3rd Semester

INSTRUMENTATION AND CONTROL ENGINEERING

SUBJECT: DIGITAL ELECTRONICS

SUBJECT CODE : 181536

DISCLAIMER

This document does not claim any originality and cannot be used as a substitute for prescribed textbooks. The matter presented here is prepared for their teaching and online assignments by referring the text books/e-books and reference books. Further, this document is not intended to be used for commercial purpose and the committee members are not accountable for any issues, legal or otherwise, arising out of use of this document.

Chapter:- 1

(Introduction)

Distinction b/w analog & digital signal:-

BASIS FOR COMPARISON	ANALOG SIGNAL	DIGITAL SIGNAL
Basic	An analog signal is a continuous wave that changes over a time period.	A digital signal is a discrete wave that carries information in binary form.
Description	An analog signal is described by the amplitude, period or frequency, and phase.	A digital signal is described by bit rate and bit intervals.
Range	Analog signal has no fixed range	Digital signal has a finite numbers i.e. 0 and 1.
Distortion	An analog signal is more prone to distortion.	A digital signal is less prone to distortion.
Transmit	An analog signal transmit data in the form of a wave	A digital signal carries data in the binary form i.e. 0 nad 1.

Advantages of digital signal:-

- The frequency of the carrier wave is usually higher.
- The combined wave is transmitted.
- Carry more information per second than analogue signals.
- Maintain quality over long distances better than analogue signals.
- They're automatic.
- Easier to remove noise.
- Can be very immune to noise.

Application of digital signal:-

- Audio and video compression (the quality depends on the sampling rate chosen - higher sampling rate = higher quality.
- Audio signal processing (example: applying a low pass or bandpass filter to reduce external noise from an audio…
- Image processing (example: using FFT, filtering and inverse FFT in order to remove noise from an image)
- Medical applications (example: applying a histogram equalization to enhance an x-ray image)

Chapter:-2

Number System

A binary system is a system of two astronomical bodies which are close enough that their gravitational attraction causes them to orbit each other around a barycenter (also see animated examples). More restrictive definitions require that this common center of mass is not located within the interior of either object, in order to exclude the typical planet–satellite systems and planetary systems.

The most common binary systems are binary stars and binary asteroid, but brown dwarfs, planets, neutron stars, black holes and galaxies can also form binaries.

A multiple system is like a binary system but consists of three or more objects such as for trinary stars and trinary asteroids.

In a binary system, the brighter object is referred to as primary, and the other the secondary.

They are also classified based on orbit. Wide binaries are objects with orbits that keep them apart from one another. They evolve separately and have very little effect on each other. Close binaries are close to each other and are able to transfer mass from one another.

They can also be classified based on how we observe them. Visual binaries are two stars separated enough that they can be viewed through a telescope or binoculars.

Eclipsing binaries are where the objects' orbits are at an angle that when one passes in front of the other it causes an eclipse, as seen from Earth.

Octal Number System:-

Octal Number System is one the type of Number Representation techniques, in which there value of base is 8. That means there are only 8 symbols or possible digit values, there are 0, 1, 2, 3, 4, 5, 6, 7. It requires only 3 bits to represent value of any digit. Octal numbers are indicated by the addition of either an 0o prefix or an 8 suffix.

Position of every digit has a weight which is a power of 8. Each position in the Octal system is 8 times more significant than the previous position, that means numeric value of an octal number is determined by multiplying each digit of the number by the value of the position in which the digit appears and then adding the products. So, it is also a positional (or weighted) number system.

Representation of Octal Number:-

Each Octal number can be represented using only 3 bits, with each group of bits having a distich values between 000 (for 0) and 111 (for 7 = 4+2+1). The equivalent binary number of Octal number are as given below –

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

Octal number system is similar to Hexadecimal number system. Octal number system provides convenient way of converting large binary numbers into more compact and smaller groups, however this octal number system is less popular.

Example-1 – The number 111 is interpreted as

111 = 1x82+5x81+7x80 = 157

Here, right most bit 7 is the least significant bit (LSB) and left most bit 1 is the most significant bit (MSB).

Hxadecimal To Binary:-

Whereas Hexadecimal number is one of the number systems which has value is 16 and it has only 16 symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and A, B, C, D, E, F. Where A, B, C, D, E and F are single bit representations of decimal value 10, 11, 12, 13, 14 and 15 respectively. Whereas Decimal system is most familiar number system to the general public. It is base 10 which has only 10 symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9.

Conversion from Hexadecimal to Decimal number system

There are various indirect or direct methods to convert a hexadecimal number into decimal number. In an indirect method, you need to convert a hexadecimal number into binary or octal number, then you can convert it into decimal number.

Example – Convert hexadecimal number F1 into decimal number.

First convert it into binary or octal number,

$= (F1)_{16}$

$= (1111\ 0001)_2$ or $(011\ 110\ 001)_2$

Because in binary, value of F and 1 are 1111 and 0001 respectively. Then convert it into decimal number multiplying power of its position of base.

$= (1x2^7+1x2^6+1x2^5+1x2^4+0x2^3+0x2^2+0x2^1+1x2^0)_{10}$

or $(3\ 6\ 1)_8$

$= (1x2^7+1x2^6+1x2^5+1x2^4+0x2^3+0x2^2+0x2^1+1x2^0)_{10}$ or $(3x8^2+6x8^1+1x8^0)10$

$= (241)_{10}$

However, there is a simple direct method to convert a hexadecimal number to decimal number. Since, there are only 16 digits (from 0 to 7 and A to F) in hexadecimal number system, so we can represent any digit of hexadecimal number system using only 4 bit as following below.

Hexa	0	1	2	3	4	5	6	7
Binary	0000	0001	0010	0011	0100	0101	0110	0111
Hexa	8	9	A=10	B=11	C=12	D=13	E=14	F=15
Binary	1000	1001	1010	1011	1100	1101	1110	1111

Hexadecimal number system provides convenient way of converting large binary numbers into more compact and smaller groups. These are weights of hexadecimal of respective position of hexadecimal (value of base is 16).

Most Significant Bit (MSB)	Hexa Point		Least Significant Bit (LSB)		
16^2	16^1	16^0	16^{-1}	16^{-2}	16^{-3}
256	16	1	1/16	1/256	1/4096

Since number numbers are type of positional number system. That means weight of the positions from right to left are as 16^0, 16^1, 16^2, $16^3$and so on. for the integer part and weight of the positions from left to right are as 16^{-1}, 16^{-2}, $16^{-3}$and so on. for the fractional part.

You can directly convert a hexadecimal number into decimal number using reverse method of decimal to hexadecimal number.

Assume any unsigned hexadecimal number is $h_nh_{(n-1)}$ ... $h_1h_0.h_{-1}h_{-2}$ ... $h_{(m-1)}h_m$. Then the decimal number is equal to the sum of hexadecimal digits (hn) times their power of 16 (16n), i.e.,

$= h_nh_{(n-1)}$ ... $h_1h_0.h_{-1}h_{-2}$ ... $h_{(m-1)}h_m$

$= h_nx16^n+h_{(n-1)}x16^{(n-1)}$+ ... $+h_1x16^1+h_0x16^0+h_{-1}x16^{-1}+h_{-2}x16^{-2}$+ ... $+h_{(m-1)}x16^{-(m-1)}+h_{-m}x16^{-m}$

This is simple algorithm where you have to multiply positional value of binary with their digit and get the sum of these steps.

Example-1 – Convert hexadecimal number ABCDEF into decimal number.

Since value of Symbols – A, B, C, D, E, F are 10, 11, 12, 13, 14, 15 respectively. Therefore equivalent decimal number is,

$= (ABCDEF)_{16}$

$= (10x16^5+11x16^4+12x16^3+13x16^2+14x16^1+15x16^0)_{10}$

$= (10485760+720896+49152+3328+224+15)_{10}$

$= (11259375)_{10}$ which is answer.

How to Convert Binary to Hexadecimal:-

Binary is the simplest kind of number system that uses only two digits of 0 and 1 (i.e. value of base 2). Since digital electronics have only these two states (either 0 or 1), so

binary number is most preferred in modern computer engineer, networking and communication specialists, and other professionals.

Whereas **Hexadecimal** number is one of the number systems which has value is 16 and it has only 16 symbols – 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and A, B, C, D, E, F. Where A, B, C, D, E and F are single bit representations of decimal value 10, 11, 12, 13, 14 and 15 respectively.

Conversion from Binary to Hexadecimal number system

Hexadecimal number system provides convenient way of converting large binary numbers into more compact and smaller groups. There are various ways to convert a binary number into hexadecimal number. You can convert using direct methods or indirect methods. First, you need to convert a binary into other base system (e.g., into decimal, or into octal). Then you need to convert it hexadecimal number.

Most Significant Bit (MSB)	Hexa Point		Least Significant Bit (LSB)		
16^2	16^1	16^0	16^{-1}	16^{-2}	16^{-3}
256	16	1	1/16	1/256	1/4096

Since number numbers are type of positional number system. That means weight of the positions from right to left are as 16^0, 16^1, 16^2, $16^3$and so on. for the integer part and weight of the positions from left to right are as 16^{-1}, 16^{-2}, $16^{-3}$and so on. for the fractional part.

Example – Convert binary number 1101010 into hexadecimal number.

First convert this into decimal number:

$= (1101010)_2$

$= 1x2^6+1x2^5+0x2^4+1x2^3+0x2^2+1x2^1+0x2^0$

$= 64+32+0+8+0+2+0$

$= (106)_{10}$

Then, convert it into hexadecimal number

$= (106)_{10}$

$= 6x16^1+10x16^0$

$= (6A)_{16}$ which is answer.

What is Binary Addition:-

The binary addition operation works similarly to the base 10 decimal system, except that it is a base 2 system. The binary system consists of only two digits, 1 and 0. Most of the functionalities of the computer system use the binary number system. The binary

code uses the digits 1's and 0's to make certain processes turn off or on. The process of the addition operation is very familiar to the decimal system by adjusting to the base 2. Before attempting the binary addition process, we should have complete knowledge of how the place works in the binary number system. Because most of the modern digital computers and electronic circuits perform the binary operation by representing each bit as a voltage signal. The bit 0 represents the "OFF" state, and the bit 1 represents the "ON" state.

Rules of Binary Addition

Binary addition is much easier than the decimal addition when you remember the following tricks or rules. Using these rules, any binary number can be easily added. The four rules of binary addition are:

- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 0 = 1
- 1 + 1 =10

How To Do Binary Addition?

Now, look at the example of the binary addition:101 + 101

Procedure for Binary Addition of Numbers:

```
  101
(+) 101
```

- **Step 1:** First consider the 1's column, and add the one's column,(1+1) and it gives the result 10 as per the condition of binary addition.
- **Step 2:** Now, leave the 0 in the one's column and carry the value 1 to the 10's column.

```
   1
  101
(+) 101
_______
    0
```

- **Step 3: Now add 10's place, 1+(0 + 0) = 1.** So, nothing carries to the 100's place and leave the value 1 in the 10's place

```
  1
 101
(+) 101
--------
  10
```

In binary addition using 1's complement:-

A. Addition of a positive and a negative binary number

We discuss the following cases under this.

Case I: When the positive number has greater magnitude.

In this case addition of numbers is performed after taking 1's complement of the negative number and the end-around carry of the sum is added to the least significant bit.

complement: The following examples will illustrate this method in binary addition using 1's

1. Find the sum of the following binary numbers:

(i) + 1110 and - 1101

Solution:

```
+ 1 1 1 0   ⇒   0 1 1 1 0
- 1 1 0 1   ⇒   1 0 0 1 0   (taking 1's complement)
               0 0 0 0 0
                     1   carry

               0 0 0 0 1
```

Hence the required sum is + 0001.

(ii) + 1101 and - 1011

(Assume that the representation is in a signed 5-bit register).

Solution:

+ 1 1 0 1 ⇒ 0 1 1 0 1

- 1 0 1 1 ⇒ 1 0 1 0 0 (taking 1's complement)

0 0 0 0 1

1 carry

0 0 0 1 0

Hence the required sum is + 0010.

Case II: When the negative number has greater magnitude.

In this case the addition is carried in the same way as in case 1 but there will be non end-around carry. The sum is obtained by taking 1's complement of the magnitude bits of the result and it will be negative.

The following examples will illustrate this method in binary addition using 1's complement:

Find the sum of the following binary numbers represented in a sign-plus-magnitude 5-bit register:

(i) + 1010 and - 1100

Solution:

+ 1 0 1 0 ⇒ 0 1 0 1 0

- 1 1 0 0 ⇒ 1 0 0 1 1 (1's complement)

1 1 1 0 1

Hence the required sum is – 0010.

(ii) + 0011 and - 1101.

Solution:

+ 0 0 1 1 ⇒ 0 0 0 1 1

- 1 1 0 1 ⇒ 1 0 0 1 0 (1's complement)

1 0 1 0 1

Hence the required sum is – 1010.

B. When the two numbers are negative

For the addition of two negative numbers 1's complements of both the numbers are to be taken and then added. In this case an end-around carry will always appear. This along with a carry from the MSB (i.e. the 4th bit in the case of sign-plus-magnitude 5-bit register) will generate a 1 in the sign bit. 1's complement of the magnitude bits of the result of addition will give the final sum.

The following examples will illustrate this method in binary addition using 1's complement:

Find the sum of the following negative numbers represented in a sign-plus-magnitude 5-bit register:
(i) -1010 and -0101

Solution:

- 1 0 1 0 ⇒ 1 0 1 0 1 (1's complement)
- 0 1 0 1 ⇒ 1 1 0 1 0 (1's complement)

0 1 1 1 1
1 carry
1 0 0 0 0

1's complement of the magnitude bits of sum is 1111 and the sign bit is 1.

Hence the required sum is -1111.

(ii) -0110 and -0111.

Solution:

- 0 1 1 0 ⇒ 1 1 0 0 1 (1's complement)
- 0 1 1 1 ⇒ 1 1 0 0 0 (1's complement)

1 0 0 0 1
1 carry
1 0 0 1 0

1's complement of 0010 is 1101 and the sign bit is 1.

Hence the required sum is - 1101.

Chapter:-3

(Codes & parity)

What Is Weighted & Non-Weighted Coding:-

The digital circuitry you see inside computers and other electronic devices can only communicate via two concepts: on and off. These concepts are represented to us in the form of binary numbering, where 0 is off and 1 is on. To truly communicate with a computer, further conversions are necessary to bring the computer language to a more human format. The first step in the conversion process is to convert binary coding into a more readable decimal system. Weighted and non-weighted coding refers to the method in which binary numbers are converted to decimal. With weighted coding, each digit in a number is assigned a weighted value before the conversion. Non-weighted coding methods use slightly varied formulas but perform the conversions without the weight value.

Numbering Systems:-

Numbering systems are indicated by a base, which is the highest number you can count to before having to add another digit. For example, the numbering system we all learn as children is called base 10, because the first ten numbers in the sequence, 0 through 9, can be counted using single digits. Once you get to 10, you have to shift everything over and count in two-digit numbers until you get to 100, and then you count in three-digit numbers. This base 10 system is also called the decimal system.

Positional Notation

Positional notation occurs when you assign a positional value to each digit in a real number, from right to left. For the number 4782, for example, starting with the 2 and counting from right to left, the positions are 0, 1, 2, 3 as in the following:

4782 = number 3210 = positional values

Weighted Coding

In the example above, the positional assignments 0 through 3 can be the weighted values of their assigned digits. So the weight of the 4 is 3 and the weight of the 7 is 2. The weight of a number comes into play when converting from any base numbering system to the decimal (base 10) numbering system. One formula for converting a weighted number is to multiply each digit by its base to the power of its position, and then add all the resulting digits. In the example below, 100101, which is a binary base 2 number, is converted to a decimal (base 10) number.

100101 = Binary (base 2) number 543210 = positional weights (1 x 2^5) + (0 x 2^4) + (0 x 2^3) + (1 x 2^2) + (0 x 2^1) + (1 x 2^0) = 32 + 0 + 0 + 4 + 0 + 1 = 37 37base10 = decimal conversion

Other weighted methods include BCD and 2421, each of which uses a similar formula to assign weights and convert to decimal.

Non-Weighted Coding

Gray code is a non-weighted coding method that alters just one bit in a binary number when moving from one decimal number to the next. In normal binary coding, the digits 10 would represent the decimal number 2. When using gray code, one bit of that binary number changes so the decimal number 2 is represented by the binary digits 0011. Sequentially, the decimal number 3, which would normally be represented by the binary digits 0011, is now converted to 0010, because only the one bit can change. Excess-3 is another non-weighted coding method and was once used in older computers and adding machines. With excess-3, you add 3 to a decimal number before converting it to binary. So the decimal number 2, for example, would first increase by 3, making it 5. The binary conversion of 2 using the Excess-3 method would be 0101 instead of its normal binary value of 0010.

Binary Coded Decimal (BCD) code:-

In this code each decimal digit is represented by a 4-bit binary number. BCD is a way to express each of the decimal digits with a binary code. In the BCD, with four bits we can represent sixteen numbers (0000 to 1111). But in BCD code only first ten of these are used (0000 to 1001). The remaining six code combinations i.e. 1010 to 1111 are invalid in BCD.

Decimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Advantages of BCD Codes:-

- It is very similar to decimal system.
- We need to remember binary equivalent of decimal numbers 0 to 9 only.

Disadvantages of BCD Codes:-

- The addition and subtraction of BCD have different rules.
- The BCD arithmetic is little more complicated.
- BCD needs more number of bits than binary to represent the decimal number. So BCD is less efficient than binary.

Gray Code:-

It is the non-weighted code and it is not arithmetic codes. That means there are no specific weights assigned to the bit position. It has a very special feature that, only one bit will change each time the decimal number is incremented as shown in fig. As only one bit changes at a time, the gray code is called as a unit distance code. The gray code is a cyclic code. Gray code cannot be used for arithmetic operation.

Decimal	BCD	Gray
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1

Application of Gray code:-

- Gray code is popularly used in the shaft position encoders.
- A shaft position encoder produces a code word which represents the angular position of the shaft.

Parity:-

In mathematics, parity refers to the evenness or oddness of an integer, which for a binary number is determined only by the least significant bit. In telecommunications and computing, parity refers to the evenness or oddness of the number of bits with value one within a given set of bits, and is thus determined by the value of all the bits. It

can be calculated via a XOR sum of the bits, yielding 0 for even parity and 1 for odd parity. This property of being dependent upon all the bits and changing value if any one bit changes allows for its use in error detection schemes.

Error detection:-

If an odd number of bits (including the parity bit) are transmitted incorrectly, the parity bit will be incorrect, thus indicating that a **parity error** occurred in the transmission. The parity bit is only suitable for detecting errors; it cannot correct any errors, as there is no way to determine which particular bit is corrupted. The data must be discarded entirely, and re-transmitted from scratch. On a noisy transmission medium, successful transmission can therefore take a long time, or even never occur. However, parity has the advantage that it uses only a single bit and requires only a number of XOR gates to generate. See Hamming code for an example of an error-correcting code.

Parity bit checking is used occasionally for transmitting ASCII characters, which have 7 bits, leaving the 8th bit as a parity bit.

For example, the parity bit can be computed as follows. Assume Alice and Bob are communicating and Alice wants to send Bob the simple 4-bit message 1001.

Type of bit parity	Successful transmission scenario
Even parity	Alice wants to transmit: 1001 Alice computes parity bit value: 1+0+0+1 (mod 2) = 0 Alice adds parity bit and sends: 10010 Bob receives: 10010 Bob computes parity: 1+0+0+1+0 (mod 2) = 0 Bob reports correct transmission after observing expected even result.
Odd parity	Alice wants to transmit: 1001 Alice computes parity bit value: 1+0+0+1 (mod 2) = 0 Alice adds parity bit and sends: 1001**1** Bob receives: 10011 Bob computes overall parity: 1+0+0+1+1 (mod 2) = 1 Bob reports correct transmission after observing expected odd result.

This mechanism enables the detection of single bit errors, because if one bit gets flipped due to line noise, there will be an incorrect number of ones in the received data. In the two examples above, Bob's calculated parity value matches the parity bit in its received value, indicating there are no single bit errors. Consider the following example with a transmission error in the second bit using XOR:

Type of bit parity error	Failed transmission scenario
Even parity Error in the second bit	Alice wants to transmit: 1001 Alice computes parity bit value: 1^0^0^1 = 0 Alice adds parity bit and sends: 10010 **...TRANSMISSION ERROR...** Bob receives: 1**1**010 Bob computes overall parity: 1^1^0^1^0 = 1 Bob reports incorrect transmission after observing unexpected odd result.
Even parity Error in the parity bit	Alice wants to transmit: 1001 Alice computes even parity value: 1^0^0^1 = 0 Alice sends: 10010 **...TRANSMISSION ERROR...** Bob receives: 1001**1** Bob computes overall parity: 1^0^0^1^1 = 1 Bob reports incorrect transmission after observing unexpected odd result.

There is a limitation to parity schemes. A parity bit is only guaranteed to detect an odd number of bit errors. If an even number of bits have errors, the parity bit records the correct number of ones, even though the data is corrupt. (See also error detection and correction.) Consider the same example as before with an even number of corrupted bits:

Type of bit parity error	Failed transmission scenario
Even parity Two corrupted bits	Alice wants to transmit: 1001 Alice computes even parity value: 1^0^0^1 = 0 Alice sends: 10010 **...TRANSMISSION ERROR...** Bob receives: 1**1**01**1** Bob computes overall parity: 1^1^0^1^1 = 0 Bob reports correct transmission though actually incorrect.

Chapter 4:-

(Logic gates families)

logic gate (AND, OR, XOR, NOT, NAND, NOR and XNOR)

HYPERLINK "https://www.linkedin.com/in/margaretrouse"

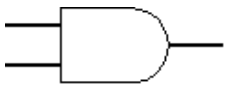
A logic gate is a building block of a digital circuit. Most logic gates have two inputs and one output and are based on Boolean algebra. At any given moment, every terminal is in one of the two binary conditions *false* (high) or *true* (low). False represents 0, and true represents 1. Depending on the type of logic gate being used and the combination of inputs, the binary output will differ. A logic gate can be thought of like a light switch, wherein one position the output is off—0, and in another, it is on—1. Logic gates are commonly used in integrated circuits (IC).

Basic logic gates

There are seven basic logic gates: AND, OR, XOR, NOT, NAND, NOR, and XNOR.

AND | OR | XOR | NOT | NAND | NOR | XNOR

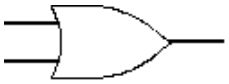
The *AND gate* is so named because, if 0 is called "false" and 1 is called "true," the gate acts in the same way as the logical "and" operator. The following illustration and table show the circuit symbol and logic combinations for an AND gate. (In the symbol, the input terminals are at left and the output terminal is at right.) The output is "true" when both inputs are "true." Otherwise, the output is "false." In other words, the output is 1 only when both inputs one AND two are 1.



AND gate

Input 1	Input 2	Output
	1	
1		
1	1	1

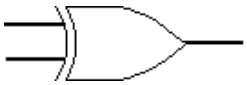
The *OR gate* gets its name from the fact that it behaves after the fashion of the logical inclusive "or." The output is "true" if either or both of the inputs are "true." If both inputs are "false," then the output is "false." In other words, for the output to be 1, at least input one OR two must be 1.



OR gate

Input 1	Input 2	Output
	1	1
1		1
1	1	1

The *XOR* (*exclusive-OR*) *gate* acts in the same way as the logical "either/or." The output is "true" if either, but not both, of the inputs are "true." The output is "false" if both inputs are "false" or if both inputs are "true." Another way of looking at this circuit is to observe that the output is 1 if the inputs are different, but 0 if the inputs are the same.



XOR gate

Input 1	Input 2	Output
	1	1
1		1
1	1	

A logical *inverter*, sometimes called a *NOT gate* to differentiate it from other types of electronic inverter devices, has only one input. It reverses the logic state. If the input is 1, then the output is 0. If the input is 0, then the output is**:**

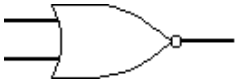
Input	Output
1	
	1

The *NAND gate* operates as an AND gate followed by a NOT gate. It acts in the manner of the logical operation "and" followed by negation. The output is "false" if both inputs are "true." Otherwise, the output is "true."

NAND gate

Input 1	Input 2	Output
		1
	1	1
1		1
1	1	

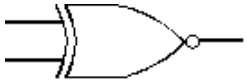
The *NOR gate* is a combination OR gate followed by an inverter. Its output is "true" if both inputs are "false." Otherwise, the output is "false."



NOR gate

Input 1	Input 2	Output
		1
	1	
1		
1	1	

The *XNOR (exclusive-NOR) gate* is a combination XOR gate followed by an inverter. Its output is "true" if the inputs are the same, and "false" if the inputs are different.



XNOR gate

Input 1	Input 2	Output
		1
	1	
1		
1	1	1

Using combinations of logic gates, complex operations can be performed. In theory, there is no limit to the number of gates that can be arrayed together in a single device. But in practice, there is a limit to the number of gates that can be packed into a given physical space. Arrays of logic gates are found in digital ICs. As IC technology advances, the required physical volume for each individual logic gate decreases and digital devices of the same or smaller size become capable of performing ever-more-complicated operations at ever-increasing speeds.

Composition of logic gates:-

High or low binary conditions are represented by different voltage levels. The logic state of a terminal can, and generally does, change often as the circuit processes data. In most logic gates, the low state is approximately zero volts (0 V), while the high state is approximately five volts positive (+5 V).

Logic gates can be made of resistors and transistors, or diodes. A resistor can commonly be used as a pull-up or pull-down resistor. Pull-up or pull-down resistors are used when there are any unused logic gate inputs to connect to either a logic level 1 or 0 respectively. This prevents any false switching of the gate. Pull-up resistors are connected to Vcc (+5V), and pull-down resistors are connected to ground (0 V).

Commonly used logic gates are TTL and CMOS. TTL, or Transistor-Transistor Logic, ICs will use NPN and PNP type Bipolar Junction Transistors. CMOS, or Complementary Metal-Oxide-Silicon, ICs are constructed from MOSFET or JFET type Field Effect Transistors. TTL IC's may commonly be labeled as the 7400 series of chips, while CMOS ICs may often be marked as a 4000 series of chips.

Parameter	**TTL**	**CMOS**
Basic gate	NAND	INVERTER
Circuit	Transistor-Transistor	Complementary MOS
Speed of operation	More	Less
Propagation delay per gate	4 - 12 ns	50 ns
Nominal supply voltage	5 V	3 to 15 V
Wired collector capability	With passive pull up	With tristate output
Fan-out	10	50
Noise immunity	Good	Very good to excellent
Compatibility with other families	With DTL	No, but compatible with TTL for 5 V supply
Available functions	Very high	High

Chapter:-5

Logic simplification

DeMorgan's Theorem-

DeMorgan's Theorem is mainly used to solve the various Boolean algebra expressions.The Demorgan's theorem defines the uniformity between the gate with same inverted input and output. It is used for implementing the basic gate operation likes NAND gate and NOR gate. The Demorgan's theorem mostly used in digital programming and for making digital circuit diagrams. There are two DeMorgan's Theorems. They are described below in detail.

DeMorgan's First Theorem

According to DeMorgan's first theorem, a NOR gate is equivalent to a bubbled AND gate. The Boolean expressions for the bubbled AND gate can be expressed by the equation shown below. For NOR gate, the equation is

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

For the bubbled AND gate the equation is

DeMorgan's Theorem-

DeMorgan's Theorem is mainly used to solve the various Boolean algebra expressions.The Demorgan's theorem defines the uniformity between the gate with same inverted input and output. It is used for implementing the basic gate operation likes NAND gate and NOR gate. The Demorgan's theorem mostly used in digital programming and for making digital circuit diagrams. There are two DeMorgan's Theorems. They are described below in detail.

DeMorgan's First Theorem

According to DeMorgan's first theorem, a NOR gate is equivalent to a bubbled AND gate. The Boolean expressions for the bubbled AND gate can be expressed by the equation shown below. For NOR gate, the equation is

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

For the bubbled AND gate the equation is

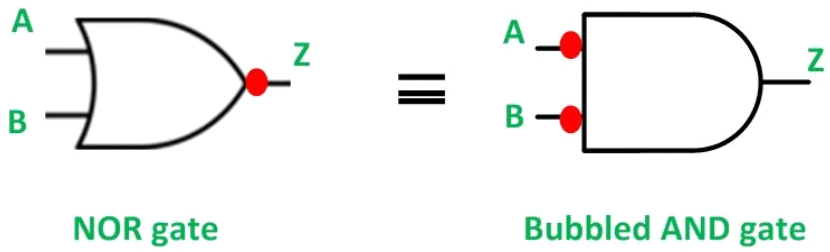
$$Z = \overline{A}.\overline{B}$$

As the NOR and bubbled gates are interchangeable, i.e., both gates have exactly identical outputs for the same set of inputs.

Therefore, the equation can be written as shown below.

$$\overline{A + B} = \overline{A}.\overline{B} \ldots\ldots\ldots\ldots(1)$$

This equation (1) or identity shown above is known as DeMorgan's Theorem. The symbolic representation of the theorem is shown in the figure below.



DeMorgan's Second Theorem

DeMorgan's Second Theorem states that the NAND gate is equivalent to a bubbled OR gate.

The Boolean expression for the NAND gate is given by the equation shown below.

$$Z = \overline{A.B}$$

The Boolean expression for the bubbled OR gate is given by the equation shown below.

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

Since NAND and bubbled OR gates are interchangeable, i.e., both gates have identical outputs for the same set of inputs. Therefore, th The **74181** is a 4-bit slice arithmetic logic unit (ALU), implemented as a 7400 series TTL integrated circuit. The first

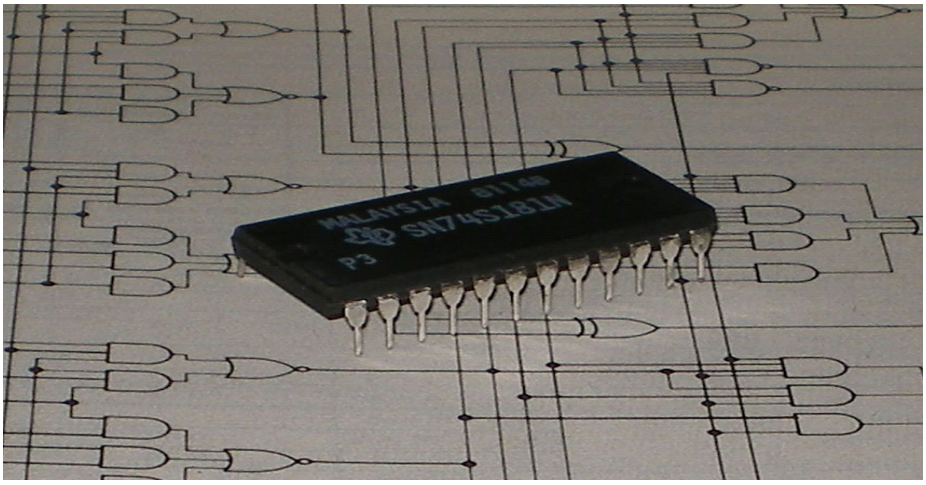
complete ALU on a single chip,[1] it was used as the arithmetic/logic core in the CPUs of many historically significant minicomputers and other devices.

The 74181 represents an evolutionary step between the CPUs of the 1960s, which were constructed using discrete logic gates, and today's single-chip CPUs or microprocessors. Although no longer used in commercial products, the 74181 is still referenced in computer organization textbooks and technical papers. It is also sometimes used in 'hands-on' college courses, to train future computer architects.

Significance:-

Although the 74181 is only an ALU and not a complete microprocessor, it greatly simplified the development and manufacture of computers and other devices that required high speed computation during the late 1960s through the early 1980s, and is still referenced as a "classic" ALU design.

Prior to the introduction of the 74181, computer CPUs occupied multiple circuit boards and even very simple computers could fill multiple cabinets. The 74181 allowed an entire CPU and in some cases, an entire computer to be constructed on a single large printed circuit board. The 74181 occupies a historically significant stage between older CPUs based on discrete logic functions spread over multiple circuit boards and modern microprocessors that incorporate all CPU functions in a single component. The 74181 was used in various minicomputers and other devices beginning in the 1970s, but as microprocessors became more powerful the practice of building a CPU from discrete components fell out of favor and the 74181 was not used in any new designs.

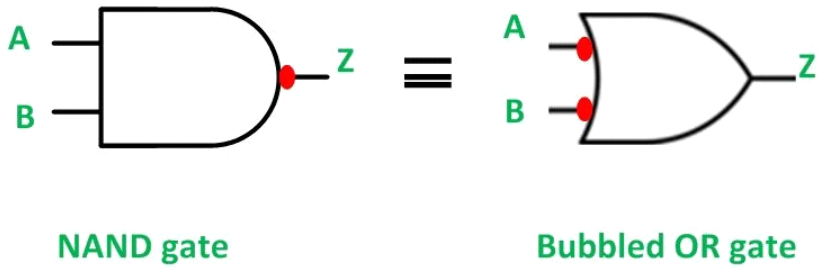


e equations become as given below.

$$\overline{A \cdot B} = \overline{A} + \overline{B} \ \ldots\ldots\ldots (2)$$

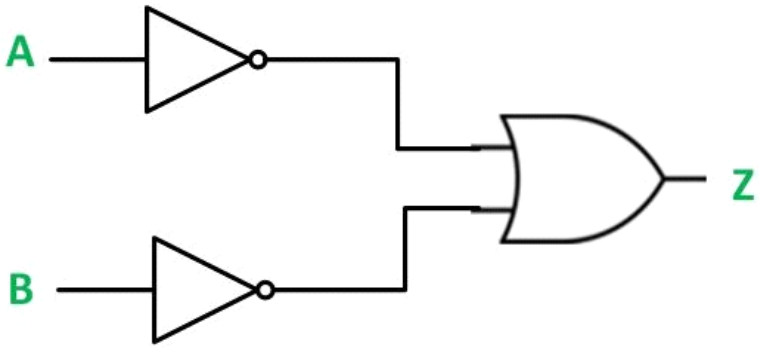
This identity or equation (2) shown above is known as DeMorgan's Second Theorem.

The symbolic representation of the theorem is shown in the figure below.



Circuit Globe The Bubbled OR Gate

The logic circuit of the bubbled OR gate is shown below.



The **truth table** for bubbled OR gate is shown below.

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

In this, both the inputs are inverted before they are applied to an OR gate. The output of a bubbled OR gate can be derived from its logic circuit and can be expressed by the equation shown below.

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

Here are the results when the logic circuit of bubbled OR gate when all the possible sets of inputs are applied such as 00, 01, 10 or 11.

For AB: 00

$$Z = \overline{0} + \overline{0} = 1 + 1 = 1$$

For AB: 01

$$Z = \overline{0} + \overline{1} = 1 + 0 = 1$$

For AB: 10

$$Z = \overline{1} + \overline{0} = 0 + 1 = 1$$

For AB: 11

$$Z = \overline{1} + \overline{1} = 0 + 0 = 0$$

The truth table for the bubbled AND gate is exactly identical to the truth table of a NAND gate. Hence, NAND and bubbled OR gate is interchangeable.

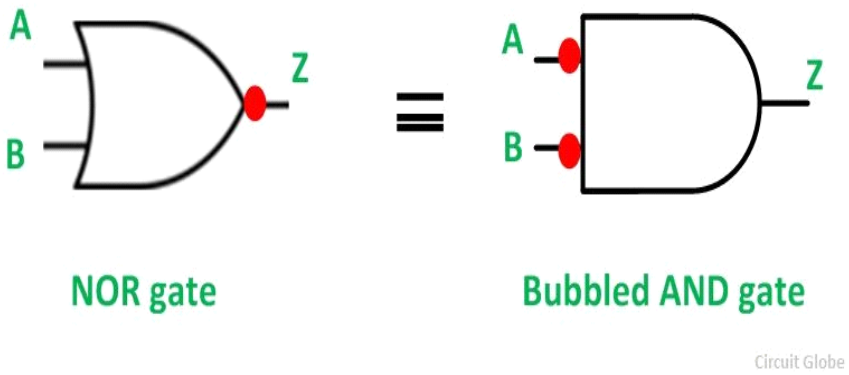
$$Z = \overline{A} . \overline{B}$$

As the NOR and bubbled gates are interchangeable, i.e., both gates have exactly identical outputs for the same set of inputs.

Therefore, the equation can be written as shown below.

$$\overline{A + B} = \overline{A} . \overline{B} \quad \text{.....(1)}$$

This equation (1) or identity shown above is known as DeMorgan's Theorem. The symbolic representation of the theorem is shown in the figure below.



DeMorgan's Second Theorem

DeMorgan's Second Theorem states that the NAND gate is equivalent to a bubbled OR gate.

The Boolean expression for the NAND gate is given by the equation shown below.

$$Z = \overline{A \cdot B}$$

The Boolean expression for the bubbled OR gate is given by the equation shown below.

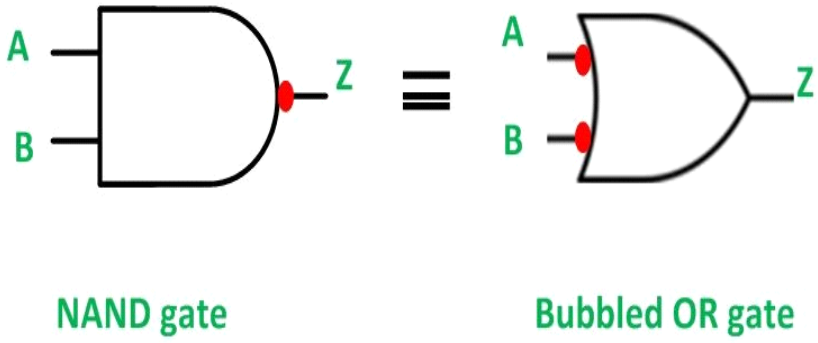
$$Z = \overline{A} + \overline{B}$$

Since NAND and bubbled OR gates are interchangeable, i.e., both gates have identical outputs for the same set of inputs. Therefore, the equations become as given below.

$$\overline{A \cdot B} = \overline{A} + \overline{B} \ \ldots\ldots\ldots (2)$$

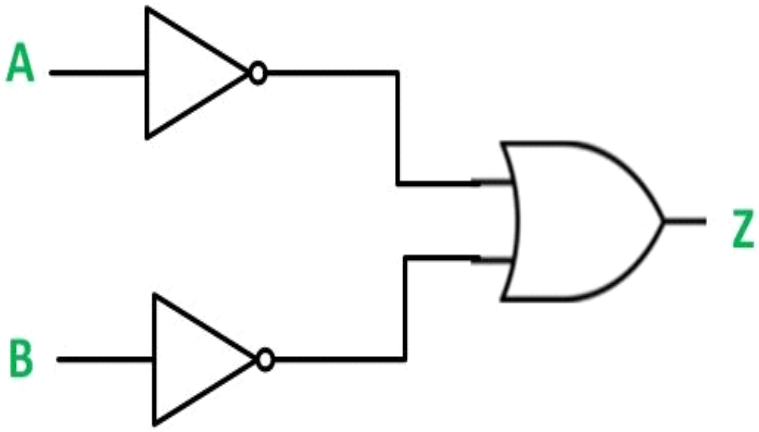
This identity or equation (2) shown above is known as DeMorgan's Second Theorem.

The symbolic representation of the theorem is shown in the figure below.



The Bubbled OR Gate

The logic circuit of the bubbled OR gate is shown below.



The **truth table** for bubbled OR gate is shown below.

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

In this, both the inputs are inverted before they are applied to an OR gate. The output of a bubbled OR gate can be derived from its logic circuit and can be expressed by the equation shown below.

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

Here are the results when the logic circuit of bubbled OR gate when all the possible sets of inputs are applied such as 00, 01, 10 or 11.

For AB: 00

$$Z = \overline{0} + \overline{0} = 1 + 1 = 1$$

For AB: 01

$$Z = \overline{0} + \overline{1} = 1 + 0 = 1$$

For AB: 10

$$Z = \overline{1} + \overline{0} = 0 + 1 = 1$$

For AB: 11

$$Z = \overline{1} + \overline{1} = 0 + 0 = 0$$

The truth table for the bubbled AND gate is exactly identical to the truth table of a NAND gate. Hence, NAND and bubbled OR gate is interchangeable.

K-MAP:-

Karnaugh Mapping

Up to this point we have considered logic reduction problems where the input conditions were completely specified. That is, a 3-variable truth table or Karnaugh map had $2^n = 2^3$ or 8-entries, a full table or map.

It is not always necessary to fill in the complete truth table for some real-world problems. We may have a choice to not fill in the complete table.

For example, when dealing with BCD (Binary Coded Decimal) numbers encoded as four bits, we may not care about any codes above the BCD range of (0, 1, 2…9). The 4-bit binary codes for the hexadecimal numbers (Ah, Bh, Ch, Eh, Fh) are not valid BCD codes.

Thus, we do not have to fill in those codes at the end of a truth table, or K-map, if we do not care to.

We would not normally care to fill in those codes because those codes (1010, 1011, 1100, 1101, 1110, 1111) will never exist as long as we are dealing only with BCD encoded numbers. These six invalid codes are *don't cares* as far as we are concerned.

That is, we do not care what output our logic circuit produces for these don't cares.

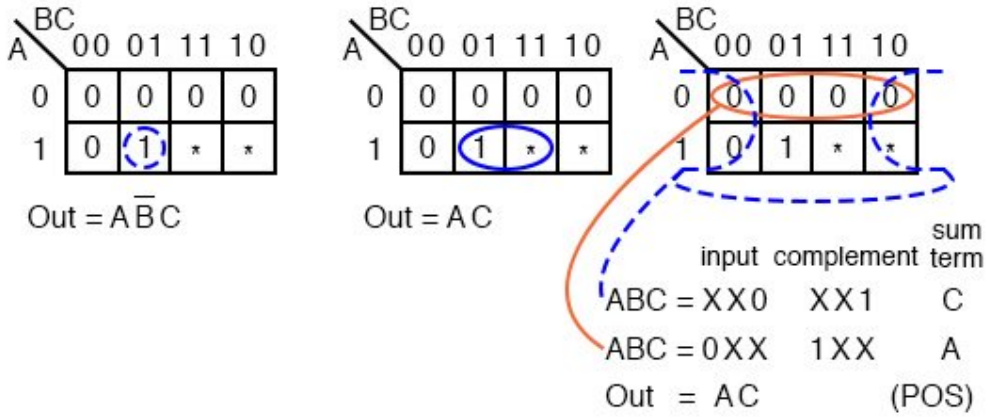
Don't Cares

Don't cares in a Karnaugh map, or truth table, may be either **1**s or **0**s, as long as we don't care what the output is for an input condition we never expect to see. We plot these cells with an asterisk, *, among the normal **1**s and **0**s.

When forming groups of cells, treat the don't care cell as either a **1** or a **0**, or ignore the don't cares.

This is helpful if it allows us to form a larger group than would otherwise be possible without the don't cares. There is no requirement to group all or any of the don't cares.

Only use them in a group if it simplifies the logic.



Above is an example of a logic function where the desired output is **1** for input **ABC = 101** over the range from **000 to 101**. We do not care what the output is for the other possible inputs (**110, 111)**. Map those two as don't cares. We show two solutions.

The solution on the right Out = AB'C is the more complex solution since we did not use the don't care cells. The solution in the middle, Out=AC, is less complex because we grouped a don't care cell with the single **1** to form a group of two.

The third solution, a Product-Of-Sums on the right, results from grouping a don't care with three zeros forming a group of four **0**s. This is the same, less complex, **Out=AC**.

We have illustrated that the don't care cells may be used as either **1**s or **0**s, whichever is useful.

Chapter:-6

(Airthamtic circuits)

Definition of Half Adder:-

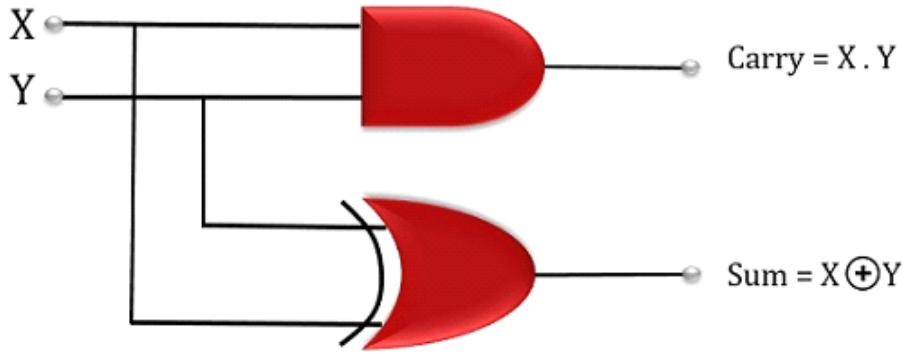
An arithmetic circuit that carries out summation of 2 inputs of one-bit is known as a half adder.

Everyone is familiar with basic addition technique in which when two bits are required to be added then, the addition begins with the rightmost column. As there is always a possibility of carry term to exists. The same rule is utilized in the operation of half adders.

The figure below represents the logic symbol for half adder:

The figure clearly shows 2 applied inputs and 2 outputs for half adder. Out of the two, one output shows the summation while other shows carrry generated.

Let us have a look at the logic circuit of half adder in order to understand its operation more clearly:



Logic circuit of half adder

Here X and Y are the two inputs applied whereas S and C denotes the sum and carry bits. A half adder is composed of 1 AND gate and 1 XOR logic gate.

The summation bit generated by the half adder is represented by the XOR operation and the carry bit generated by the half adder is represented by the AND operation.

Let us have a look at the truth table of half adder then we will summarize its operation:

Inputs		Outputs	
X	Y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Truth table for Half - Adder

Electronics Desk Case 1: When

both the applied inputs at the half adder is logic low i.e., 0 –

We know that summation of 0 and 0 will result in output 0 and in this case no any carry term is produced.

Case 2: When the first applied input is logic low i.e., 0 and second applied input is logic high i.e., 1 –

We know the addition of 0 and 1 will provide 1 as the sum but no any carry is generated in this case.

Case 3: When the first input is 1 and the second applied input is 0 –

Again the addition of 0 and 1 will generate 1 as the sum but no any carry bit will get generated.

Case 4: When both the applied inputs are logic high i.e., 1 –

The addition of 1 and 1 will generate 2 but in binary terms, we write 2 as 10.

Thus, in this case, 0 will be the sum and 1 will become the carry bit.

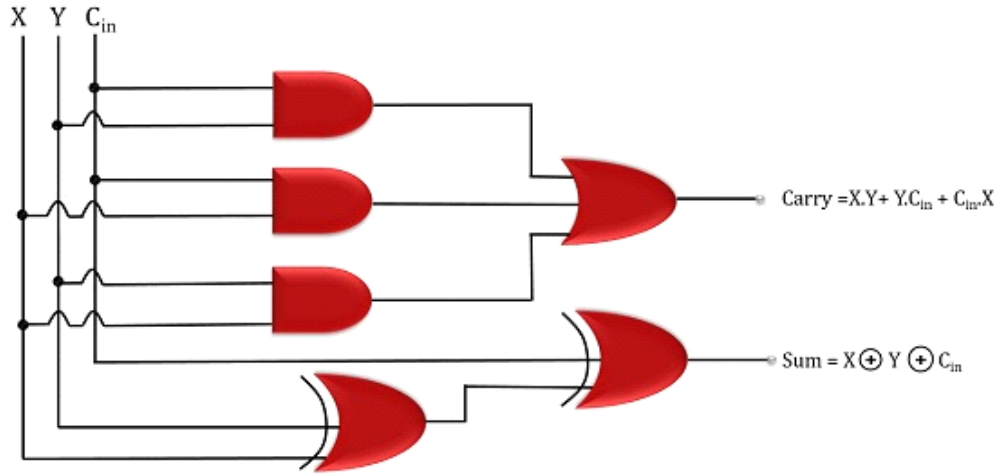
Definition of Full Adder:-

Full adders are the arithmetic circuits that generate a summation of 3 inputs of one- bit.

The figure below represents the logic symbol of a full adder:

This figure clearly shows the 3 inputs applied to the full adder and the 2 outputs. Here, also the one output shows the summation result and the other shows the carry bit generated.

The figure below represents the equivalent logic circuit for full adder:



Logic circuit of full adder

Here, X, Y and C_{in} are the 3 inputs applied to the adder whereas S and C denotes the sum and carry generated.

Let us now analyze the truth table for full adder:

Inputs			Outputs	
X	Y	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Truth table for Full - Adder

We already know the fact that if only one of the inputs is logic high i.e., 1 and the rest are 0. Then, the sum generated will be 1 and the carry will be 0.

But if out of 3, two inputs are logic high i.e., 1. Then its summation will generate 10 in binary forms representing 2. So, in this case, 0 is stored as the sum and 1 is stored as the carry.

When all the 3 applied inputs are logic high i.e., 1 then, in this case, the sum generated will be 1 as well as the carry generated will also be 1.
This is so because the addition of 1 and 1 give 0 as the sum and 1 as the carry this summed output 0 is then added with the last input 1. Thereby generating 1 as overall summation and 1 as the carry.

4. Bit Binary adder :-

Binary adders are implemented to add two binary numbers. So in order to add two 4 bit binary numbers, we will need to use 4 full-adders. The connection of full-adders to create binary adder circuit is discussed in block diagram below.

In this implementation, carry of each full-adder is connected to previous carry.

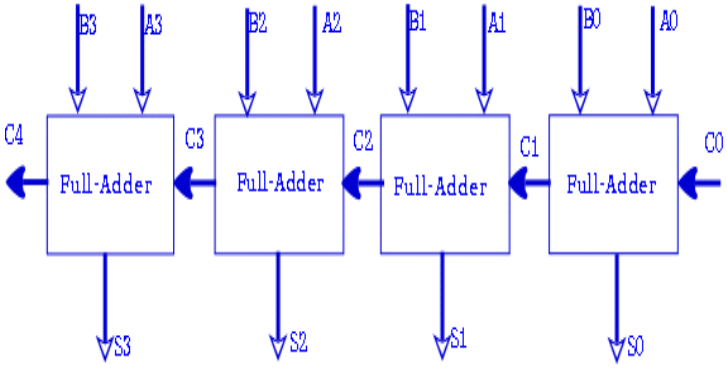
Lets start with the expressions for the FULL ADDER :-

Sum_out = [(in_x) XOR (in_y)] XOR [(carry_in)]

Carry_out = [(in_x) AND (in_y)] OR [(in_x XOR in_y) AND carry_in]

Next we will draw the circuit of 4 Bit Binary Adder

Circuit of 4 Bit Binary Adder consisits of a sequence of full-adders. Details below with circuit and a truth-table.



We will need to discuss an **Example** to understand this in more details. In this example we will use some terms from Register Transfer Level (RTL) implementations.

Understand more about RTL.

Problem: Add two binary numbers 7 and 15 with previous carry = 0.

Solution: Load the values into registers R1 and R2.

So, R1 = 7 (decimal) = 0111 (in binary A3A2A1A0)

& R2 = 15 (decimal) = 1111 (in binary B3B2B1B0)

Lets implement a table and then continue...Memory Tutorial

Stage	Previous carry	Augends bits A	Addend bits B	Sum	Next carry
0	C0=0	A0=1	B0=1	S0=0	C1=1
1	C1=1	A1=1	B1=1	S1=1	C2=1
2	C2=1	A2=1	B2=1	S2=1	C3=1
3	C3=1	A3=0	B3=1	S3=0	C4=1

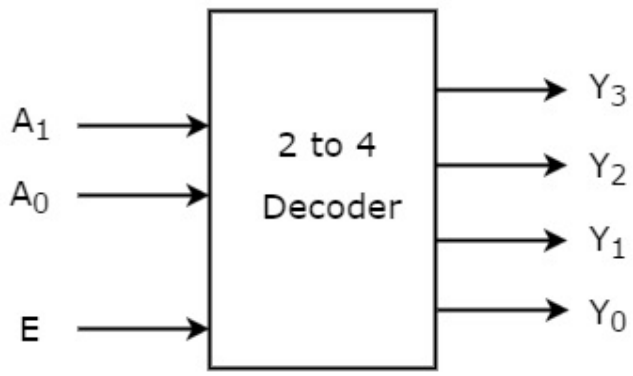
Chapter 7

(Decoders, Multiplexers, & Encoder)

Decoder:- is a combinational circuit that has 'n' input lines and maximum of 2^n output lines. One of these outputs will be active High based on the combination of inputs present, when the decoder is enabled. That means decoder detects a particular code. The outputs of the decoder are nothing but the **min terms** of 'n' input variables lineslines, when it is enabled.

2 to 4 Decoder

Let 2 to 4 Decoder has two inputs A_1 & A_0 and four outputs Y_3, Y_2, Y_1 & Y_0. The **block diagram** of 2 to 4 decoder is shown in the following figure.



One of these four outputs will be '1' for each combination of inputs when enable, E is '1'. The **Truth table** of 2 to 4 decoder is shown below.

Enable	Inputs		Outputs			
E	**A_1**	**A_0**	**Y_3**	**Y_2**	**Y_1**	**Y_0**
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

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

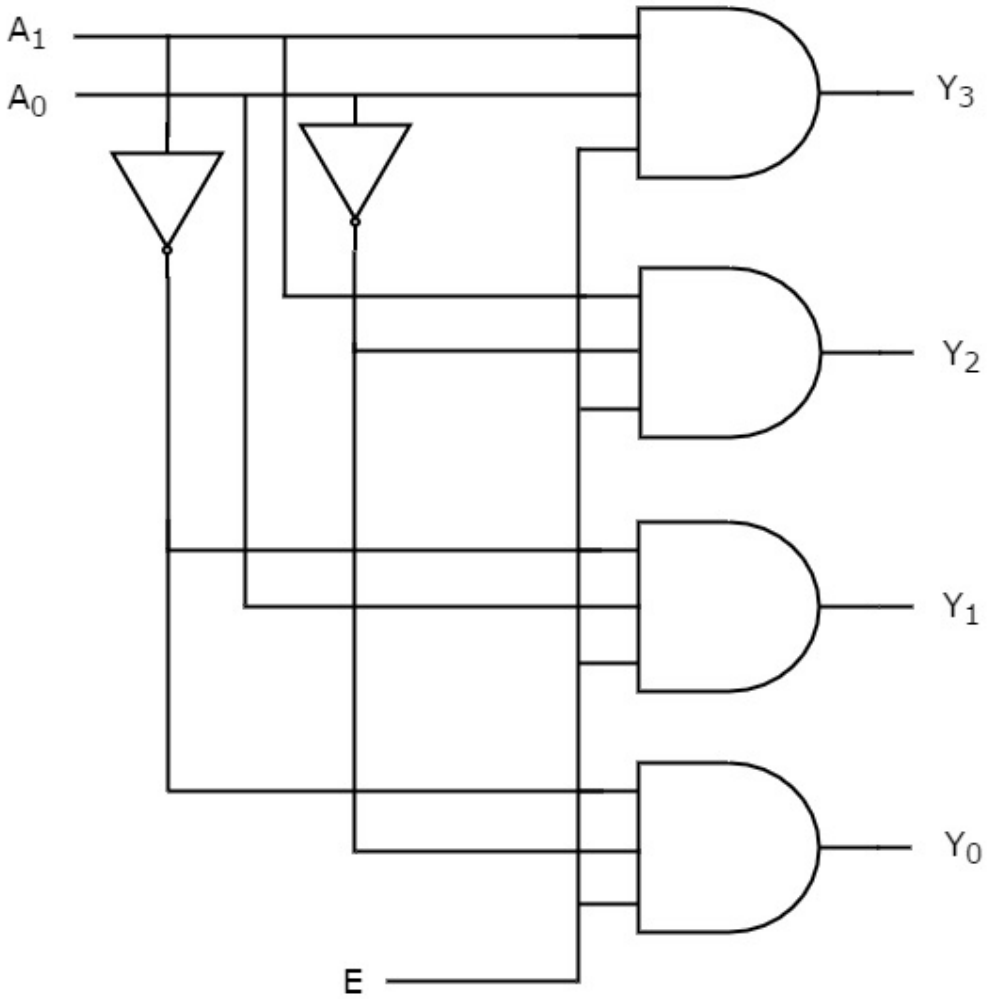
$$Y3=E.A1.A0Y3=E.A1.A0$$

$$Y2=E.A1.A0'Y2=E.A1.A0'$$

$$Y1=E.A1'.A0Y1=E.A1'.A0$$

$$Y0=E.A1'.A0'Y0=E.A1'.A0'$$

Each output is having one product term. So, there are four product terms in total. We can implement these four product terms by using four AND gates having three inputs each & two inverters. The **circuit diagram** of 2 to 4 decoder is shown in the following figure.



Therefore, the outputs of 2 to 4 decoder are nothing but the **min terms** of two input variables A_1 & A_0, when enable, E is equal to one. If enable, E is zero, then all the outputs of decoder will be equal to zero.

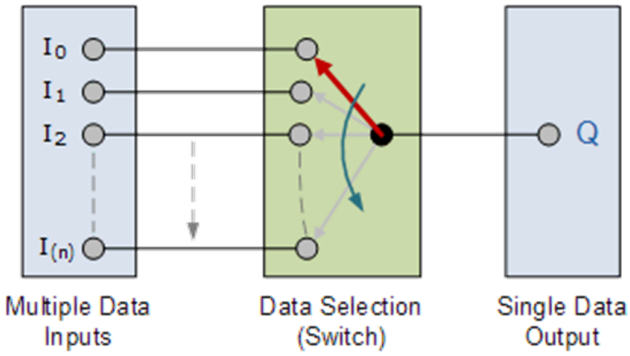
Similarly, 3 to 8 decoder produces eight min terms of three input variables A_2, A_1 & A_0 and 4 to 16 decoder produces sixteen min terms of four input variables A_3, A_2, A_1 & A_0.

Multiplexing is the generic term used to describe the operation of sending one or more analogue or digital signals over a common transmission line at different times or speeds and as such, the device we use to do just that is called a **Multiplexer**. The *multiplexer*, shortened to "MUX" or "MPX", is a combinational logic circuit designed to switch one of several input lines through to a single common output line by the application of a control signal. Multiplexers operate like very fast acting multiple position rotary switches connecting or controlling multiple input lines called "channels" one at a time to the output.

Multiplexers, or MUX's, can be either digital circuits made from high speed logic gates used to switch digital or binary data or they can be analogue types using transistors, MOSFET's or relays to switch one of the voltage or current inputs through to a single output.

The most basic type of multiplexer device is that of a one-way rotary switch as shown.

Basic Multiplexing Switch



The rotary switch, also called a wafer switch as each layer of the switch is known as a wafer, is a mechanical device whose input is selected by rotating a shaft. In other words, the rotary switch is a manual switch that you can use to select individual data or signal lines simply by turning its inputs "ON" or "OFF". So how can we select each data input automatically using a digital device.

In digital electronics, multiplexers are also known as data selectors because they can "select" each input line, are constructed from individual Analogue Switches encased in a single IC package as opposed to the "mechanical" type selectors such as normal conventional switches and relays.

They are used as one method of reducing the number of logic gates required in a circuit design or when a single data line or data bus is required to carry two or more different digital signals. For example, a single 8-channel multiplexer.

Generally, the selection of each input line in a multiplexer is controlled by an additional set of inputs called *control lines* and according to the binary condition of these control inputs, either "HIGH" or "LOW" the appropriate data input is connected directly to the output. Normally, a multiplexer has an even number of 2^n data input lines and a number of "control" inputs that correspond with the number of data inputs.

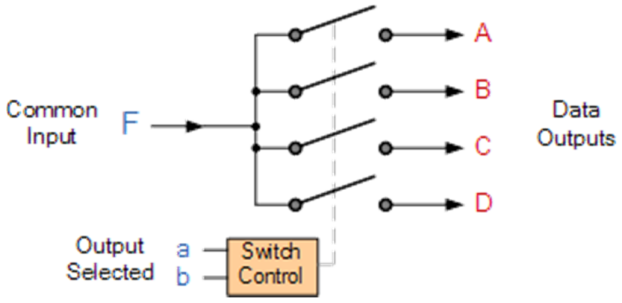
Note that multiplexers are different in operation to *Encoders*. Encoders are able to switch an n-bit input pattern to multiple output lines that represent the binary coded (BCD) output equivalent of the active input.

The Demultiplexer

The demultiplexer is a combinational logic circuit designed to switch one common input line to one of several seperate output line

The *demultiplexer* takes one single input data line and then switches it to any one of a number of individual output lines one at a time. The **demultiplexer** converts a serial data signal at the input to a parallel data at its output lines as shown below.

1-to-4 Channel De-multiplexer



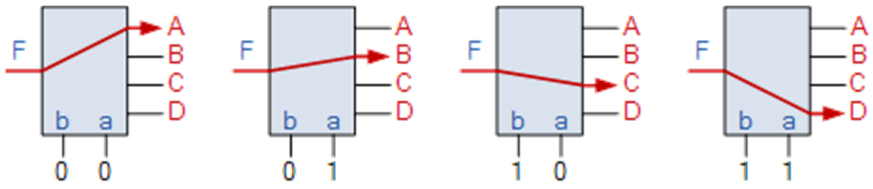
Output Select		Data Output Selected
a	b	
0	0	A
0	1	B
1	0	C
1	1	D

The Boolean expression for this 1-to-4 **Demultiplexer** above with outputs A to D and data select lines a, b is given as:

$$F = abA + abB + abC + abD$$

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

Demultiplexer Output Line Selection



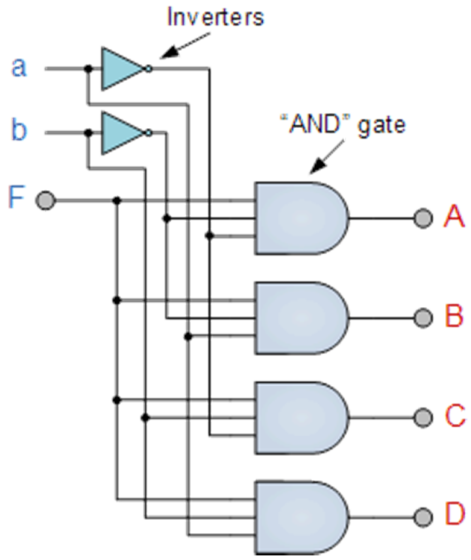
As with the previous multiplexer circuit, adding more address line inputs it is possible to switch more outputs giving a 1-to-2^n data line outputs.

Some standard demultiplexer IC´s also have an additional “enable output” pin which disables or prevents the input from being passed to the selected output. Also some have latches built into their outputs to maintain the output logic level after the address inputs have been changed.

However, in standard decoder type circuits the address input will determine which single data output will have the same value as the data input with all other data outputs having the value of logic “0”.

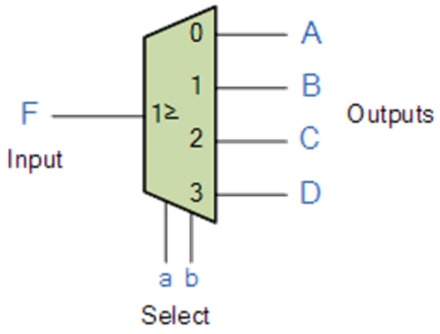
The implementation of the Boolean expression above using individual logic gates would require the use of six individual gates consisting of AND and NOT gates as shown.

4 Channel Demultiplexer using Logic Gates



The symbol used in logic diagrams to identify a demultiplexer is as follows.

The Demultiplexer Symbol



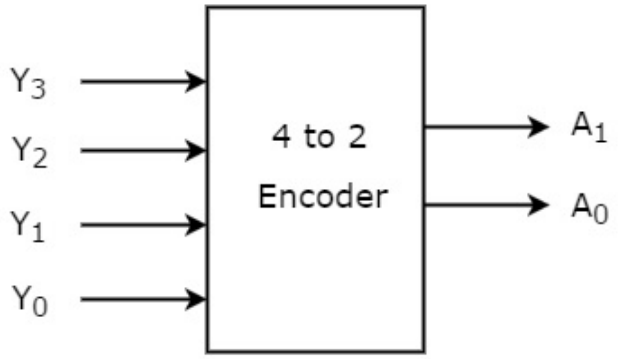
Again, as with the previous multiplexer example, we can also use the demultiplexer to digitally control the gain of an operational amplifier as shown.

Encoder:-

An encoder is a combinational circuit that performs the reverse operation of Decoder. It has maximum of 2^n input lines and 'n' output lines. It will produce a binary code equivalent to the input, which is active High. Therefore, the encoder encodes 2^n input lines with 'n' bits. It is optional to represent the enable signal in encoders.

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.

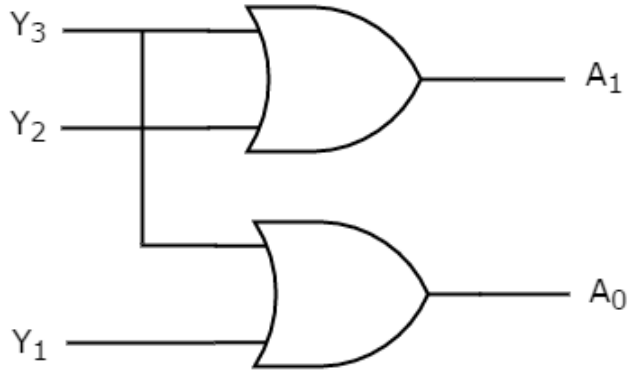
Inputs				Outputs	
Y_3	Y_2	Y_1	Y_0	A_1	A_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

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

$$A1=Y3+Y2A1=Y3+Y2$$

$$A0=Y3+Y1A0=Y3+Y1$$

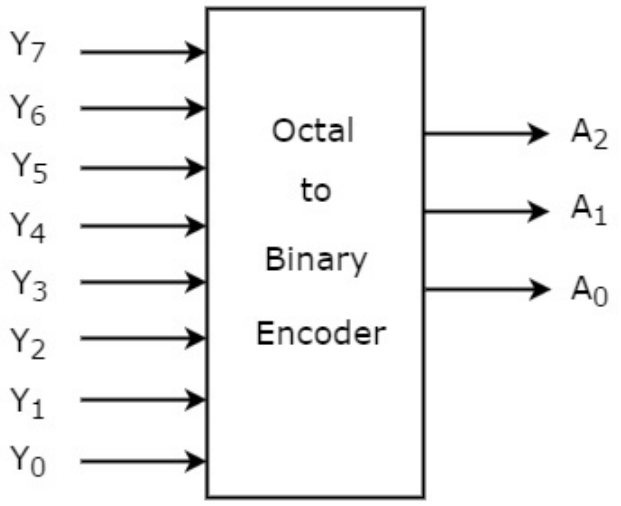
We can implement the above two Boolean functions by using two input OR gates. The **circuit diagram** of 4 to 2 encoder is shown in the following figure.



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

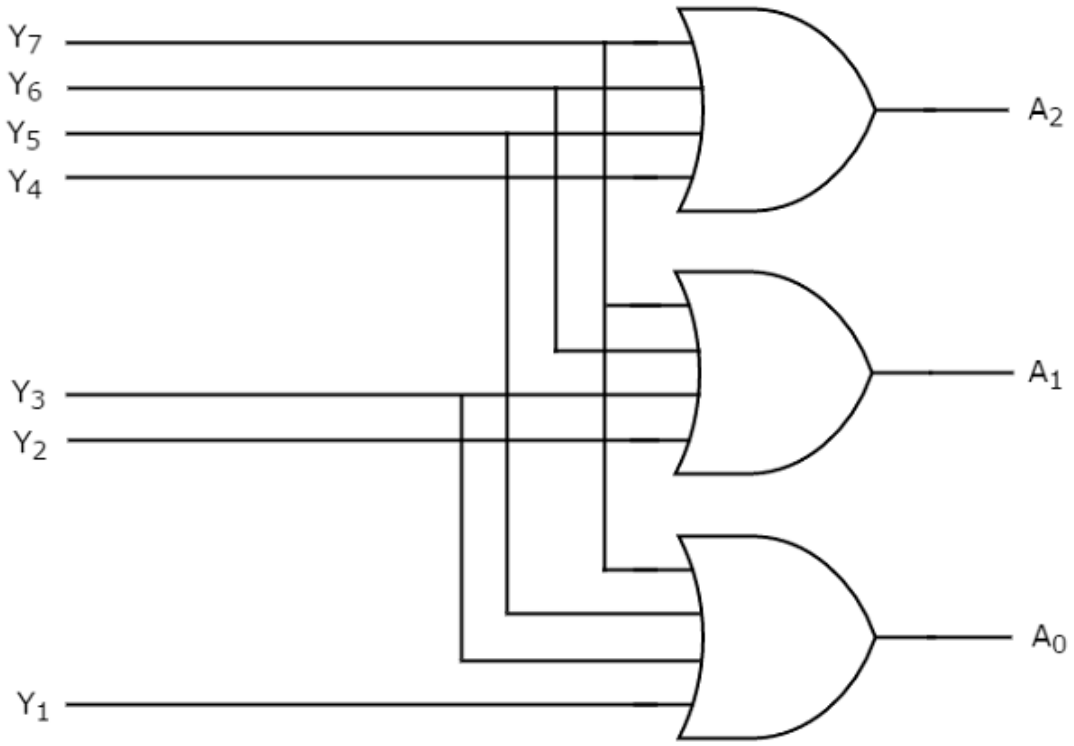
From Truth table, we can write the **Boolean functions** for each output as

$$A2=Y7+Y6+Y5+Y4A2=Y7+Y6+Y5+Y4$$

$$A1=Y7+Y6+Y3+Y2A1=Y7+Y6+Y3+Y2$$

$$A0=Y7+Y5+Y3+Y1A0=Y7+Y5+Y3+Y1$$

We can implement the above Boolean functions by using four input OR gates. The **circuit diagram** of octal to binary encoder is shown in the following figure.



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

Drawbacks of Encoder:-

Following are the drawbacks of normal encoder.

- There is an ambiguity, when all outputs of encoder are equal to zero. Because, it could be the code corresponding to the inputs, when only least significant input is one or when all inputs are zero.
- If more than one input is active High, then the encoder produces an output, which may not be the correct code. For **example**, if both Y_3 and Y_6 are '1', then the encoder produces 111 at the output. This is neither equivalent code corresponding to Y_3, when it is '1' nor the equivalent code corresponding to Y_6, when it is '1'.

So, to overcome these difficulties, we should assign priorities to each input of encoder. Then, the output of encoder will be the binarybinary code corresponding to the active High inputss, which has higher priority. This encoder is called as priority encoder.

Chapter 8
(Latches & flip flops)

Latch:- latch is one kind of a logic circuit and it is also known as a bistable multivibrator. Because it has two stable states namely active high as well as active low. It works like a storage device by holding the data through a feedback lane. It stores 1-bit of data as long as the apparatus is activated. Once enable is declared then instantly latch can change the stored data. It constantly trials the inputs once enable signal is activated. The working of these circuits can be done in 2-states based on the enable signal being high or else low. When the latch circuit is the in an active high state, then both the i/ps are low. Similarly, when the latch circuit is then an active low state, then both the i/ps are high.

Different Types of Latches

The latches can be classified into different types which include SR Latch, **Gated S-R Latch**, **D latch**, Gated D Latch, JK Latch, and T Latch.

SR Latch

An **SR (Set/Reset) latch** is an asynchronous apparatus, and it works separately for control signals by depending on the S-state & R-inputs. The SR-latch using 2-NOR gates with a cross loop connection is exhibited below. These latches can be built with **NAND gates** also; however, the two inputs are exchanged as well as canceled. So it is called as SR'-latch.

SR Latch

Whenever a high input is given to the S-line of the latch, then the output Q goes high. In the feedback process, the output Q will stay high, when the S-input goes low once more. In this way, the latch works as a memory device.

Equally, a high input is given to the R-line of the latch, then the Q output goes low (and Q' high), then the memory of the latch will effectively reset. When both the inputs of the latch are low, then it stays in its earlier set state or reset state.

The **state transition table or truth table** of SR latch is shown below.

S	R	Q	Q'
0	0	Latch	Latch
0	1	0	1
1	0	1	0
1	1	0	0

When both inputs are high at once, there is trouble: it is being told toward concurrently generating a high Q & low Q. This generates a race condition in the circuit either flip flop achieves something in altering first will respond to the other & declares itself. Preferably, both **Logic gates** are equal and the device will be in an undefined condition for an indefinite stage.

Gated SR Latch

In some cases, it may be popular to order when the latch can & cannot latch. The simple extension of an **SR latch** is nothing but a **gated SR latch**. It gives an Enable line that should be driven high before information can be latched. Although a control line is necessary, the latch is not synchronous due to the inputs which can alter the output even in the middle of an enable pulse.

Gated SR Latch

When the input of an Enable is low, the o/ps from the gates must also be less, therefore the Q & Q outputs stay latched toward the earlier information. Simply when the enable i/p is high can change the position of the latch, as shown in the tabular form. As the enable line is stated, a gated SR-latch is equal in the process toward an SR latch. Sometimes, an enable line is a CLK signal; however, it is a read/write strobe.

CLK	S	R	Q(t+1)
0	X	X	Q(t) (no change)
1	0	0	Q(t) (no change)
1	0	1	0
1	1	0	1
1	1	1	X

D Latch

The data latch is an easy expansion to the gated SR-latch that eliminates the chance of unacceptable states of input. Because the gated SR latch lets us fastener the output without employing the inputs of S or R, we can eliminate one of the i/ps by driving both the inputs with an opposite driver. We eliminate one input & automatically make it opposite of the residual input.

D Latch

The D-latch outputs the input of the D when the Enable line is high, otherwise, the output is whatever the D input was whenever the Enable input was last high. This is the reason it is known as a transparent latch. When Enable is stated, then the latch is called as transparent and signals spread straightly through it since if it isn't present.

E	D	Q	Q'
0	0	Latch	Latch
0	1	Latch	Latch
1	0	0	1
1	1	1	0

Gated D Latch

A **gated D latch** is designed simply by changing a gated SR-latch, and the only change in the gated SR-latch is that the input R must be modified to inverted S. Gated latch cannot be formed from SR-latch using NOR is shown below.

Gated D Latch

Whenever the CLK otherwise enable is high, the o/p latches anything is on the input of the D. Similarly when the CLK is low, then the D i/p for the final enable high is the output.

CLK	D	Q(t+1)
0	X	Q(t)
1	0	0
1	1	1

The circuit of the latch will not at all experience a Race state due to the only D input is reversed to offer to both the inputs. Therefore, there is no possibility for similar input state. Thus the circuit of D-latch can be securely used in several circuits.

JK Latch

The both **JK latch**, as well as RS latch, is similar. This latch comprises two inputs namely J and K which are shown in the following logic gate diagram. In this type of latch, the unclear state has been removed here. When the JK latch inputs are high, the output will be toggled. The only difference we can observe here is the output feedback toward inputs, which is not present in the RS-latch.

T Latch

The **T latch** can be formed whenever the JK latch inputs are shorted. The function of T Latch will be like this when the input of the latch is high, and then the output will be toggled.

Advantages of Latches

- The designing of latches is very flexible when we compare with **FFs (flip-flops)**
- The latches utilize less power.
- The performance of latch in the design of the high-speed circuit is quick because these are asynchronous within the design and there is no need of CLK signal.
- The shape of the latch is very small and occupies less area
- If the operation of latch based circuit is not finished in a set time, they borrow the necessary time from other to complete the operation
- Latches give aggressive clocking when contrasted with flip-flop circuits.

Disadvantages of Latches

- There will be a chance of affecting the race condition, so these are less expected.
- When a latch is level sensitive, then there is a chance of meta-stability.
- Analyzing the circuit is difficult due to the property of level sensitive.
- The circuit can be tested by using an extra CAD program.

Application of Latche:-

- Generally, latches are used to keep the conditions of the bits to encode binary numbers
- Latches are single bit storage elements which are widely used in computing as well as data storage.
- Latches are used in the circuits like power gating & clock as a storage device.
- D latches are applicable for asynchronous systems like input or output ports.
- Data latches are used in synchronous two-phase systems for reducing the transit count.

Thus, this is all about an overview of latches. These are the building blocks for **sequential circuits**. The designing of this can be done using logic gates. Ioperation mainly depends on the input of an enable function.

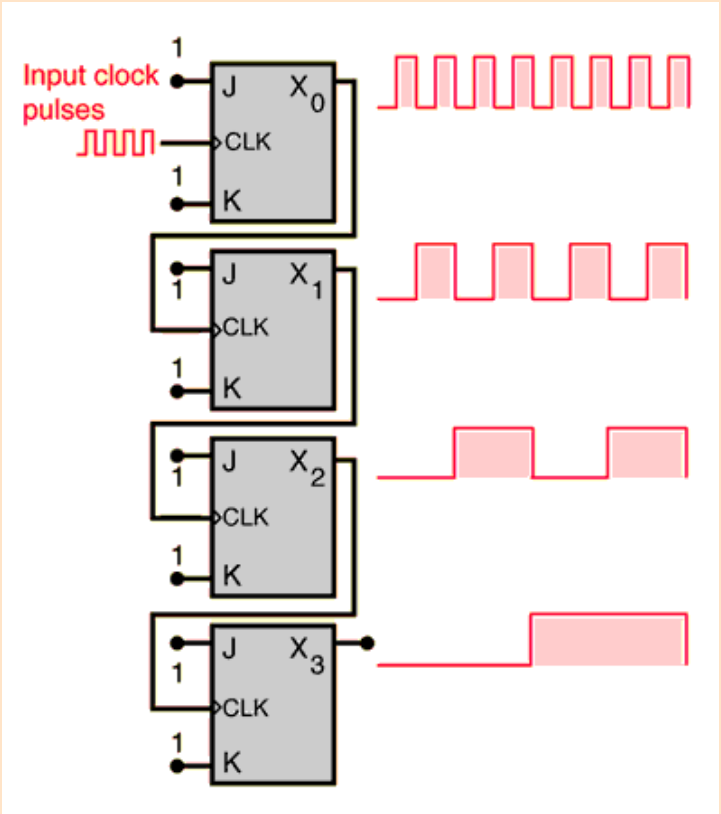
Chapter 9

Counters

S.NO	SYNCHRONOUS COUNTER	ASYNCHRONOUS COUNTER
1.	In synchronous counter, all flip flops are triggered with same clock simultaneously.	In asynchronous counter, different flip flops are triggered with different clock, not simultaneously.
2.	Synchronous Counter is faster than asynchronous counter in operation.	Asynchronous Counter is slower than synchronous counter in operation.
3.	Synchronous Counter does not produce any decoding errors.	Asynchronous Counter produces decoding error.
4.	Synchronous Counter is also called Parallel Counter.	Asynchronous Counter is also called Serial Counter.
5.	Synchronous Counter designing as well implementation are complex due to increasing the number of states.	Asynchronous Counter designing as well as implementation is very easy.
6.	Synchronous Counter will operate in any desired count sequence.	Asynchronous Counter will operate only in fixed count sequence (UP/DOWN).
7.	Synchronous Counter examples are: Ring counter, Johnson counter.	Asynchronous Counter examples are: Ripple UP counter, Ripple DOWN counter.

Binary Counting

A binary counter can be constructed from J-K flip-flops by taking the output of one cell to the clock input of the next. The J and K inputs of each flip-flop are set to 1 to produce a toggle at each cycle of the clock input. For each two toggles of the first cell, a toggle is produced in the second cell, and so on down to the fourth cell. This produces a binary number equal to the number of cycles of the input clock signal. This device is sometimes called a "ripple through" counter. The same device is useful as a frequency divider.



Ripple Counter in Digital Logic:-

A **counter** is basically used to count the number of clock pulses applied to a flip-flop. It can also be used for Frequency divider, time measurement, frequency measurement, distance measurement and also for generating square waveforms. In this, the flip-flops are asynchronous counters and are supplied with different clock signals, there may be a delay in producing output.

Also, a few numbers of logic gates are needed to design asynchronous counters. So they are elementary in design and also are less expensive.

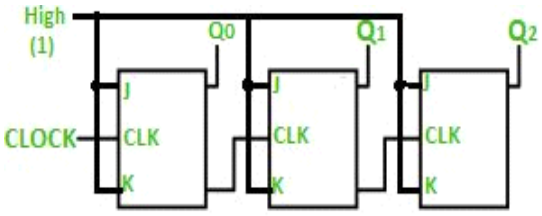
Ripple counter –

A n-bit ripple counter can count up to 2^n states. It is also known as MOD n counter. It is known as ripple counter because of the way the clock pulse ripples its way through the flip-flops. Some of the features of ripple counter are:

- It is an asynchronous counter.
- Different flip-flops are used with a different clock pulse.
- All the flip-flops are used in toggle mode.
- Only one flip-flop is applied with an external clock pulse and another flip-flop clock is obtained from the output of the previous flip-flop.
- The flip-flop applied with external clock pulse act as LSB (Least Significant Bit) in the counting sequence.

A counter may be an up counter that counts upwards or can be a down counter that counts downwards or can do both i.e.count up as well as count downwards depending on the input control. The sequence of counting usually gets repeated after a limit. When counting up, for n-bit counter the count sequence goes from 000, 001, 010, ... 110, 111, 000, 001, ... etc. When counting down the count sequence goes in the opposite manner: 111, 110, ... 010, 001, 000, 111, 110, ... etc.

A 3-bit Ripple counter using JK flip-flop:-



In the circuit shown in above figure, Q0(LSB) will toggle for every clock pulse because JK flip-flop works in toggle mode when both J and K are applied 1, 1 or high input. The following counter will toggle when the previous one changes from 1 to 0

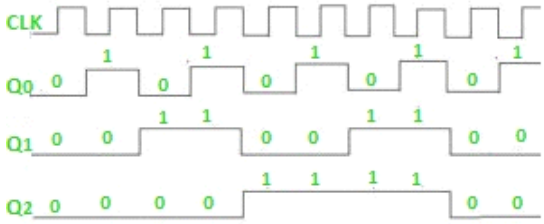
Truth Table :-

Counter State	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

The 3-bit ripple counter used in the circuit above has eight different states, each one of which represents a count value. Similarly, a counter having n flip-flops can have a maximum of 2 to the power n states. The number of states that a counter owns is known as its mod (modulo) number. Hence a 3-bit counter is a mod-8 counter.

A mod-n counter may also be described as a divide-by-n counter. This is because the most significant flip-flop (the furthest flip-flop from the original clock pulse) produces one pulse for every n pulses at the clock input of the least significant flip-flop (the one triggers by the clock pulse). Thus, the above counter is an example of a divide-by-4 counter.

Timing diagram – Let us assume that the clock is negative edge triggered so above counter will act as an up counter because the clock is negative edge triggered and output is taken from Q.



Counters are used very frequently to divide clock frequencies and their uses mainly involve in digital clocks and in multiplexing. The widely known example of the counter is parallel to serial data conversion logic.

Attention reader! Don’t stop learning now. Get hold of all the important DSA concepts with the DSA Self Paced Course at a student-friendly price and become industry ready.

Chapter 10

(Shift Register)

In **Serial In Parallel Out (SIPO) shift registers**, the data is stored into the register serially while it is retrieved from it in parallel-fashion. Figure 1 shows an n-bit synchronous **SIPO shift register** sensitive to positive edge of the clock pulse. Here the data word which is to be stored (Data in) is fed serially at the input of the first flip-flop (D_1 of FF_1). It is also seen that the inputs of all other flip-flops (except the first flip-flop FF_1) are driven by the outputs of the preceding ones say for example, the input of FF_2 is driven by the output of FF_1. In this kind of shift register, the data stored within the register is obtained as a parallel-output data word (Data out) at the individual output pins of the flip-flops (Q_1 to Q_n).

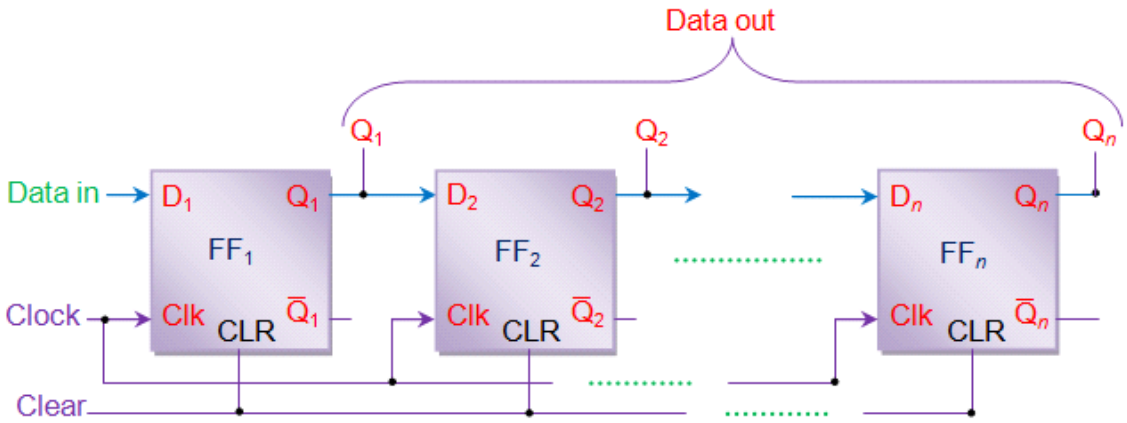


Figure 1 *n*-bit Serial-In Parallel-Out Right-Shift Shift Register

In general, the register contents are cleared by applying high on the clear pins of all the flip-flops at the initial stage. After this, the first bit, B_1 of the input data word is fed at the D_1 pin of FF_1.

This bit (B_1) will enter into FF_1, get stored and thereby appears at its output Q_1 on the appearance of first leading edge of the clock. Further at the second clock tick, the bit B_1 right-shifts and gets stored into FF_2 while appearing at its output pin Q_2 while a new bit, B_2 enters into FF_1. Similarly at each clock tick the data within the register moves towards right by a single bit while a new bit of the input word enters into the register. Meanwhile one can extract the bits stored within the register in parallel-fashion at the individual flip-flop outputs.

Analyzing on the same grounds, one can note that the n-bit input data word is obtained as an n-bit output data word from the shift register at the rising edge of the n^{th} clock pulse. This working of the shift-register can be summarized as in Table I and the corresponding wave forms are given by Figure 2.

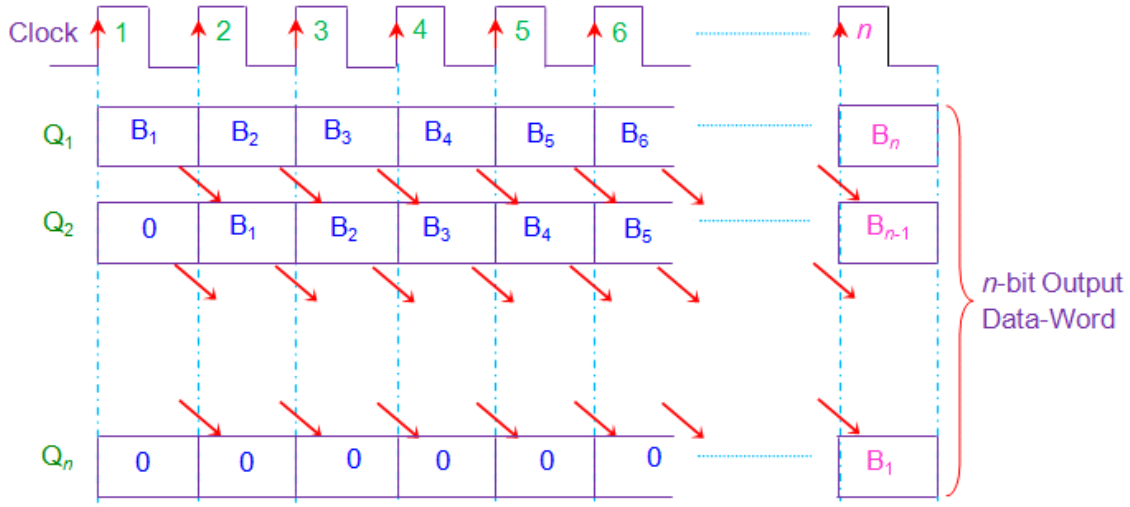


Figure 2 Output Waveform of *n*-bit Right-Shift SIPO Shift Register

In the right-shift SIPO shift-register, data bits shift from left to right for each clock tick. However if the data bits are made to shift from right to left in the same design, one gets a left-shift SIPO shift-register as shown by Figure 3. Nevertheless the basic working principle remains the same except the fact that now B_n down to B_1 is stored in Q_n down to Q_1 i.e. $Q_1 = B_1$, $Q_2 = B_2$ ... $Q_n = B_n$ at the n^{th} clock tick.

Serial In Serial Out (SISO) shift registers are a kind of shift registers where both data loading as well as data retrieval to/from the shift register occurs in serial-mode. Figure 1 shows a n-bit synchronous **SISO shift register** sensitive to positive edge of the clock pulse. Here the data word which is to be stored is fed bit-by-bit at the input of the first flip-flop. Further it is seen that the inputs of all other flip-flops (except the first flip-flop FF_1) are driven by the outputs of the preceding ones say for example, the input of FF_2 is driven by the output of FF_1. At last the data stored within the register is obtained at the output pin of the n^{th} flip-flop in serial-fashion.

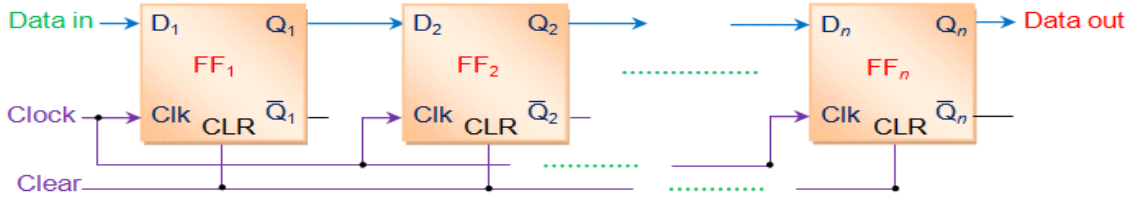


Figure 1 *n*-bit Right-Shift Serial-in Serial-Out Shift Register

Initially all the flip-flops in the register are cleared by applying high on their clear pins. Next the input data word is fed serially to FF_1.

This causes the bit appearing at the D_1 pin (B_1) to be stored into FF_1 as soon as the first leading edge of the clock appears. Further at the second clock tick, B_1 gets stored into FF_2while a new bit enters into FF_1 (B_2).

This kind of shift in data bits continues for every rising edge of the clock pulse. This indicates that for every single clock pulse the data within the register moves towards right by a single bit. Thus the design shown in Figure 1 is regarded as a right-shift **SISO shift register**. Following the data transmission as explained, one can note that the first bit of an input word appears at the output of n^{th} flip-flop for the n^{th} clock tick. On applying further clock cycles, one gets the next successive bits of the input data word as the serial output (Table I). The waveforms pertaining to the same are shown by Figure 2.

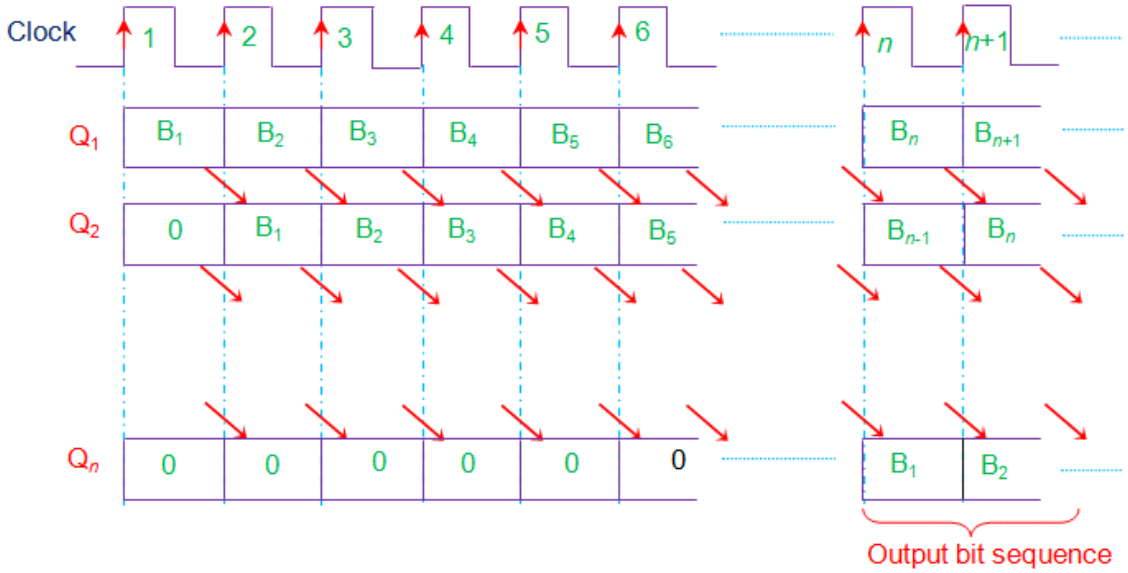


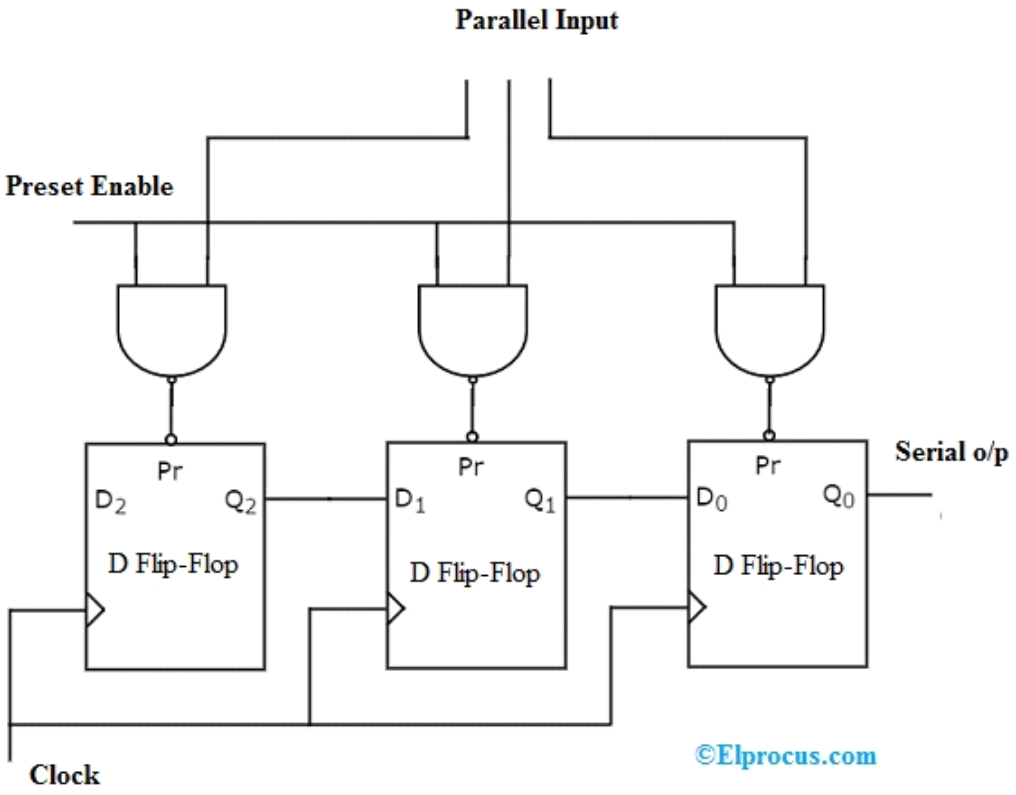
Figure 2 **Output Waveform of *n*-bit Right-Shift SISO Shift Register**

Similar to the right-shift SISO shift-register shown, there can exist a left-shift **SISO shift-register** also (Figure 3). However the working principle remains the same except the fact that the data movement will be from right to left.

Parallel in-Serial out (PISO) Shift Register:-

This shift register allows parallel input and generates a serial output, so this is known as Parallel in Serial out (PISO) Shift Register.

The Parallel in Serial out (PISO) Shift Register circuit is shown above. This circuit can be built with four D-flip-flops, where the CLK signal is connected directly to all the FFs. However, the input data is connected separately to every FF using a **multiplexer** HYPERLINK "https://www.elprocus.com/multiplexer-and-demultiplexer/" at every FF's input.

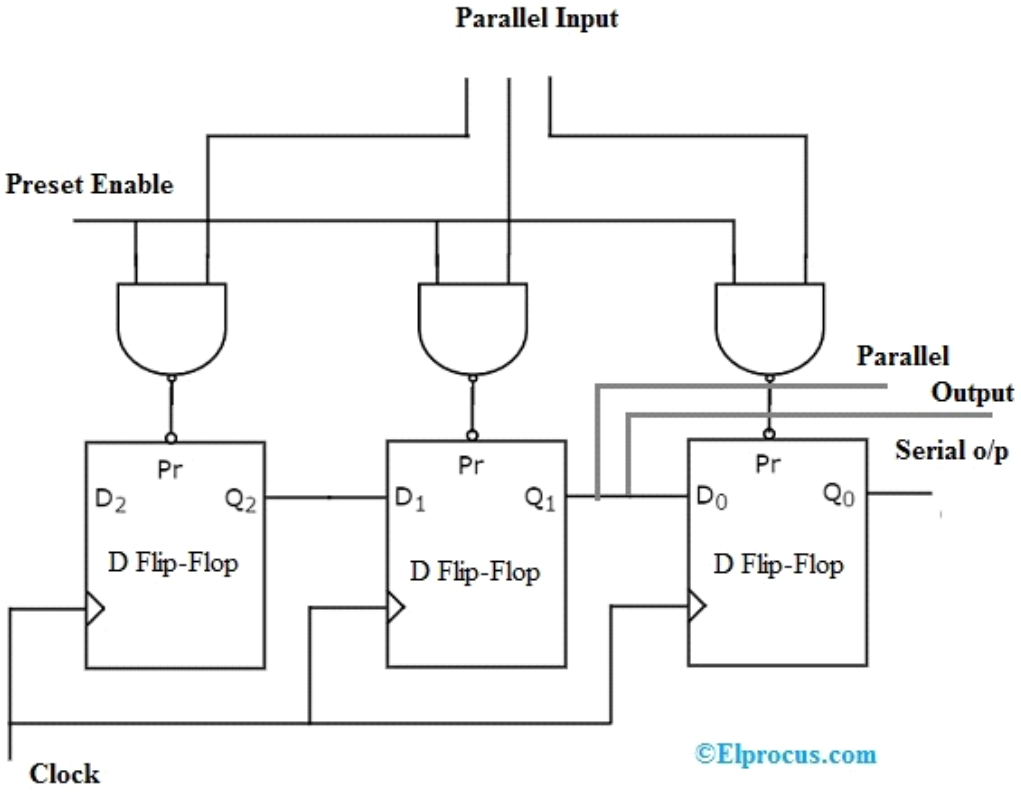


The earlier FF output, as well as parallel data input, is connected toward the multiplexer's input & multiplexer's output can be connected to the second flip flop. Once the same CLK signal is given to every flip flop, then all the flip flops will be synchronous with each other. The applications of these registers include converting parallel data to the serial data.

Parallel in-Parallel out (PIPO) Shift Register:-

The shift register, which allows parallel input (data is given separately to each flip flop and in a simultaneous manner) and also produces a parallel output is known as Parallel-In parallel-Out shift register.

The logic circuit given below shows a parallel in parallel out shift register. The circuit consists of four D flip-flops which are connected. The clear (CLR) signal and clock signals are connected to all the 4 flip flops. In this type of register, there is no interconnection between the individual flip-flops as no data serial shifting is necessary. Here the data is given as input individually for every flip-flop, as well as the output is also received separately from every flip flop.



A PIPO (Parallel in Parallel out) shift register can be utilized like a temporary storage device, similar to SISO Shift register, and it performs like a delay element.

What is a Universal Shift Register:-

Definition: A register that can store the data and /shifts the data towards the right and left along with the parallel load capability is known as a universal shift register. It can be used to perform input/output operations in both serial and parallel modes. Unidirectional shift registers and bidirectional shift registers are combined together to get the design of the universal shift register. It is also known as a parallel-in-parallel-out shift register or shift register with the parallel load.

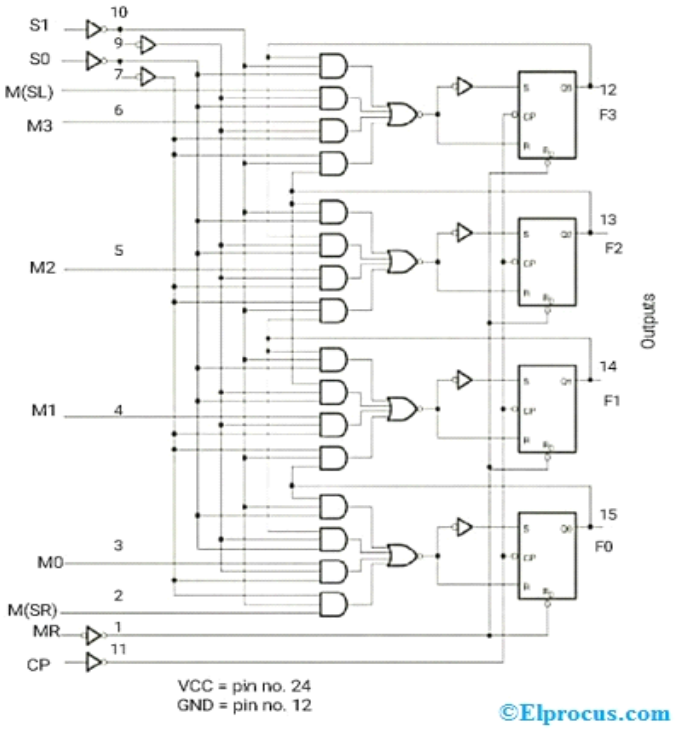
Universal shift registers are capable of performing 3 operations as listed below.

- **Parallel load operation –** stores the data in parallel as well as the data in parallel
- **Shift left operation –** stores the data and transfers the data shifting towards left in the serial path
- **Shift right operation –** stores the data and transfers the data by shifting towards right in the serial path.

Hence, Universal shift registers can perform input/output operations with both serial and parallel loads.

Universal Shift Register Diagram:-

4-bit Universal shift register diagram is shown below.



- Serial input for shift-right control enables the data transfer towards the right and all the serial input and output lines are connected to the shift-right mode. The input is given to the AND gate-1 of the flip-flop -1 as shown in the figure via serial input pin.
- Serial input for shift-left enables the data transfer towards the left and all the serial input and output lines are connected to shift-left mode.
- In parallel data transfer, all the parallel inputs and outputs lines are associated with the parallel load.
- Clear pin clears the register and set to 0.
- CLK pin provides clock pulses to synchronize all the operations.
- In the control state, the information or data in the register would not change even though the clock pulse is applied.
- If the register operates with a parallel load and shifts the data towards the right and left, then it acts as a universal shift register.

Universal Shift Register Working:-

- From the above figure, selected pins the mode of operation of the universal shift register. Serial input shifts the data towards the right and left and stores the data within the register.
- Clear pin and CLK pin are connected to the flip-flop.
- M0, M1, M2, M3 are the parallel inputs while F0, F1, F2, F3 are the parallel outputs of flip-flops
- When the input pin is active HIGH, then the universal shift register loads / retrieve the data in parallel. In this case, the input pin is directly connected to 4×1 MUX
- When the input pin (mode) is active LOW, then the universal shift register shifts the data. In this case, the input pin is connected to 4×1 MUX via NOT gate.
- When the input pin (mode) is connected to GND (Ground), then the universal shift register acts as a Bi-directional shift register.
- To perform the shift-right operation, the input pin is fed to the 1st AND gate of the 1st flip-flop via serial input for shit-right.
- To perform the shift-left operation, the input pin is fed to the 8th AND gate of the last flip-flop via input M.
- If the selected pins S0= 0 and S1 = 0, then this register doesn't operate in any mode. That means it will be in a Locked state or no change state even though the clock pulses are applied.

- If the selected pins S0 = 0 and S1 = 1, then this register transfers or shifts the data to left and stores the data.
- If the selected pins S0 = 1 and S1 = 0, then this register shifts the data to right and hence performs the shift-right operation.
- If the selected pins S0 = 1 and S1 = 1, then this register loads the data in parallel. Hence it performs the parallel loading operation and stores the data.

S0	S1	Mode of Operation
0	0	Locked state (No change)
0	1	Shift-Left
1	0	Shift-Right
1	1	Parallel Loading

From the above table, we can observe that this register operates in all modes with serial/parallel inputs using 4×1 multiplexers and flip-flops.

Advantages:-

The advantages of a universal shift register include the following-

- This register can perform 3 operations such as shift-left, shift-right, and parallel loading.
- Stores the data temporarily with in the register.
- It can perform serial to parallel, parallel to serial, parallel to parallel and serial to serial operations.
- It can perform input-output operations in both the modes serial and parallel.
- A Combination of the unidirectional shift register and bidirectional shift register gives the universe shift register.
- This register acts as an interface between one device to another device to transfer the data.

Applications

The applications of a universal shift register include the following.

- Used in micro-controllers for I/O expansion
- Used as a serial-to-serial converter
- Used as a parallel-to-parallel data converter
- Used as a serial-to-parallel data converter.
- Used in serial – to – serial data transfer
- Used in parallel data transfer.
- Used as a memory element in digital electronics like computers.
- Used in time delay applications
- Used as frequency counters, binary counters, and Digital clocks
- Used in data manipulation applications.

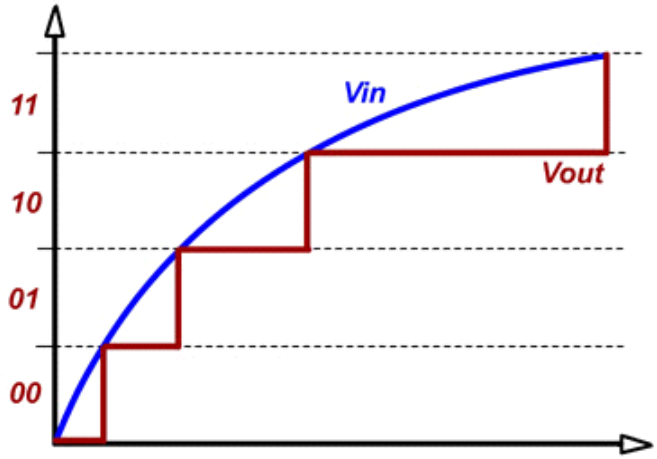
Chapter 11

A/D Converter and D/A Converter

Digital Signal by ADC Converter

One of the major benefits of ADC converter is high data acquisition rate even at multiplexed inputs. With the invention of a wide variety of ADC integrated circuits (IC's), data acquisition from various sensors becomes more accurate and faster. Dynamic characteristics of the high performance ADCs are improved measurement repeatability, low power consumption, precise throughput, high linearity,excellent Signal-to-Noise Ratio (SNR) and so on.

How to Convert the Analog Signal to

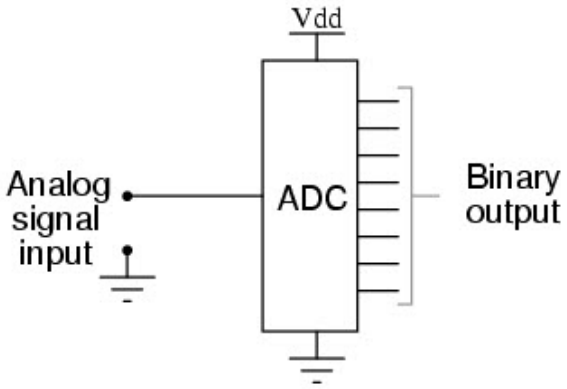


A variety of applications of the ADCs are measurement and control systems, industrial instrumentation, communication systems and all other sensory based systems.

Classification of ADCs based on factors like performance, bit rates, power, cost, etc

Analog to Digital Converter

Almost every environmental measurable parameter is in analog form like temperature, sound, pressure, light, etc. Consider a temperature monitoring system wherein acquiring, analyzing and processing temperature data from sensors is not possible with digital computers and processors. Therefore, this system needs an intermediate device to convert the analog temperature data into digital data in order to communicate with the digital processors like microcontrollers and microprocessors.



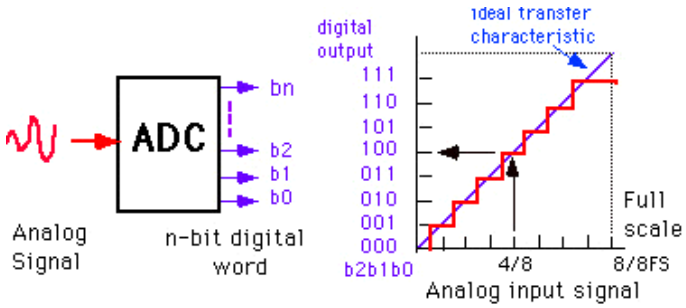
Analog to Digital Converter (ADC) is an electronic integrated circuit used to convert the analog signals such as voltages to digital or binary form consisting of 1s and 0s.Most of the ADCs take a voltage input as 0 to 10V, -5V to +5V, etc. and correspondingly produces digital output as some sort of a binary number.

Analog to Digital Conversion Process

Analog to Digital Converter:-

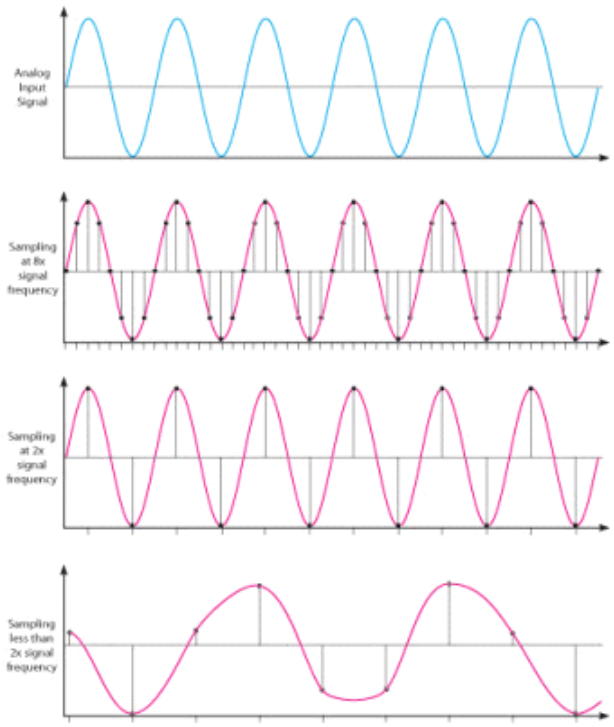
It samples the analog signal on each falling or rising edge of sample clock. In each cycle, the ADC gets of the analog signal, measures and converts it into a digital value. The ADC converts the output data into a series of digital values by approximates the signal with fixed precision.

In ADCs, two factors determine the accuracy of the digital value that captures the original analog signal. These are quantization level or bit rate and sampling rate.Below figure depicts how analog to digital conversion takes place. Bit rate decides decides the resolution of of digitized output and you can observe in below figure where 3-bit ADC is used for converting analog signal.



Assume that one volt signal has to be converted from digital by using 3-bit ADC as shown below. Therefore, a total of 2^3=8 divisions are available for producing 1V output. This results 1/8=0.125V is called as minimum change or quantization level represented for each division as 000 for 0V, 001 for 0.125, and likewise upto 111 for 1V. If we increase the bit rates like 6, 8, 12, 14, 16, etc. we will get a better precision of the signal. Thus, bit rate or quantization gives the smallest output change in the analog signal value that results from a change in the digital representation.

There is an absolute chance of misrepresenting the input signal at output side if it is sampled at different frequency than desired one. Therefore, another important consideration of the ADC is the sampling rate. Niquest theorem states that the acquired signal reconstruction introduces distortion unless it is sampled at (minimum) twice the rate of the largest frequency content of the signal as you can observe in the diagram. But this rate is 5-10 times the maximum frequency of the signal in practical.



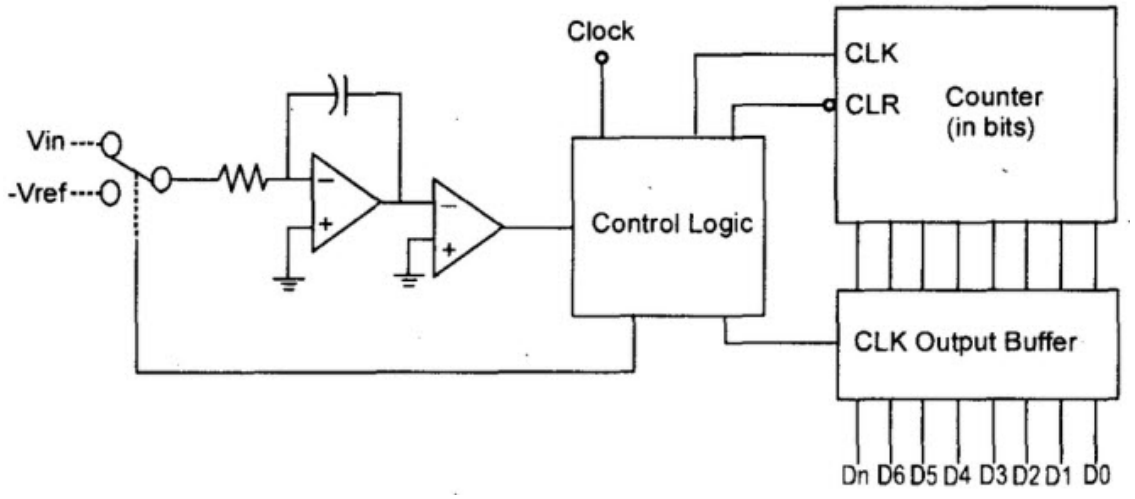
Types of Analog to Digital Converters

Some of the types of analog to digital converters include:

- Dual Slope A/D Converter
- Flash A/D Converter
- Successive Approximation A/D Converter

Dual Slope A/D Converter:-

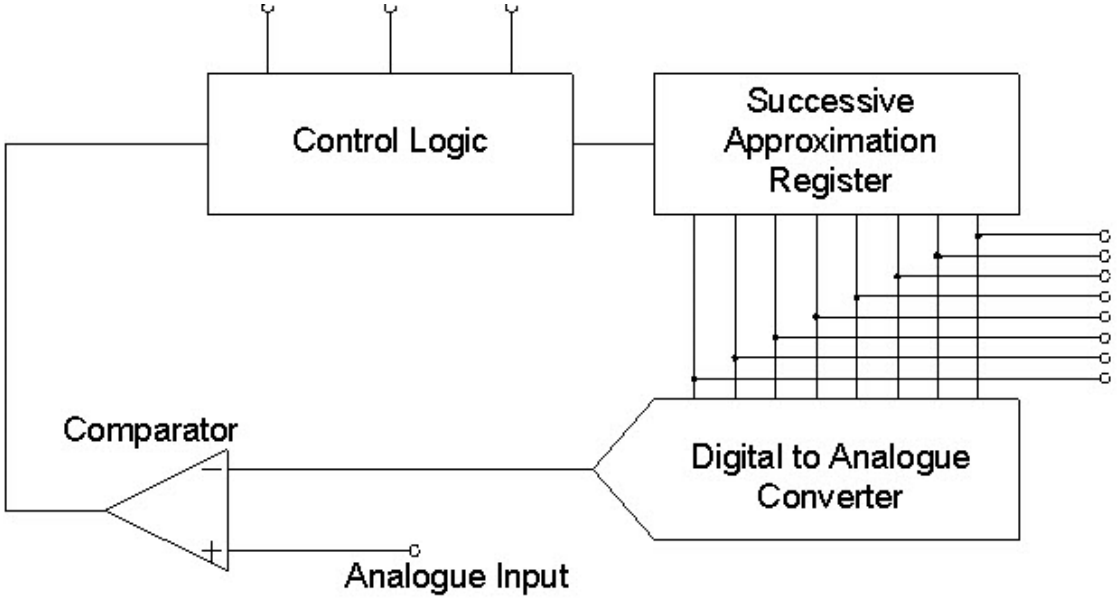
In this type of ADC converter comparison voltage is generated by using an integrator circuit which is formed by a resistor, capacitor and operational amplifier combination. By the set value of Vref, this integrator generates a sawtooth waveform on its output from zero to the value Vref. When the integrator waveform is started correspondingly counter starts counting from 0 to 2^n-1 where n is the number of bits of ADC. When the input voltage Vin equal to the voltage of the waveform, then control circuit captures the counter value which is the digital value of corresponding analog input value. This Dual slope ADC is relatively medium cost and slow speed device.



Successive Approximation A/D Converter

The SAR ADC a most modern ADC IC and much faster than dual slope and flash ADCs since it uses a digital logic that converges the analog input voltage to the closest value. This circuit consists of a comparator, output latches, successive approximation register (SAR) and D/A converter.

At the start, SAR is reset and as the LOW to HIGH transition is introduced, the MSB of the SAR is set. Then this output is given to the D/A converter that produces an analog equivalent of the MSB, further it is compared with the analog input Vin.If comparator output is LOW, then MSB will be cleared by the SAR, otherwise the MS



B will be set to the next position.This process continues till all the bits are tried and after Q0, the SAR makes the parallel output lines to contain valid data.

This is about the ADC converter and its types. For easier understanding only few ADC converters are discussed in this article. We hope this furnished content is more informative to readers. Any further queries, doubts and technical help on this topic you can comment below.

Principles of ADC (Analog to Digital Conversion):-

Principles of ADC – The input signal is compared with an internally generated voltage which is increased in steps starting from zero. The number of steps needed to reach the full compensation is counted. A simple compensation type is the staircase ramp.

The Staircase Ramp

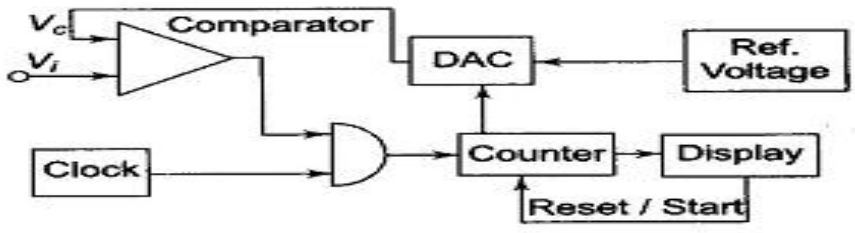


Fig. 5.8 Block Diagram of a Staircase Ramp Type

The basic principle is that the input signal V_i is compared with an internal staircase voltage, V_c generated by a series circuit consisting of a pulse generator (clock), a counter counting the pulses and a digital to analog converter, converting the counter output into a dc signal. As soon as V_c is equal to V_i, the input comparator closes a gate between the clock and the counter, the counter stops and its output is shown on the display. The basic block diagram is shown in Fig5.8.

Operation of the Circuit:-

The clock generates pulses continuously. At the start of a measurement, the counter is reset to 0 at time $t_1$so that the output of the digital to analog converter (DAC) is also 0. If V_i is not equal to zero, the input comparator applies an output voltage that opens the gate so that clock pulses are passed on to the counter through the gate. The counter starts counting and the DAC starts to produce an output voltage increasing by one small step at each count of the counter. The result is a staircase voltage applied to the second input of the comparator, as shown in Fig. 5.9.

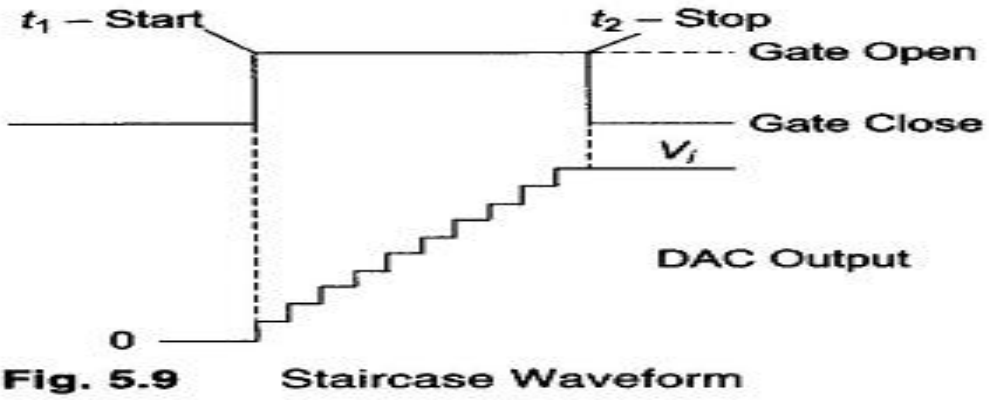


Fig. 5.9 Staircase Waveform

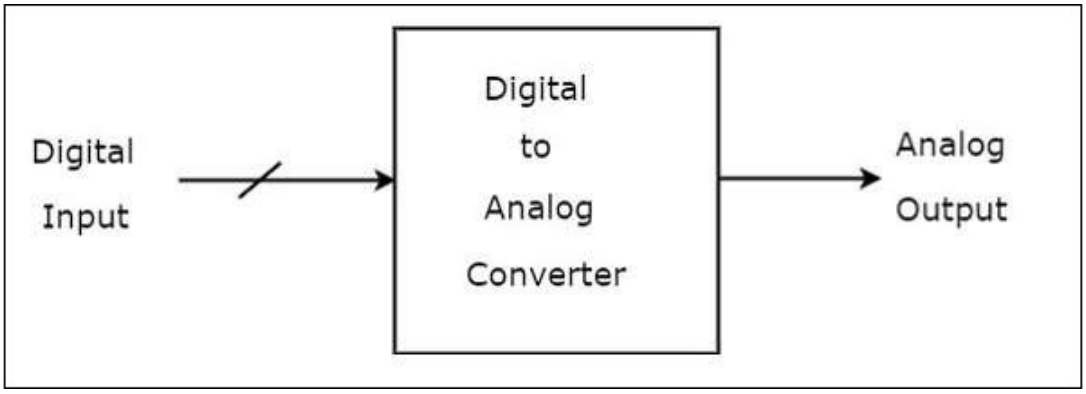
This process continues until the staircase voltage is equal to or slightly greater than the input voltage V_i. At that instant t_2, the output voltage of the input comparator changes state or polarity, so that the gate closes and the counter is stopped.

The display unit shows the result of the count. As each count corresponds to a constant dc step in the DAC output voltage, the number of counts is directly proportional to V_c and hence to V_i. By appropriate choice of reference voltage, the step height of the staircase voltage can be determined. For example, each count can represent 1 mV and direct reading of the input voltage in volts can be realised by placing a decimal point in front of the 10 decade.

Digital to analog Converter:-

A **Digital to Analog Converter (DAC)** converts a digital input signal into an analog output signal. The digital signal is represented with a binary code, which is a combination of bits 0 and 1. This chapter deals with Digital to Analog Converters in detail.

The **block diagram** of DAC is shown in the following figure –



A Digital to Analog Converter (DAC) consists of a number of binary inputs and a single output. In general, the **number of binary inputs** of a DAC will be a power of two.

Types of DACs

There are **two types** of DACs

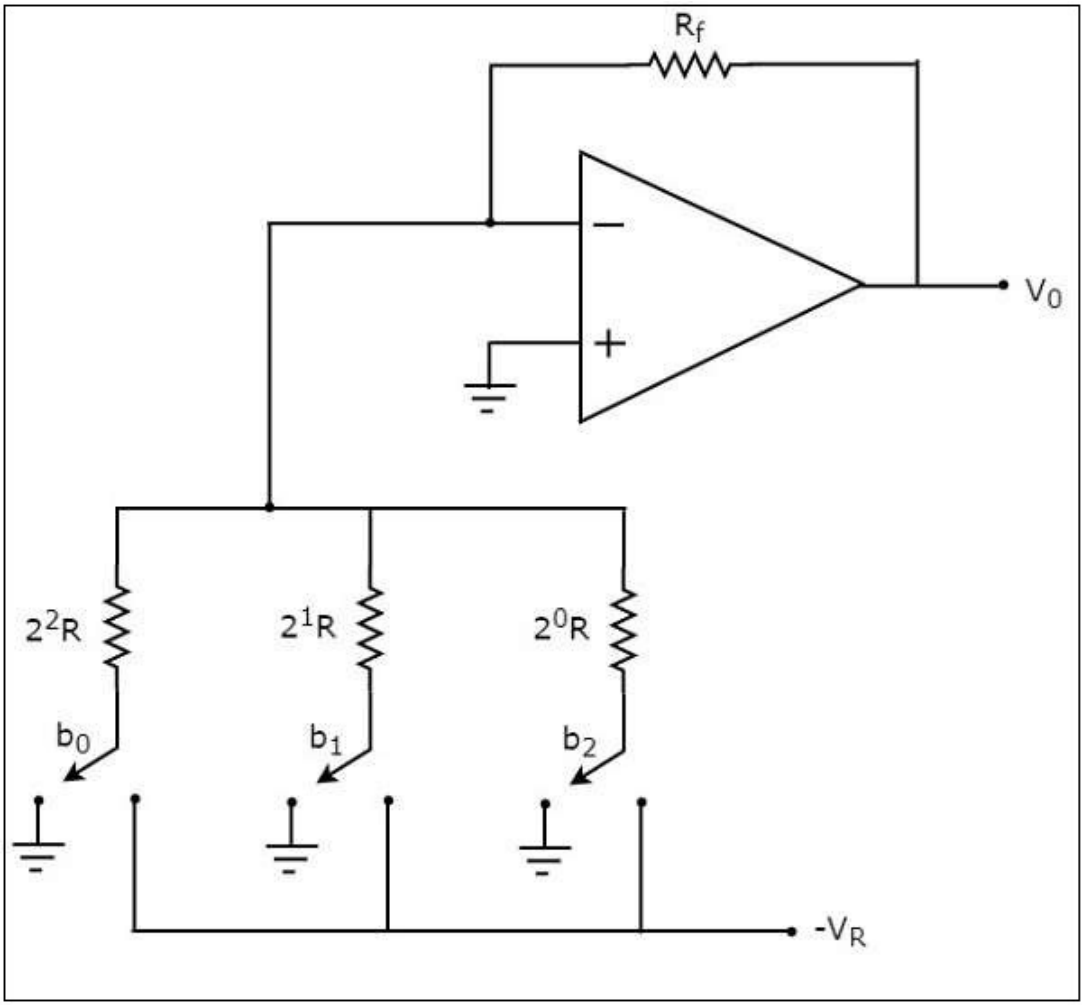
- Weighted Resistor DAC
- R-2R Ladder DAC

This section discusses about these two types of DACs in detail –

Weighted Resistor DAC

A weighted resistor DAC produces an analog output, which is almost equal to the digital (binary) input by using **binary weighted resistors** in the inverting adder circuit. In short, a binary weighted resistor DAC is called as weighted resistor DAC.

The **circuit diagram** of a 3-bit binary weighted resistor DAC is shown in the following figure –



Recall that the bits of a binary number can have only one of the two values. i.e., either 0 or 1. Let the **3-bit binary input** is b2b1b0b2b1b0. Here, the bits b2b2 and b0b0 denote the **Most Significant Bit (MSB) and Least Significant Bit (LSB)** respectively.

The **digital switches** shown in the above figure will be connected to ground, when the corresponding input bits are equal to '0'. Similarly, the digital switches shown in the above figure will be connected to the negative reference voltage, −VR−VR when the corresponding input bits are equal to '1'.

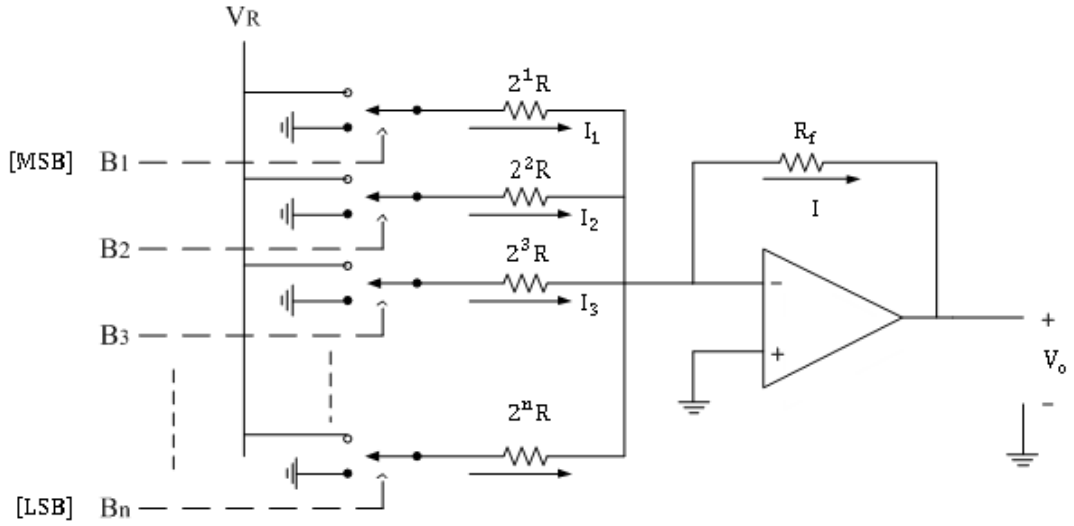
In the above circuit, the non-inverting input terminal of an op-amp is connected to ground. That means zero volts is applied at the non-inverting input terminal of op-amp.

According to the **virtual short concept**, the voltage at the inverting input terminal of opamp is same as that of the voltage present at its non-inverting input terminal. So, the voltage at the inverting input terminal's node will be zero volts.

Binary Weighted Resistor DAC:-

In the weighted resistor type DAC, each digital level is converted into an equivalent analog voltage or current.

The following figure shows the circuit diagram of the binary weighted resistor type DAC.



It consists of parallel binary weighted resistor bank and a feedback resistor Rf. The switch positions decides the binary word (i.e. B1 B2B3...Bn).

In the circuit op-amp is used as current to voltage converter.

Analysis:

Let us analyze the circuit using normal analysis concepts used in op-amp. When the switches are closed the respective currents are flowing through resistors as shown in the circuit diagram above.

Since input current to the op-amp is zero, the addition current flows through feedback resistor.

∴I=I1+I2+I3+ .....+In

The inverting terminal of op-amp is virtually at ground potential.

Disadvantages:

1) When number of binary input increases, it is not easy to maintain the resistance ratio.

2) Very wide ranges of different values of resistors are required.

For high accuracy of conversion, the values of resistances must be accurate.

3) Different current flows through resistors, so their wattage ratings are also different.

4) Accuracy and stability of conversion depends primarily on the absolute accuracy of the resistors and tracking of each other with temperature.

eg. For 10 digit converter

small resistance value = 10 kΩ and

large resistance value = 5.12 MΩ

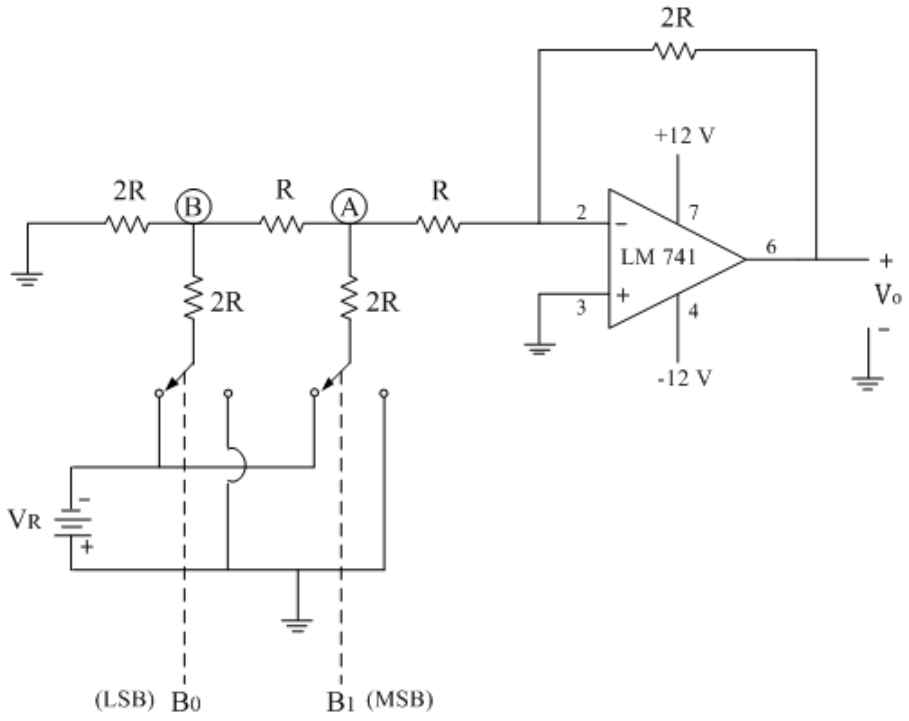
It is very difficult and expensive to obtain stable precise resistances of such value.

5) Since 'R' is very large, op-amp bias currents gives a drop which offsets output.

6) Resistances of switches may be comparable with smallest resistor.

R-2R Ladder DAC:-

The following circuit diagram shows the basic 2 bit R-2R ladder DAC circuit using op-amp. Here only two values of resistors are required i.e. R and 2R. The number of digits per binary word is assumed to be two (i.e. n = 2). The switch positions decides the binary word (i.e. B1 B0)



The typical value of feedback resistor is Rf = 2R. The resistance R is normally selected any value between 2.5 kΩ to 10 kΩ.

The generalized analog output voltage equation can be given as

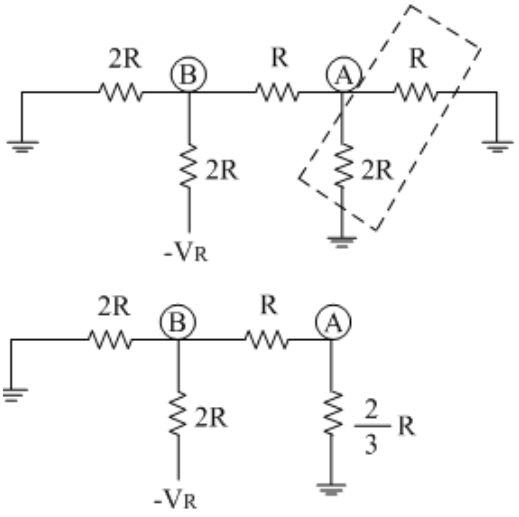
$$V_o = -V_R \frac{R_f}{R}\left[\frac{B_1}{2^1} + \frac{B_2}{2^2} + \frac{B_3}{2^3} + ---- + \frac{B_n}{2^n}\right]$$

$$\therefore V_o = -V_R \frac{R_f}{R \times 2^n}[B_1 2^{n-1} + B_2 2^{n-2} + B_3 2^{n-3} + -- + B_n 2^{n-n}]$$

$$\therefore V_o = -V_R \frac{R_f}{R \times 2^n}[B_1 2^{n-1} + B_2 2^{n-2} + B_3 2^{n-3} + -- + B_n 2^{0}]$$

The operation of the above ladder type DAC is explained with the binary word (B1B0= 01)

The above circuit can be drawn as,



Applying the nodal analysis concept at point (A), we gets following equations

$$\frac{V_A}{\frac{2}{3}R} + \frac{V_A - V_B}{R} = 0$$

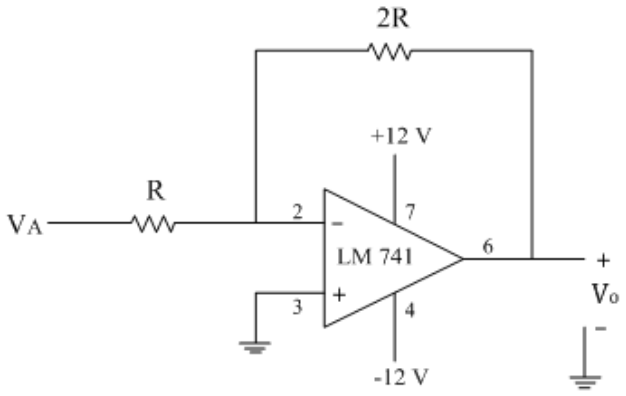
$$\therefore \frac{3V_A}{2R} + \frac{V_A - V_B}{R} = 0$$

$$\therefore \frac{3V_A + 2V_A - 2V_B}{2R} = 0$$

$$\therefore 5V_A = 2V_B$$

$$\therefore V_B = \frac{5V_A}{2}$$

The voltage at point A i.e. VA is applied as input to the op-amp which is in inverting amplifier mode as shown in figure below.



The output voltage of the complete setup

∴Vo=-(2R/R) VA

∴Vo=-(2R/R)(-VR/8)

∴Vo=VR/4

Similarly for other three combinations of digital input the analog output voltage Vo is calculated as follows

Sr. No.	Digital Input		Analog Output, V_o(V)
	B_1	B_0	
01	0	0	0
02	0	1	$\frac{V_R}{4}$
03	1	0	$\frac{2V_R}{4}$
04	1	1	$\frac{3V_R}{4}$

ADC applications	DAC applications
1. Digital Voltmeter: measures voltage in analog and convert to digital form using ADC for digital representation form.	1. Modems need DAC to convert data to analog form so that it can be carried over telephone wires.

2. Mobile phone: Analog voice is converted to digital form for further processing(speech compression, encoding etc.) before it is converted back to analog form for transmission.	2. Video adapters also need DACs known as RAMDACs to convert digital form of data to analog form.
3. Scanner: When we take photo, the scanner uses ADC internally to convert analog information provided by picture into digital information.	3. Digital Motor Control
4. Voice Recorder: It uses ADC to convert analog voice information to the digital information. Latest VOIP solutions utilize the same concept.	4. Printers

Chapter 12.

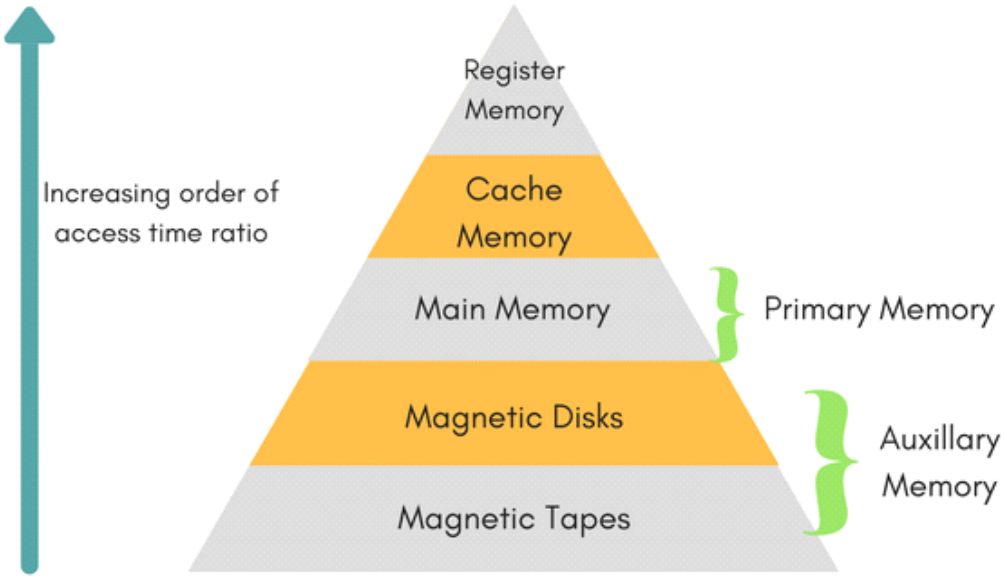
Semiconductor memory

Memory Organization Architecture:-

A memory unit is the collection of storage units or devices together. The memory unit stores the binary information in the form of bits. Generally, memory/storage is classified into 2 categories:

- **Volatile Memory**: This loses its data, when power is switched off.
- **Non-Volatile Memory**: This is a permanent storage and does not lose any data when power is switched off.

Memory Hierarchy



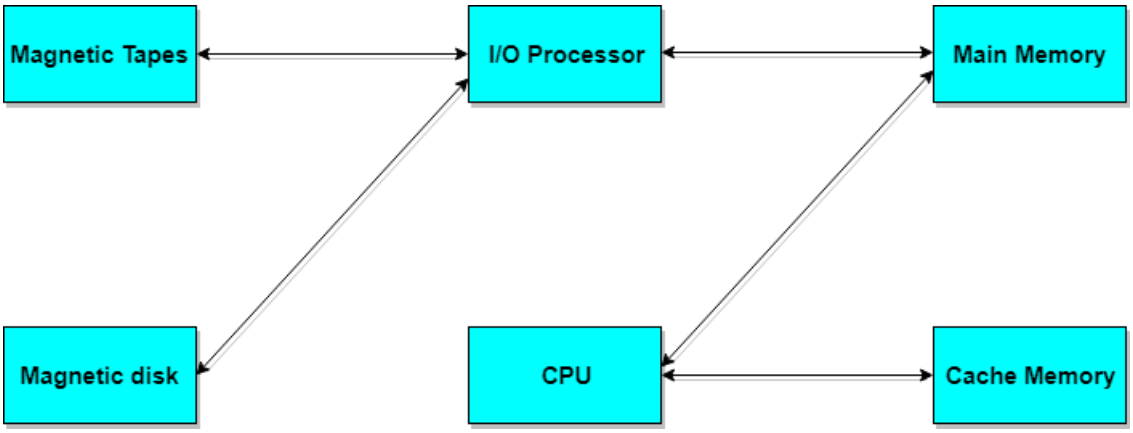
The total memory capacity of a computer can be visualized by hierarchy of components. The memory hierarchy system consists of all storage devices contained in a computer system from the slow Auxiliary Memory to fast Main Memory and to smaller Cache memory.

Auxillary memory access time is generally **1000 times** that of the main memory, hence it is at the bottom of the hierarchy.

The **main memory** occupies the central position because it is equipped to communicate directly with the CPU and with auxiliary memory devices through Input/output processor (I/O).

When the program not residing in main memory is needed by the CPU, they are brought in from auxiliary memory. Programs not currently needed in main memory are transferred into auxiliary memory to provide space in main memory for other programs that are currently in use.

The **cache memory** is used to store program data which is currently being executed in the CPU. Approximate access time ratio between cache memory and main memory is about **1 to 7~10**



Main Memory

The memory unit that communicates directly within the CPU, Auxillary memory and Cache memory, is called main memory. It is the central storage unit of the computer system. It is a large and fast memory used to store data during computer operations. Main memory is made up of **RAM** and **ROM**, with RAM integrated circuit chips holing the major share.

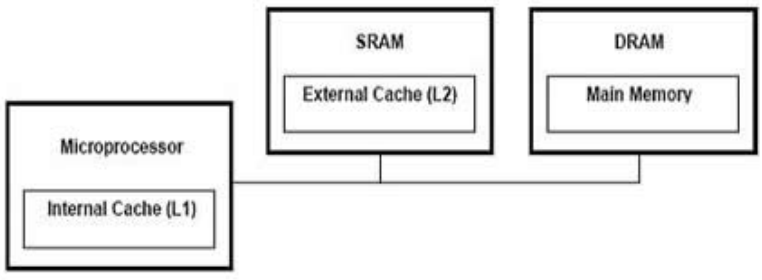
- RAM: Random Access Memory
    - **DRAM**: Dynamic RAM, is made of capacitors and transistors, and must be refreshed every 10~100 ms. It is slower and cheaper than SRAM.
    - **SRAM**: Static RAM, has a six transistor circuit in each cell and retains data, until powered off.
    - **NVRAM**: Non-Volatile RAM, retains its data, even when turned off. Example: Flash memory.
- ROM: Read Only Memory, is non-volatile and is more like a permanent storage for information. It also stores the **bootstrap loader** program, to load and start the operating system when computer is turned on. **PROM**(Programmable

ROM), **EPROM**(Erasable PROM) and **EEPROM**(Electrically Erasable PROM) are some commonly used ROMs.

BASIS FOR COMPARISON	RAM	ROM
Basic	It is a read-write memory.	It is read only memory.
Use	Used to store the data that has to be currently processed by CPU temporarily.	It stores the instructions required during bootstrap of the computer.
Volatility	It is a volatile memory.	It is a nonvolatile memory.
Stands for	Random Access Memory.	Read Only Memory.
Modification	Data in RAM can be modified.	Data in ROM can not be modified.
Capacity	RAM sizes from 64 MB to 4GB.	ROM is comparatively smaller than RAM.
Cost	RAM is a costlier memory.	ROM is comparatively cheaper than RAM.

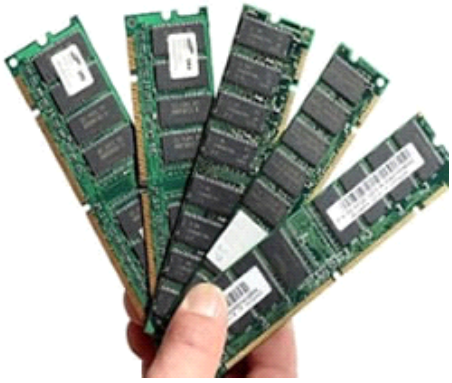
Difference between Dynamic and Static RAM:-

Key Difference: A dynamic RAM chip holds millions of memory cells, each made up of a transistor and a capacitator. The chip constantly needs to be refreshed. Static RAM differs as it holds information in a flip flop manner, which means it does not require to constantly refresh and do not use capacitators.



Random Access Memory (RAM) is a form of memory that is used by computers in order to store information. Dynamic and static RAM are two types of RAMs that is simultaneously used by the computer in order to store an access data.

Dynamic RAM is the most commonly used RAM and is also considerably cheaper, but even static RAM has benefits. A dynamic RAM chip holds millions of memory cells, each made up of a transistor and a capacitator. The capacitator stores electrons in computer memory cells and is responsible for holding information. In dynamic RAM, information is held as a charge in 1 or 0, where 1 means it has information and 0 means it is empty.



The transistor acts as a switch that lets the control circuitry on the memory chip read the capacitor or change its state. The problem in dynamic RAM is that capacitator leaks energy and can only hold a charge for a fraction of a second. Hence, it needs to constantly refresh the circuit to read the charge. The refresh happens hundreds of times every second and all the cells must be read over and over again to see if the information is there. So, at every charge the computer reads and re-writes each bit of information.

Static RAM differs as it holds information in a flip flop manner, which means it does not require to constantly refresh and do not use capacitators. The computer accesses the information as required, which makes them more energy efficient. However, they have very limited storage capacity.

.

Comparison between Dynamic and Static RAM:

	Dynamic RAM	**Static RAM**
Transistors	Requires 3-4 transistors	Requires 6-8 transistors
Memory Refresh	Memory can be deleted and refreshed while running the program	Memory cannot be deleted and refreshed while running programs
Storage	Data is stored as a charge in a capacitor	Data is stored in flip flop level

Space	Takes up less space	Requires more space
System	Is used to create larger RAM space system	Is used to create speed-sensitive cache
Expensive	Is cheaper	Is more expensive
Power	Consumes less power	Consumes more power
Time	Requires more time to access stored data	Requires less time to access stored data
Storage Capacity	Has higher storage capacity	Has less storage capacity

74181 ALU:-

The **74181** is a 4-bit slice arithmetic logic unit (ALU), implemented as a 7400 series TTL integrated circuit. The first complete ALU on a single chip,[1] it was used as the arithmetic/logic core in the CPUs of many historically significant minicomputers and other devices.

The 74181 represents an evolutionary step between the CPUs of the 1960s, which were constructed using discrete logic gates, and today's single-chip CPUs or microprocessors. Although no longer used in commercial products, the 74181 is still referenced in computer organization textbooks and technical papers. It is also sometimes used in 'hands-on' college courses, to train future computer architects.

Significance:-

Although the 74181 is only an ALU and not a complete microprocessor, it greatly simplified the development and manufacture of computers and other devices that required high speed computation during the late 1960s through the early 1980s, and is still referenced as a "classic" ALU design.

Prior to the introduction of the 74181, computer CPUs occupied multiple circuit boards and even very simple computers could fill multiple cabinets. The 74181 allowed an entire CPU and in some cases, an entire computer to be constructed on a single large printed circuit board. The 74181 occupies a historically significant stage between older CPUs based on discrete logic functions spread over multiple circuit boards and

modern microprocessors that incorporate all CPU functions in a single component. The 74181 was used in various minicomputers and other devices beginning in the 1970s, but as microprocessors became more powerful the practice of building a CPU from discrete components fell out of favor and the 74181 was not used in any new designs.