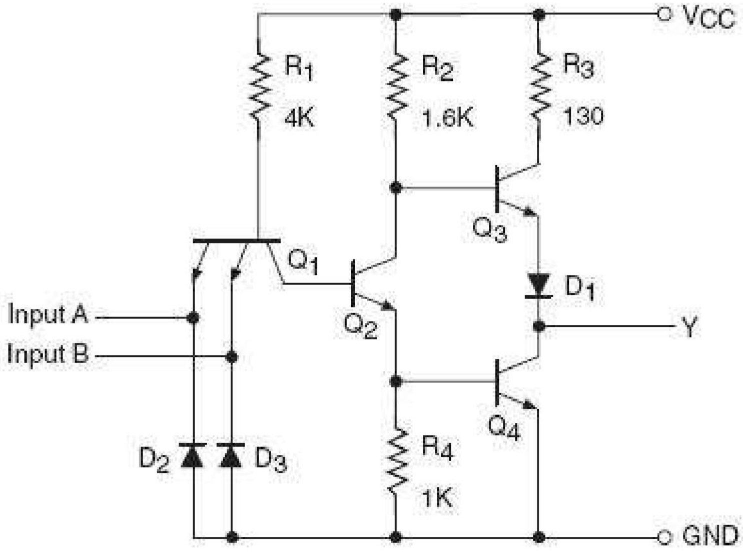
- When the output of a logic gate is connected to one or more inputs of other gates, a load on the driving gate is created. There is a limit to the number of load gates that a given gate can drive. This limit is called the 'Fan-Out' of the gate.

TRANSISTOR-TRANSISTOR LOGIC:-

- In Transistor-Transistor logic or just TTL, logic gates are built only around transistors.
- TTL was developed in 1965. Through the years basic TTL has been improved to meet performance requirements. There are many versions or families of TTL.
- For example
    - Standard TTL
    - High Speed TTL (twice as fast, twice as much power)
    - Low Power TTL (1/10 the speed, 1/10 the power of “standard" TTL)
    - Schhottky TTL etc. (for high-frequency uses)
- All TTL logic families have three configurations for outputs
    1. Totem pole output
    2. Open collector output
    3. Tristate output

Totem pole output:-

- Addition of an active pull up circuit in the output of a gate is called totem pole.
- To increase the switching speed of the gate which is limited due to the parasitic capacitance at the output totem pole is used.
- The circuit of a totem-pole NAND gate is shown below, which has got three stages
    1. Input Stage
    2. Phase Splitter Stage
    3. Output Stage



- Transistor Q1 is a two-emitter NPN transistor, which is equivalent two NPN transistors with their base and emitter terminals tied together.
- The two emitters are the two inputs of the NAND gate

    In TTL technology multiple emitter transistors are used for the input devices

    Diodes D2 and D3 are protection diodes used to limit negative input voltages.
- When there is large negative voltage at input, the diode conducts and shorting it to the ground Q2 provides complementary voltages for the output transistors Q3 and Q4.
- The combination of Q3 and Q4 forms the output circuit often referred to as a totem pole arrangement (Q4 is stacked on top of Q3). In such an arrangement, either Q3 or Q4 conducts at a time depending upon the logic status of the inputs

    Diode D1 ensures that Q4 will turn off when Q2 is on (HIGH input)

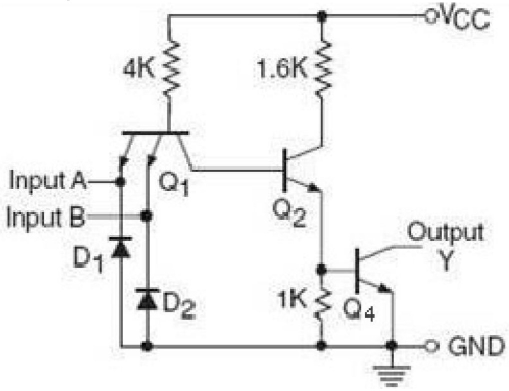
    The output Y is taken from the top of Q3

Advantages of Totem Pole Output:-

- The features of this arrangement are
    1. Low power consumption
    2. Fast switching
    3. Low output impedance

OPEN COLLECTOR OUTPUT:-

- Figure below shows the circuit of a typical TTL gate with open-collector output Observe here that the circuit elements associated with Q3 in the totem-pole circuit are missing and the collector of Q4 is left open-circuited, hence the name open-collector.



- An open-collector output can present a logic LOW output. Since there is no internal path from the output Y to the supply voltage V_{CC} , the circuit cannot present a logic HIGH on its own.

Advantages of Open Collector Outputs:-

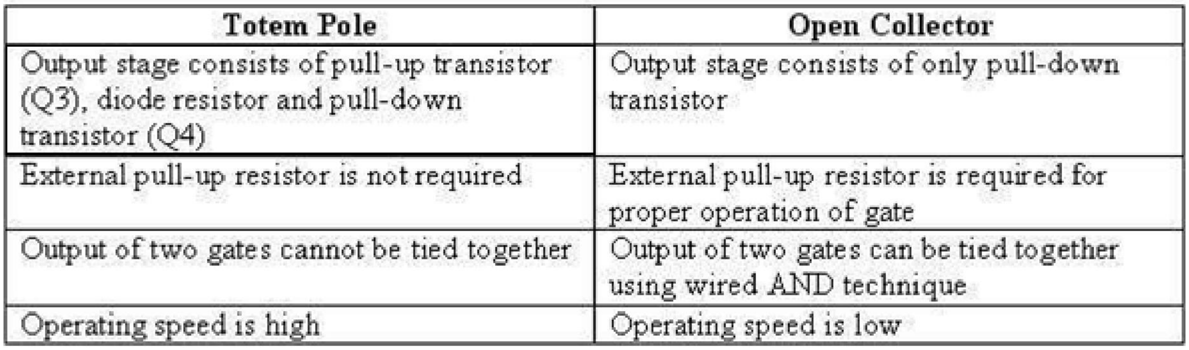
- Open-collector outputs can be tied directly together which results in the logical ANDing of the outputs. Thus the equivalent of an AND gate can be formed by simply connecting the outputs.
- Increased current levels - Standard TTL gates with totem-pole outputs can only provide a HIGH current output of 0.4 mA and a LOW current of 1.6 mA. Many open-collector gates have increased current ratings.
- Different voltage levels - A wide variety of output HIGH voltages can be achieved using open-collector gates. This is useful in interfacing different logic families that have different voltage and current level requirements.

Disadvantage of open-collector gates:-

- They have slow switching speed. This is because the value of pull-up resistor is in kW, which results in a relatively long time Constants

Comparison of Totem Pole and Open Collector Output:-

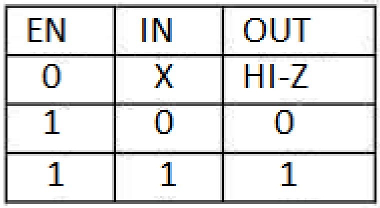
- The major advantage of using a totem-pole connection is that it offers low-output impedance in both the HIGH and LOW output states



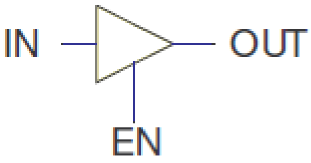
Totem Pole	Open Collector
Output stage consists of pull-up transistor (Q3), diode resistor and pull-down transistor (Q4)	Output stage consists of only pull-down transistor
External pull-up resistor is not required	External pull-up resistor is required for proper operation of gate
Output of two gates cannot be tied together	Output of two gates can be tied together using wired AND technique
Operating speed is high	Operating speed is low

TRISTATE (THREE-STATE) LOGIC OUPUT:-

- Tristate output combines the advantages of the totem-pole and open collector circuits.
- Three output states are HIGH, LOW, and high impedance (Hi-Z).

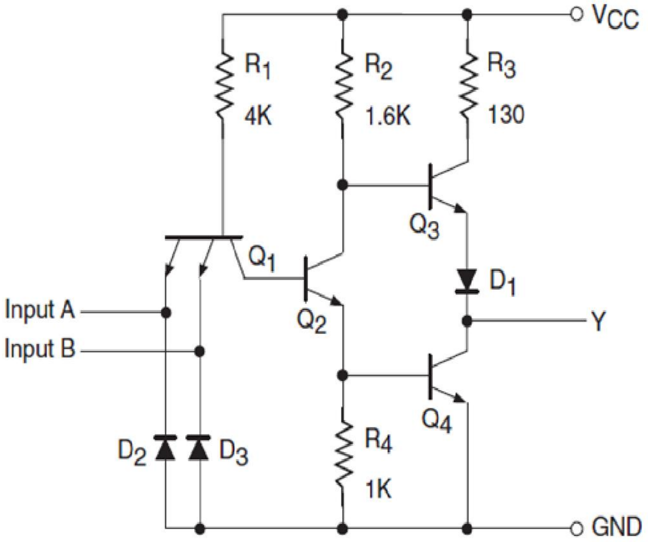


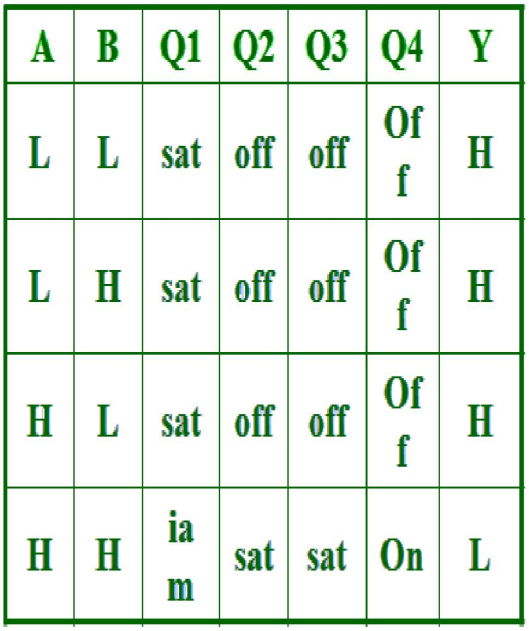
EN	IN	OUT
0	X	HI-Z
1	0	0
1	1	1



- For the symbol and truth table, IN is the data input, and EN, the additional enable input for control. For EN = 0, regardless of the value on IN(denoted by X), the output value is Hi-Z. For EN = 1, the output value follows the input value.
- Data input, IN, can be inverted. Control input, EN, can be inverted by addition of "bubbles" to signals IN OUT EN.
- This requires two inputs: input and enable EN is to make output Hi-Z or follow input.

STANDARD TTL NAND GATE:





A	B	Q1	Q2	Q3	Q4	Y
L	L	sat	off	off	Off	H
L	H	sat	off	off	Off	H
H	L	sat	off	off	Off	H
H	H	iam	sat	sat	On	L

CMOS TECHNOLOGY:-

- MOS stands for Metal Oxide Semiconductor and this technology uses FETs.
- MOS can be classified into three sub-families:

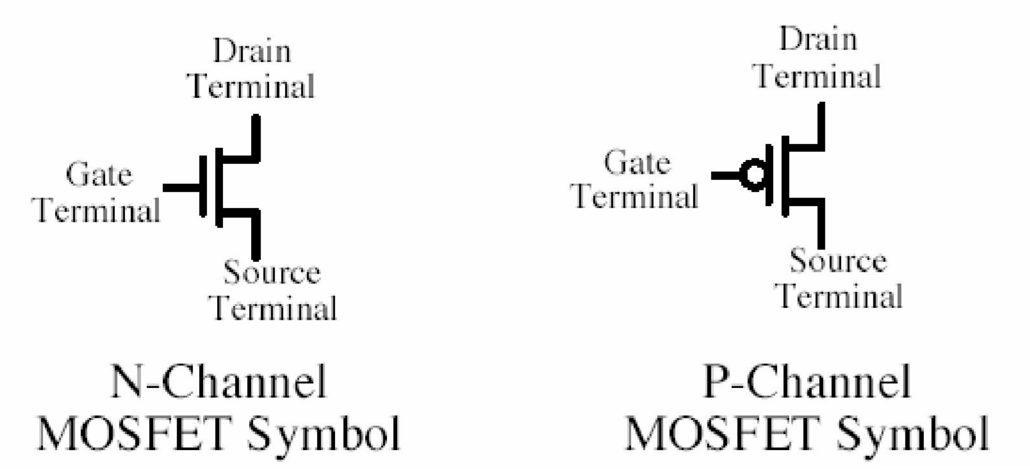
  PMOS (P-channel)
  NMOS (N-channel)
  CMOS (Complementary MOS, most common)
- The following simplified symbols are used to represent MOSFET transistors in most CMOS. The gate of a MOS transistor controls the flow of the current between the drain and the source. The MOS transistor can be viewed as a simple ON/OFF switch.

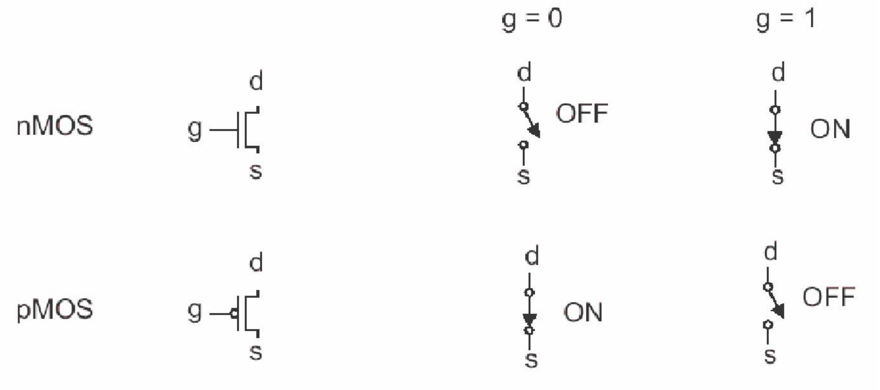
Advantages of MOS Digital ICs:-

- They are simple and inexpensive to fabricate.
- Can be used for Higher integration and consume little power.

Disadvantages of MOS Digital ICs:-

- There is possibility for Static-electricity damage.
- They are slower than TTL.



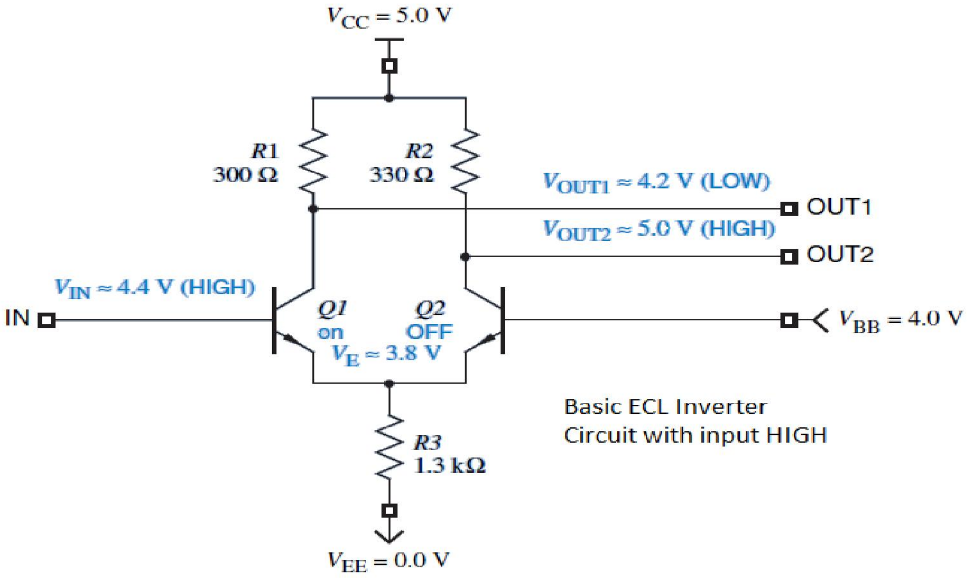


<u>ECL: EMITTER-COUPLED LOGIC:-</u>

- The key to reduce propagation delay in a bipolar logic family is to prevent a gate’s transistors from saturating. It is possible to prevent saturation by using a radically different circuit structure, called current-mode logic (CML) or emitter-coupled logic (ECL).
- Unlike the other logic families in this chapter, ECL does not produce a large voltage swing between the LOW and HIGH levels but it has a small voltage swing, less than a volt, and it internally switches current between two possible paths, depending on the output state.

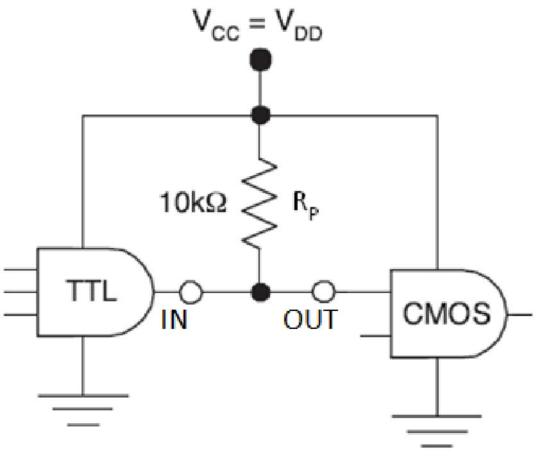
<u>Basic ECL Circuit</u>

- The basic idea of current-mode logic is illustrated by the inverter/buffer circuit in the figure. This circuit has both an inverting output (OUT1) and a non-inverting output (OUT2).
- Two transistors are connected as a differential amplifier with a common emitter resistor.
- The supply voltages for this example are VCC = 5.0, VBB = 4.0, and VEE = 0 V, and the input LOW and HIGH levels are defined to be 3.6 and 4.4 V. This circuit actually produces output LOW and HIGH levels that are 0.6 V higher (4.2 and 5.0 V).

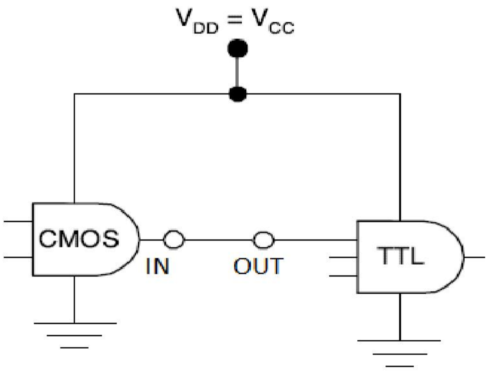


Basic ECL Inverter
Circuit with input HIGH

INTERFACING OF TTL TO CMOS



INTERFACING OF CMOS TO TTL



TTL vs. CMOS:-

- TTL has less propagation delay than CMOS i.e. TTL is good where high speed is needed.
- And CMOS 4000 is good for Battery equipment and where speed is not so important.
- CMOS requires less power than TTL i.e. power dissipation and hence power consumption is less for CMOS.