UNIT-1

Introduction: Analog and Digital Signals

Analog and digital signals are the types of **signals** carrying information.

We live in an analog world. There are an infinite amount of colors to paint an object (even if the difference is indiscernible to our eye), there are an infinite number of tones we can hear, and there are an infinite number of smells we can smell. The common theme among all of these analog signals is their **infinite** possibilities.

Digital signals and objects deal in the realm of the **discrete** or **finite**, meaning there is a limited set of values they can be. That could mean just two total possible values, 255, 4,294,967,296, or anything as long as it's not ∞ (infinity).

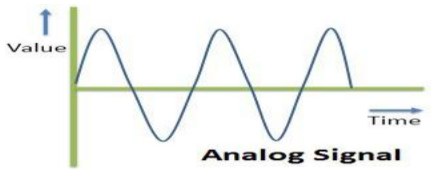
Working with electronics means dealing with both analog and digital signals, inputs and outputs. Our electronics projects have to interact with the real, analog world in some way, but most of our microprocessors, computers, and logic units are purely digital components.

The physical world is analog, but digital techniques are often far superior than analog techniques in the context of electronic design.

Definition of Analog Signal

Analog signal is a kind of continuous wave form that changes over time. An analog signal is further classified into simple and composite signals. A simple analog signal is a sine wave that cannot be decomposed further. On the other hand, a composite analog signal can be further decomposed into multiple sine waves.

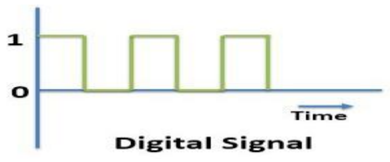
An analog signal is described using amplitude, period or frequency and phase. Amplitude marks the maximum height of the signal. Frequency marks the rate at which signal is changing. Phase marks the position of the wave with respect to time zero.



An analog signal is not immune to noise hence; it faces distortion and decrease the quality of transmission. The range of value in an analog signal is not fixed.

Definition of Digital Signal

Digital signals also carry information like analog signals but is somewhat is different from analog signals. Digital signal is non continuous, discrete time signal. Digital signal carries information or data in the binary form i.e. a digital signal represent information in



the form of bits. Digital signal can be further decomposed into simple sine waves that are called harmonics. Each simple wave has different amplitude, frequency and phase. Digital signal is described with bit rate and bit interval. Bit interval describes the time require for sending a single bit. On the other hand, bit rate describes the frequency of bit interval.

A digital signal is more immune to the noise; hence, it hardly faces any distortion. Digital signals are easier to transmit and are more reliable when compared to analog signals. Digital signal has a finite range of values. The digital signal consists 0s and 1s.

Key Differences Between Analog and Digital Signal

1. An analog signal represents a continuous wave that keeps changing over a time period. On the other hand, a digital signal represents a noncontiguous wave that carries information in a binary format and has discrete values.
2. An analog signal is always represented by the continuous sine wave whereas, a digital signal is represented by square waves.
3. While talking of analog signal we describe the behavior of the wave in respect of amplitude, period or frequency, and phase of the wave. On the other hand, while talking of discrete signals we describe the behavior of the wave in respect of bit rate and bit interval.
4. The range of an analog signal is not fixed whereas the range of the digital signal is finite and which can be 0 or 1.
5. An analog signal is more prone to distortion in response to noise, but a digital signal has immunity in response to noise hence it rarely faces any distortion.
6. An analog signal transmits data in the form of wave whereas, a digital signal transmits the data in the binary form i.e. in the form of bits.
7. The best example of an analog signal is a human voice, and the best example of a digital signal is the transmission of data in a computer.

Difference Between Analog and Digital Signal

Vs
0100111101
Analog Signal
Digital Signal

Analog and Digital are the different forms of signals. Signals are used to carry information from one device to another. Analog signal is a continuous wave that keeps on changing over a time period. Digital signal is discrete in nature.

The fundamental difference between analog and digital signal is that analog signal is represented by the sine waves whereas; the digital signal is represented by square waves. Lets us learn some more differences between analog and digital signal with the help of comparison chart shown below.

Comparison Chart

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.
Representation	An analog signal is represented by a sine wave.	A digital signal is represented by square waves.
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 transmits data in the form of a wave.	A digital signal carries data in the binary form i.e. 0 and 1.
Example	The human voice is the best example of an analog signal.	Signals used for transmission in a computer are the digital signal.

Important points:

- All real life signals are analog in nature and at first sight, it seems use of analog is much better as compared to digital signal. But digital signals have various advantages over the analog signal like noise immunity.
- Digital signal is used in communication process to minimize the effect of noise
- Digital Signals carry more information per unit time as compared to analog signals
- Quality of digital signal is better over long distance transmission
- Use of bandwidth is less in case of transmission using digital signals
- We can encrypt digital signals
- Noise removal is easy in digital signals
- Unwanted signals are called noise

Differences between Digital and Analog System

Digital as well as Analog System, both are used to transmit signals from one place to another like audio/video. Digital system uses binary format as 0 and 1 whereas analog system uses electronic pulses with varying magnitude to send data.

Following are some of the important differences between Digital System and Analog System.

Sr. No.	Key	Digital System	Analog System
1	Signal Type	Digital System uses discrete signals as on/off representing binary format. Off is 0, On is 1.	Analog System uses continuous signals with varying magnitude.
2	Wave Type	Digital System uses square waves.	Analog system uses sine waves.
3	Technology	Digital system first transform the analog waves to limited set of numbers and then record them as digital square waves.	Analog systems records the physical waveforms as they are originally generated.
4	Transmission	Digital transmission is easy and can be made noise proof with no loss at all.	Analog systems are affected badly by noise during transmission.
5	Flexibility	Digital system hardware can be easily modulated as per the requirements.	Analog system's hardware’s are not flexible.
6	Bandwidth	Digital transmission needs more bandwidth to carry same information.	Analog transmission requires less bandwidth.
7	Memory	Digital data is stored in form of bits.	Analog data is stored in form of waveform signals.
8	Power requirement	Digital system needs low power as compare to its analog counterpart.	Analog systems consume more power than digital systems.
9	Best suited for	Digital systems are good for computing and digital electronics.	Analog systems are good for audio/video recordings.
10	Cost	Digital systems are costly.	Analog systems are cheap.
11	Example	Digital systems are: Computer, CD, DVD.	Analog systems are: Analog electronics, voice radio using AM frequency.

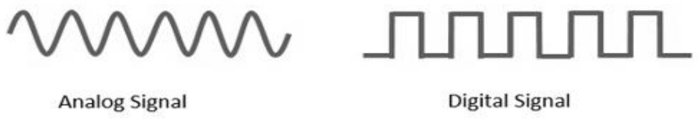
The Necessity of Digitization

The communication that occurs in our day-to-day life is in the form of signals. These signals, such as sound signals, generally, are analog in nature. When the communication needs to be established over a distance, then the analog signals are sent through wire, using different techniques for effective transmission.

The conventional methods of communication used analog signals for long distance communications, which suffer from many losses such as distortion, interference, and other losses including security breach.

In order to overcome these problems, the signals are digitized using different techniques. The digitized signals allow the communication to be more clear and accurate without losses.

The following figure indicates the difference between analog and digital signals. The digital signals consist of **1s** and **0s** which indicate High and Low values respectively.



Representation of Signals

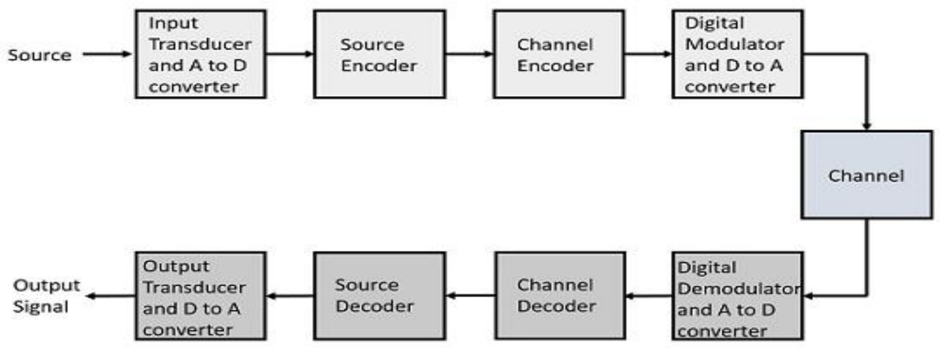
Advantages of Digital Communication

As the signals are digitized, there are many advantages of digital communication over analog communication, such as −

- The effect of distortion, noise, and interference is much less in digital signals as they are less affected.
- Digital circuits are more reliable.
- Digital circuits are easy to design and cheaper than analog circuits.
- The hardware implementation in digital circuits, is more flexible than analog.
- The occurrence of cross-talk is very rare in digital communication.
- The signal is un-altered as the pulse needs a high disturbance to alter its properties, which is very difficult.
- Signal processing functions such as encryption and compression are employed in digital circuits to maintain the secrecy of the information.
- The probability of error occurrence is reduced by employing error detecting and error correcting codes.
- Spread spectrum technique is used to avoid signal jamming.
- Combining digital signals using Time Division Multiplexing TDMTDM is easier than combining analog signals using Frequency Division Multiplexing FDM
- he configuring process of digital signals is easier than analog signals.
- Digital signals can be saved and retrieved more conveniently than analog signals.
- Many of the digital circuits have almost common encoding techniques and hence similar devices can be used for a number of purposes.
- The capacity of the channel is effectively utilized by digital signals.

Elements of Digital Communication

The elements which form a digital communication system is represented by the following block diagram for the ease of understanding.


Source
Input Transducer and A to D converter
Source Encoder
Channel Encoder
Digital Modulator and D to A converter
Channel
Digital Demodulator and A to D converter
Channel Decoder
Source Decoder
Output Transducer and D to A converter
Output Signal

Basic Elements of a Digital Communication System

Following are the sections of the digital communication system.

Source

The source can be an **analog** signal. **Example**: A Sound signal

Input Transducer

This is a transducer which takes a physical input and converts it to an electrical signal (**Example**: microphone). This block also consists of an **analog to digital** converter where a digital signal is needed for further processes.

A digital signal is generally represented by a binary sequence.

Source Encoder

The source encoder compresses the data into minimum number of bits. This process helps in effective utilization of the bandwidth. It removes the redundant bits unnecessaryexcessbits,i.e.,zeroesunnecessaryexcessbits,i.e.,zeroes.

Channel Encoder

The channel encoder, does the coding for error correction. During the transmission of the signal, due to the noise in the channel, the signal may get altered and hence to avoid this, the channel encoder adds some redundant bits to the transmitted data. These are the error correcting bits.

Digital Modulator

The signal to be transmitted is modulated here by a carrier. The signal is also converted to analog from the digital sequence, in order to make it travel through the channel or medium.

Channel

The channel or a medium, allows the analog signal to transmit from the transmitter end to the receiver end.

Digital Demodulator

This is the first step at the receiver end. The received signal is demodulated as well as converted again from analog to digital. The signal gets reconstructed here.

Channel Decoder

The channel decoder, after detecting the sequence, does some error corrections. The distortions which might occur during the transmission, are corrected by adding some redundant bits. This addition of bits helps in the complete recovery of the original signal.

Source Decoder

The resultant signal is once again digitized by sampling and quantizing so that the pure digital output is obtained without the loss of information. The source decoder recreates the source output.

Output Transducer

This is the last block which converts the signal into the original physical form, which was at the input of the transmitter. It converts the electrical signal into physical output (**Example**: loud speaker).

Output Signal

This is the output which is produced after the whole process. **Example** − The sound signal received.

This unit has dealt with the introduction, the digitization of signals, the advantages and the elements of digital communications.

UNIT-2

NUMBER SYSTEM

INTRODUCTION

A digital computer is also known as data processor. It processes data while solving a mathematical problem or doing translation from one language to another etc. Before a computer can process data, the data has to be converted into a form acceptable to a computer.

We use decimal number system in our work. This system has digits 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. Computers cannot use these numbers. Instead, a computer works on binary digits. A binary number system has only two digits 0 and 1. This is because of the reason that computers use integrated circuits with thousands of transistors. Due to variation of parameters the behavior of transistor can be very erratic and quiescent point may shift from one position to another. Nevertheless the cut off and saturation points are fixed. When a transistor is cut off, a large change in values is needed to change the state to saturation Similar is the situation when it is in saturation. Thus a transistor is a very reliable two state device. One state represents digit 0 and the other state represents digit 1. All input voltages are recognized as either 0 or 1.

DECIMAL NUMBER SYSTEM

We are all familiar with the decimal number system. It uses ten digits (0, 1, 2, 3, 4, 5, 6, 7, 9) and thus its base is 10. The decimal number system of counting was evolved because we have 8 fingers and 2 thumbs on our two hands so that we can count 10. By using the different digits in different positions we can express any number. For numbers bigger than 9 we use two or more digits. The position of each digit in the number indicates the magnitude that this number represents. In the number 27 the digit 7 represents 7 X 10^0 or 7 and the digit 2 represents 2 X 10^1 or 20. The sum of 7 and 20 makes 27. Similarly the number 263 can be expressed as

Decimal 263 = (2 X 10^2) + (6 X 10^1) + (3 X 10^0) = 200 + 60 + 3 = 263.

Since the base in decimal number system is 10, the number 263 can be written as 263_{10}. The suffix 10 emphasizes the fact that the base is 10.

BINARY NUMBER SYSTEM

The binary number system has only two digits 0 and 1. Thus a binary number is a string of zeros and ones. Since it has only two digits, the base is 2.
The abbreviation of binary digit is bit. The binary number 1100 has 4 bits, 101011 has 6 bits and 11001010 has 8 bits. Each bit may represent either 0 or 1. A string of 8 bits is known as a byte. A byte is the basic unit of data in computers. In most computers, the data* is processed in strings of 8 bits or some multiples (i.e., 16, 24, 32 etc.). The computer memory also stores data in strings of 8 bits or multiples of 8 bits. Table 2.1 shows the 16 combinations of a 4 bit binary word.

Table 2.1. Binary, decimal, hexadecimal and octal equivalence

Binary				**Decimal**	**Hexadecimal**	**Octal**
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	2	2	2
0	0	1	1	3	3	3
0	1	0	0	4	4	4
0	1	0	1	5	5	5
0	1	1	0	6	6	6
0	1	1	1	7	7	7
1	0	0	0	8	8	10
1	0	0	1	9	9	11
1	0	1	0	10	A	12
1	0	1	1	11	B	13
1	1	0	0	12	C	14
1	1	0	1	13	D	15
1	1	1	0	14	E	16
1	1	1	1	15	F	17

As in the decimal system the binary number system is positionally weighted. The digital on the extreme right hand side has a weight of 2^0, the next one has a weight of 2^1 and so on. Since the base is 2, the binary number is written as (say) 1011_2. The suffix 2 emphasizes the fact that base is 2. If the number of binary digits is n, the highest decimal number which can be counted is 2^n-1. Thus with 4 binary digits we can count upto (2^4 - 1) or 15 decimal number. If n = 8 we can count upto (2^8 - 1) = 255 decimal number.

Binary to Decimal Conversion

The procedure to convert a binary number to decimal is called dibble□dabble method. We start with the left hand bit. Multiply this value by 2 and add the next bit. Again multiply by 2 and add the next bit. Stop when the bit on extreme right hand side is reached.
An other fast and easy method to convert binary number to decimal number is as under:

1. Write the binary number.
2. Write the weights 2^0, 2^1, 2^2, 2^3 etc., under the binary digits starting with the bit on right hand side.
3. Cross out weights under zeros.
4. Add the remaining weights.

Example 2.1. Convert 100101_2 to decimal.

Solution : Left hand bit	1
Multiply by 2 and add next bit 2 X 1 + 0 =	2
Multiply by 2 and add next bit 2 X 2 + 0 =	4
Multiply by 2 and add next bit 2 X 4 + 1 =	9

Multiply by 2 and add next bit 2 X 9 + 0 = 18

Multiply by 2 and add next bit 2 X 18 + 1 = 37

Therefore, $100101_2 = 37_{10}$

Example 2.2 Convert 1101_2 into equivalent decimal number.

Solution :	1	1	0	1	Binary number
	8	4	2	1	Write weights

Cross out weights under zeros

8 + 4 + 0 + 1 = 13 Add weights Therefore, $1101_2 = 13_{10}$

Example 2.3. Convert 110011011001_2 into equivalent decimal number.

Solution :

1	1	0	0	1	1	0	1	1	0	0	1	Binary number
2048	1024	512	256	128	64	32	16	8	4	2	1	Write weights
2048	1024	512	256	128	64	32	16	8	4	2	1	

zeros

2048 + 1024 + 128 + 64 + 16 + 8 + 1 = 3289 Add weights

Thus $110011011001_2 = 3289_{10}$

Decimal to Binary Conversion

A systematic way to convert a decimal number into equivalent binary number is known as double dabble. This method involves successive division by 2 and recording the remainder (the remainder will be always 0 or 1). The division is stopped when we get a quotient of 0 with a remainder of 1. The remainders when read upwards give the equivalent binary number.

Example 2.4 Convert decimal number 10 into its equivalent binary number.

2		
	5	remainder 0
2	2	remainder 1
2	1	remainder 0
0	0	remainder 1

The binary number is 1010.

Example 2.5. Convert decimal number 25 into its binary equivalent.

Solution :

2	25
	0 remainder 1

The binary number is 11001.

BINARY ARITHMETIC

The rules for addition of binary numbers are as under :

0 + 0 = 0;

0 + 1 = 1 + 0 = 1

1 + 1 = 10 i.e., 1 + 1 equals 0 with a carry of 1 to next higher column

1 + 1 + = 1

column.

Binary Subtraction

The rules for subtraction of binary numbers are as under :

0 - 0 = 0

1 - 0 = 1

1 - 1 = 0

10- 1 = 1

In both the operations of addition and subtraction, we start with the least significant bit (L.S.B.), i.e., the bit on the extreme right hand side and proceed to the left (as is done in decimal addition and subtraction).

Example 2.6. (a) Convert decimal numbers 15 and 31 into binary numbers. (b) Add the binary numbers and convert the result into decimal equivalent.
Solution : (a)

2			
1		7	remainder 1
1	2	3	remainder 1
1	2		
			remainder 1

	31	
2	15	Remainder 1
2	7	Remainder 1
2	3	Remainder 1
2	1	Remainder 1
2	0	Remainder 1

Binary number is 1111

Binary Addition

		1	1	1	1
	1	1	1	1	1
1	0	1	1	1	0

The sum of binary numbers 1111 and 11111 is the binary number 101110

1	0	1	1	1	0	Binary number
32	16	8	4	2	1	Write weights
32	16	8	4	2	1	Cross cut weights under zero

32 + 8 + 4 + 2 = 46 Add weights

Example 2.7. Subtract 10001 from 11001.

Solution :	1	1	0	0	1	25
	(-1) 1	0	0	0	1	(-) 17

~~**Results** 1 0 0 0~~ ~~8~~

Example 2.8. Subtract 0111 from 1010.

Solution :	1	0	1	0		10
(-1)	0	1	1	1	(-1)	7
	~~0~~	~~0~~	~~0~~	~~1~~		~~3~~

The least significant digit.□ in the first number is 0. So we borrow 1 from the next digit and subtract 1 to give 1. Now in the second column we have 0, so we again borrow 1 from the next higher column and subtract 1 to give 1. In the third column, we borrow 1 from the next higher column and 1-1 gives 0. In the fourth column, 0 (after lending) - 0 gives 0.

Binary Multiplication

The four basic rules for binary multiplication are : 0 X 0 = 0
0 X 1 = 0

1 X 0 = 0

1 X 1 = 1

- The digit on the extreme RHS is the LSB and the digit on extreme LHS is the MSB (most significant digit)

The method of binary multiplication is similar to that in decimal multiplication. The method involves forming partial products, shifting successive partial products left one place and then adding all the partial products.

Binary Division

The division in binary system follows the same long division procedure as in decimal

system.

Example 2.9. (a) Divide 110110_2 by 101.

(b) Convert 110110_2 and 101_2 into equivalent decimal number obtain division, convert results into binary and compare the results with those in part (a).

Solution : (a) 101) 10 1 1 0 (1010 quotient

1

1 0 1

1 1 1

1 0 1

1 0 0 remainder

(b)	1	1	0	1	1	0	Binary number
	32	16	8	4	2	1	Write weights
	32	16	6	4	2	1	Cross out weights under zero

32 + 16 + 4 + 2 = 54 Add weights

1 0 1 Binary number

4 2 1 Write weights

4 2 1 Cross out weights under zero 4 + 1 = 5 Add weights

5 | 54

10 – 4

quotient

2	10	
2	5	remainder 0
2	2	remainder 1
2		
	0	remainder 1

remainder

2	4	
2	2	remainder 0
2		
	0	remainder 1

The quotient is 1010_2 and the remainder is 100_2. These are the same as in part (a)

SIGNED BINARY NUMBERS

To represent negative numbers in the binary system, digit 0 is used for the + sign and 1 for the -ve sign. The most significant bit is the sign bit followed by the magnitude bits. Numbers expressed in this manner are known as signed binary numbers. The numbers may be written in 4 bits, 8 bits, 16 bits, etc. In every case, the leading bit represents the sign and the remaining bits represent the magnitude.

Example 2.10 Express in 16-bit signed binary system : (a) + 8, (b) -8, (c) 165, (d) -165.

Solution : (a)

2	8	
2	4	remainder 0
2	2	remainder 0
2		
		remainder 1

The binary number is 1000.

For the 16 bit system, we use 16 bits, 0 (which stands for +) in the leading position, 1000 in the last 4 bits and 0 in the remaining 11 positions. So the signed 16 bit binary number is
+ 8 = 0000 0000 0000 1000

(b) In the leading bit we will have 1 (to represent the '-' sign). The rest of the representation is the same s in part (a).

- Signed binary numbers are also known as sign-magnitude numbers.

- 8 = 1000 0000 0000 1000

2	165	
2		
	0	remainder 1

So the number is 10100101.

Using 0 in the leading bit (for + sign) the 16 bit signed binary number is

+ 165 = 0000 0000 1010 0101

(d) In the leading bit position we will have 1 (for the '-' sign). Therefore, - 165 = 1000 0000 1010 0101

1'S COMPLEMENT

The 1's complement of a binary number is obtained by complementing each bit (i.e., 0 for 1 and 1 for 0).
Thus each bit in the original word is inverted to give the 1's complement. For example, for the number

	1 1 0 0	1 0 0 1
1's complement is	0 0 1 1	0 1 1 0

2'S COMPLEMENT

The signed binary numbers required too much electronic circuitry for addition and subtraction. Therefore, positive decimal numbers are expressed in sign-magnitude form but negative decimal numbers are expressed in 2's complements.
2's complement is defined as the new word obtained by adding 1 to 1's complement*
e.g., Let A = 0 1 0 1 i.e., 5

- 1's complement of A is denoted by A and 2's complement of A is denoted by A'.

$$\begin{array}{lcccccc} \text{1's complement} & A & = & 1 & 0 & 1 & 0 \\ & & + & & & & 1 \\ \hline \text{2's complement} & A' & = & 1 & 0 & 1 & 1 \quad \text{i.e., } -5 \end{array}$$

Taking the 2's complement is the same as changing the sign of the given binary number. If we take the 2's complement twice we get the original number, e.g.,

$$\begin{array}{lcccccc} & A' & = & 1 & 0 & 1 & 1 \\ \text{1's complement} & A & = & 0 & 1 & 0 & 0 \\ & & + & & & & 1 \\ \hline \text{2's complement} & A'' & = & 0 & 1 & 0 & 1 \quad = A \end{array}$$

Thus A'' = A. In view of this every number and its 2's complement form a complementary pair. In a typical computer positive numbers are expressed in sign magnitude form but negative numbers are expressed as 2's complements. The positive numbers have a leading sign bit of 0 and negative numbers have a leading sign bit of 1.

2'S COMPLEMENT ADDITION, SUBTRACTION

The use of 2's complement representation has simplified the computer hardware□ for arithmetic operations. When A and B are to be added, the B bits are not inverted so that we get

S = A + B ...(2.1)

When B is to be subtracted from A, the computer hardware forms the 2's complement of B and then adds it to A. Thus
S = A + B+' = A + (-B) = A - B

Eqns. (2.1 and 2.2) represent algebraic addition and subtraction. A and B may represent either positive or negative numbers. Moreover, the final carry has no significance and is not used.

BINARY FRACTIONS

So far we have discussed only whole numbers. However, to represent fractions is also important. The decimal number 2568 is represented as

2568000 + 500 + 60 + 8 = 2 X 10^3 + 5 X 10^2 + 6 X 10^1 + 8 X 10^0

- Hardware means electronic, mechanical and magnetic devices in a computer. The computer program is known as software.

Similarly, 25.68 can be represented as

25.68 = 20 + 5 + 0.6 + 0.08 = 2 X 10^1 + 5 X 10^0 + 6 X $10^{\square 1}$ + 8 X $10^{\square 2}$

i. Conversion of Binary to Decimal

In the binary system, the weights of the binary bits after the binary point, can be written as

0.1011 = 1 X 2^{-1} + 0 X 2^{-2} + 1 X 2^{-3} + 1 X 2 =$^{-4}$

$$= 1 \text{ X } \frac{1}{2} + 0 \text{ X } \frac{1}{4} + 1 \text{ X } \frac{1}{8} + 1 \text{ X } \frac{1}{16}$$

= 0.5 + 0 + 0.125 + 0.0625 = 0.6875 (decimal)

Example 2.11 Express the number 0.6875 into binary equivalent

Solution :

Fraction	Fraction X 2	Remainder new fraction	Integer
0.6875	1.375	0.375	1 (MSB)
0.375	0.75	0.75	0
0.75	1.5	0.5	1
0.5	1	0	1(LSB)

The binary equivalent is 0.1011.

b. DOUBLE PRECISION NUMBERS

Most of the computers used in today's world are 16 bit or more. In these computers the numbers from +32, 767 to - 32,768 can be stored in each register. To store numbers greater than these numbers, double precision system is used. In this method two storage locations are used to represent each number. The format is

First word

S	High order bits

Second word

O	Low order bits

S is the sign bit and O is a zero. Thus numbers with 31 bit length can be represented

in 16 bit registers. For still bigger numbers triple precision can be used. In triple precision 3 word lengths (each 16 bit) is used to represent each number.

c. FLOATING POINT NUMBERS

Most of the time we use very small and very large numbers, e.g., $1.02X10^{-12}$ and

$6.5 \text{ X } 10^{+17}$. In binary representation, the numbers are expressed by using a mantissa and an exponent.
The mantissa has a 10 bit length and exponent has 6 bit length. Fig. 2.1 shows one such representation.
Mantissa Exponent

0 1 1 1 0 0 1 1 0	1 0 0 1 1 1

The left most bit of mantissa is sign bit. The binary point is to the right of this sign bit

The 6 bit exponent has a base of 2. The exponent can represent numbers 0 to 63.To express negative exponents the number 32_{10} (i.e., 100000_2) has been added to the exponent. It is known as excess -32 notation and is a common floating point format. Examples of exponent in excess - 32 format are
Table 2.2

Actual exponent	**Binary representation in Excess-32 format**
- 32	000000
-1	011111
0	100000
+7	100111
+15	101111
+31	111111

The number represented in Table 2.2 is Mantissa + 0.111001101

Exponent 100111

Subtracting 100000 from exponent, we get 000111. The number is $0.111001101_2 \text{ X } 2^7 = 1110011.01_2 = 115.25_{10}$

Example 2.12. What does the floating point number 01101000000010101 represent.

Solution : Mantissa is +0. 110100000

Exponent is 010101

Subtracting 100000 from exponent, we get 110110101.

The given number is $+ 0.110100000 \text{ X } 2^{-11} = + 0.00000000000110100000_2$

$= + 0.000396728_{10}$

The advantage of floating point representation is that very large and very small numbers can be easily expressed. Since the above representation uses 10 bit long mantissa, the accuracy in above representation is 9 bit since 1 bit is used for sign). Fixed point 16 bit numbers are accurate to 15 bits. Thus breaking the 16 bit lengths into mantissa and exponent (to use floating point representation) reduces the accuracy to some extent. To ensure maximum accuracy the computers normalize the result of any floating point operation. In this process the most significant bit is placed next to the sign bit.

Example 2.13. Add the binary numbers 1 0 1 1 0 1. 0 1 0 1 and 1 0 0 0 1. 1 0 1.

```
Solution:   1 0 1 1 0 1 . 0 1   0 1
          +   1 0 0 0 1 . 1 0   1
            ----------------------
            1 1 1 1 1 0 . 1 1   1 1
```

Example 2.14. Convert the binary number 1 1 0 0 1. 0 0 1 0 1 1 to decimal.

Solution : The decimal equivalent is obtained as under

1 1 0 0 1 . 0 0 1 0 1 1

2^4 2^3 2^2 2^1 2^0 . 2^{-1} 2^{-2} 2^{-3} 2^{-4} 2^{-5} 2^{-6}
weights

Decimal equivalent = 1×2^4 $1 \times 2^3 + 1 \times 2^0 + 1 \times 2^{-3} + 1 \times 2^{-5} + 1 \times 2^{-6}$

= 16 + 8 + 1 + 0.125 + 0.03125 + 0.015625 = 25.171875.

OCTAL NUMEBR SYSTEM

The number system with base (or radix) eight is known as the octal number system. In this system, eight symbols, 0, 1, 2, 3, 4, 5, 6 and 7 are used to represent numbers. Similar to decimal and binary number systems, it is also a positional system and has, in general, two parts: integer and fractional, set apart by a radix (octal) point (.). For example, $(6327.4051)_8$ is an octal number. Using the weights it can be written as
$(6327.4051)_8 = 6 \times 8^3 + 3 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + .4 \times 8^{-1}$

$+ 0 \times 8^{-2} + 5 \times 8^{-3} + 1 \times 8^{-4}$

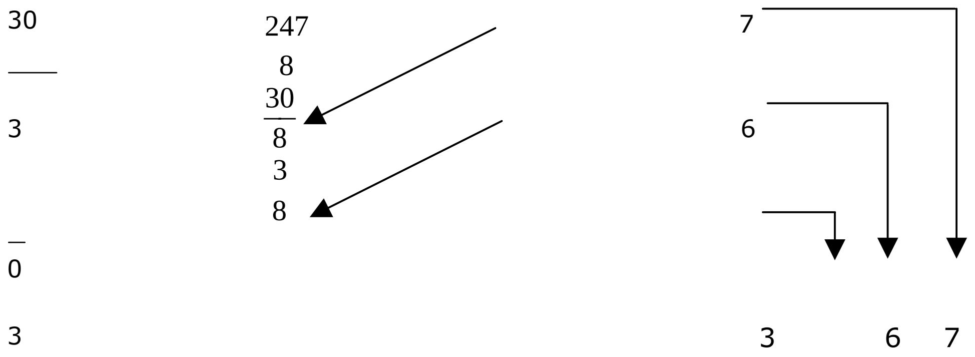
$= 3072 + 192 + 16 + 7 + \frac{4}{8} + 0 + \frac{5}{512} + \frac{1}{4096}$
$= (3287.5100098)_{10}$

Thus, $(6327.4051)_8 = (3287.5100098)_{10}$

Using the above procedure, an octal number can be converted into an equivalent decimal number or a base -8 number can be converted into an equivalent base-10 number.
The conversion from decimal to octal (base-10 to base-8) is similar to the conversion procedure for base-10 to base-2 conversion. The only difference is that number 8 is used in place of 2 for division in the case of integers and for multiplication in the case of fractional numbers.

Example 2.15. Convert $(247)_{10}$ into octal

Solution (a)



Quotient | Remainder
30 — 247/8 — 7
3 — 30/8 — 6
0 — 3/8 — 3

3 6 7

Thus $(247)_{10}$ = $(367)_8$

HEXADECIMAL NUMBER SYSTEM

Hexadecimal number system is very popular in computer uses. The base for hexadecimal number system is 16 which requires 16 distinct symbols to represent the numbers. These are numerals 0 through 9 and alphabets A through F. Since numeric digits and alphabets both are used to represent the digits in the hexadecimal number system, therefore, this is an alphanumeric number system. Table 2.3 gives hexadecimal numbers with their binary equivalents for decimal numbers 0 through 15. From the table, it is observed that there are 16 combinations of 4-bit binary numbers and sets of 4-bit binary numbers can be entered in the computer in the form of hexadecimal (hex.) digits. These numbers are required to be converted into binary representation, using hexadecimal-to-binary converter circuits before these can be processed by the digital circuits.

Table 2.3 Binary and decimal equivalents of hexadecimal numbers

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

Example 2.16. Obtain decimal equivalent of hexadecimal number $(3A.2F)_{16}$

Solution : $(3A.\ 2F)_{16} = 3 \text{ X } 16^{1} + 10 \text{ X } 16^{0} + 2 \text{ X } 16^{-1} + 15 \text{ X } 16^{-2}$

$= (58.1836)_{10}$

Decimal-to-Hexadecimal Conversion

The conversion from decimal to hexadecimal, the procedure used in binary as well as octal systems is applicable, using 16 as the dividing (for integer part) and multiplying (for fractional part) factor.

Hexadecimal-to-Binary Conversion

Hexadecimal numbers can be converted into equivalent binary numbers by replacing each hex digit by its equivalent 4-bit binary number.

Example 2.17. Convert $(2F9A)_{16}$ to equivalent binary number.

Solution : Using Table 2.7, find the binary equivalent of each hex digit. $(2F9A)_{16}$ = $(0010\ 1111\ 1001\ 1010)_2$
= $(0010111110011010)_2$

Binary-to-Hexadecimal Conversion

Binary number can be converted into the equivalent hexadecimal numbers by making groups of four bits starting from LSB and moving towards MSB for integer part and then replacing each group of four bits by its hexadecimal representation.
For the fractional part, and above procedure is repeated starting from the bit next to the binary point and moving towards the right.

Example 2.18 Convert the binary numbers of Example 2.24 to hexadecimal numbers.

Solution :

(a) 110 0111 0001. 0001 0111 1001 = $(671.179)_{16}$

(b) 10 1101 1110.1100 1010 011 = $(2DE.\ CA6)_{16}$

(c) 1 1111 0001.1001 101 = $(1F1.99A)_{16}$

From the above examples, we observe that in forming the 4-bit groupings 0's may be required to complete the first (most significant digit) group in the integer part and the last (least significant digit) group in the fractional part.

Conversion from Hex-to-Octal and Vice-Versa

Hexadecimal numbers can be converted to equivalent octal numbers and octal numbers can be converted to equivalent hex numbers by converting the hex/octal number to equivalent binary and then to octal/hex, respectively.

Example 2.19 Convert the following hex numbers to octal numbers.

(a) A72E (b) 0.BF85

Solution : (a) $\mathbf{(A72E)_{16} = (1010\ 0111\ 0010\ 1\underline{110})_2}$

= $(1010\ 0111\ 0010\ 1\underline{110})_2$

= $(1123456)_8$

Example 2.20 Convert $(247.36)_8$ to equivalent hex number.

Solution :

$(247.36)_8$ = $(010\ 100111.011\ 110)_2$

= GROUP OF FOUR BITS

= $(A\ 7.78)_{16}$

Hexadecimal Arithmetic

The rules for arithmetic operations with hexadecimal numbers are similar to the rules for decimal, octal and binary systems. The information can be handled only in binary form in a digital circuit and it is easier to enter the information using hexadecimal number system. Since arithmetic operations are performed by the digital circuits on binary numbers, therefore hexadecimal numbers are to be first converted into binary numbers. Arithmetic operations will become clear from the following examples.

Example 2.21 Add $(7F)_{16}$ and $(BA)_{16}$

Solution :

$$\begin{array}{r} 7F = 01111111 \\ (+)BA = 10111010 \\ \hline (139)16 = 100111001 \end{array}$$

UNIT-3

Codes and Parity

Computers and other digital circuits process data in the binary format. Various binary codes are used to represent data which may be numeric, alphabets or special characters. Although, in every code used the information is represented in binary form, the interpretation of this binary information is possible only if the code in which this information is available is known. For example, the binary number 1000001 represents 65 (decimal) in straight binary, 41 (decimal) in BCD and alphabet A in ASCII code. A user must be very careful about the code being used while interpreting information available in the binary format. Codes are also used for error detection and error correction in digital systems.

BINARY CODED DECIMAL (BCD)

Computers work with binary numbers. We work with decimal numbers. A code is needed to represent decimal numbers and binary numbers.

A weighted binary code is one in which each number carries a certain weight. A string of 4 bits is known as nibble. Binary coded decimal (BCD) means that each decimal digit is represented by a nibble (binary code of 4 digits). Main BCD codes have been proposed, e.g., 8421, 2421, 5211, X53. Out of these 8421 code is the most predominant BCD code. The designation 8421indicates the weights of the 4 bits (8, 4, 2 and 1 respectively starting from the left most bit). When one refers to a BCD code, it always means 8421 code. Though 16 numbers (2^4) can be represented by 4 bits, only 10 of these are used. Table 2.4 shows the BCD code. The remaining 6 combinations, i.e., 1010, 1011, 1100, 1101, 1110 and 1111 are invalid in 8421 BCD code. To express any number in BCD code, each decimal number is replaced by the appropriate four bit code of Table 2.4. BCD code is used in pocket calculators, electronic counters, digital voltmeters, digital clocks etc. Early versions of computers also used BCD code. However, the BCD code was discarded for computers because this code is slow and more complicated than binary.

Table shows some decimal numbers and their representation in octal, hexadecimal, binary and BCD systems.

Table: 8421 BCD code

Decimal	8421 BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Table: Number systems

Decimal	Octal	Hexadecimal	Binary	8421 BCD
0	0	0	0000 0000	0000 0000 0000
1	1	1	0000 0001	0000 0000 0001
2	2	2	0000 0010	0000 0000 0010
3	3	3	0000 0011	0000 0000 0011
4	4	4	0000 0100	0000 0000 0100
5	5	5	0000 0101	0000 0000 0101
6	6	6	0000 0110	0000 0000 0110
7	7	7	0000 0111	0000 0000 0111
8	10	8	0000 1000	0000 0000 1000
9	11	9	0000 1001	0000 0000 1001
10	12	A	0000 1010	0000 0001 0000
11	13	B	0000 1011	0000 0001 0001
12	14	C	0000 1100	0000 0001 0010
13	15	D	0000 1101	0000 0001 0110
14	16	E	0000 1110	0000 0001 0100
15	17	F	0000 1111	0000 0001 0101
16	20	10	0001 0000	0000 0001 0110
32	40	20	0010 0000	0000 0011 0010
64	100	40	0100 0000	0000 0110 0000
128	200	80	1000 0000	0001 0010 1000
200	310	C8	1100 1000	0010 0000 0000
255	377	FF	1111 1111	0010 0101 0101

BCD ADDITION

Addition is the most important arithmetic operation. Subtraction, multiplication and division can be done by using addition. The rules for BCD addition are :

1. Add the two numbers using binary addition (section 2.5). If the four bit sum is equal or less than 9(i.e., equal to or less than 1001) it is a valid BCD number.
2. If the four bit sum is more than 9 or a carry is generated from the group of 4 bits, the result is invalid. In such a case add 6(i.e., 0110) to the four bit sum to skip the 6 invalid states. If a carry is generated when adding 6, add the carry to the next four bit group.

Example: Represent the following decimal numbers in BCD and add (a) 5 and 4 (b) 7 and 6 (c) 15 and 17 (d) 131 and 162 (e) and 53.

```
     Solution : (a) 5   0101         (b)       7      0111
     + 4                0100                 + 6      0110
9₁₀                   ______                  ___________
                       1001      13₁₀                1101     invalid
                                                     0110
                                                     _____
                                                     10011

(c)                  15    0001 0101
                 +   17    0001 0111
                 ____     _________
                 +   32   0010 1100     Left group is valid, right group is invalid. Add 6 to
                                        the right group and carry to the left group

(d)                 131              0001 0011 0001
                   +162              0001 0110 0010
                   293₁₀             0010 1001 0011
```

GREY CODE

It is an unweighted code. The bit positions do not have any specific weights assigned to them. However, the most important characteristic of this code is that only a signal bit change occurs when going from one code number to next. (In binary systems all the 4 bits change when we go from 0111 to 1000. i.e., 7_{10} to 8_{10}). The single bit change property is important in some applications, e.g., shaft position encoders. In these applications the chances of error increase if more than one bit change occurs. Table shows the 4 bit gray code.
It is seen in Table 2.6 that in gray code change is by 1 bit only at one times. Like binary Gray code can have any number of bits.

Table Gray Code

Decimal	Binary	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101

Binary to Gray Conversion

The rules for changing binary number into equivalent Gray code are :

1. The left most bit (most significant bit) in Gray code is the same as the left most bit in binary

   1 0 1 1 Binary
   ↓
   1 Gray

2. Add the left most bit to the adjacent bit

   1 + 0 1 1
   1 1

3. Add the next adjacent pair

   1 0 + 1 1
   1 1 1 0

4. Add the next adjacent pair and discard carry

   1 0 1 + 1
   1 1 1 0

5. Continue the above process till completion.

Gray to Binary Conversion

the method to convert from Gray code to binary is an under:

1. Left most bit in binary is the same as the left most bit in Gray

   code. 1 1 0 1 1 Gray

   ↓

   1 Binary

2. Add the binary MSB to the Gray digit in the adjacent position. Discard

   carry 1 1 0 1 1 Gray

   ↓↗

   1 0 Binary

3. Add the binary digit generated in step 2 to the next Gray digit. Discard

   carry 1 1 0 1 1 Gray

   ↓ ↗

   1 0 0 Binary

4. Continue the above process till all the digits are covered. Discard carry in each case

   1 1 0 1 1 Gray

   ↓ ↗

   1 0 0 1 0 Binary

EXCESS 3 CODE

Excess 3 is a digital code obtained by adding 3 to each decimal digit and then converting the result to four bit binary. It is an unweighted code, i.e., no weights can be assigned to any of the four digit positions.

Table Excess 3 code

Decimal	Excess 3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

Out of the possible 16 code combinations ($2^4 = 16$), only 10 are used in excess 3 code. The remaining 6, i.e., 0000, 0001, 0010, 1101, 1110 and 1111 are invalid in this code.

Example: Convert the following decimal numbers to excess 3 code (a) 14 (b) 32 (c) 46 (d) 430.

Solution : In each case add 3 to each digit in decimal number and then convert into binary. (a)

$$\begin{array}{cc} 1 & 4 \\ +3 & +3 \\ \hline 4 & 7 \\ \downarrow & \downarrow \\ 0100 & 0111 \end{array}$$

(b)

$$\begin{array}{cc} 3 & 2 \\ +3 & +3 \\ \hline 6 & 5 \\ \downarrow & \downarrow \\ 0110 & 1010 \end{array}$$

(c)

$$\begin{array}{cc} 4 & 6 \\ +3 & +\ 3 \\ 7 & 9 \\ \downarrow & \downarrow \\ 0111 & 1001 \end{array}$$

(d)

$$\begin{array}{ccc} 4 & 3 & 0 \\ +3 & +\ 3 & +\ 3 \\ 7 & 6 & 3 \\ \downarrow & \downarrow & \downarrow \\ 0111 & 0110 & 0011 \end{array}$$

ERROR DETECTION CODES

Every digit of a digital system must be correct. An error in any digit can cause a problem because the computer may recognize it as something else. The correct ASCII code for A is 1000001. An error in one bit (i.e., 1000011) would mean C. Many methods have been devised to detect such errors.

Parity

Parity refers to the number of 1s in the binary word. When the number of 1s in the binary word is odd, it is said to have odd parity. When the number of words is even, it is said to have every parity, e.g.,

1100110	even parity
1000011	odd parity

One method for error detection is to use 7 bits for data and 8th (most significant) bit for parity. The parity bit can be 1 or 0. To make odd parity, the parity bit is set to 1 or 0. If the word has odd number of 1s, the parity bit is set to 0. If the word has even number of 1s, the parity bit to set to 1 so as to make the total number of 1 odd. e.g.,

Table

Parity	Data	Total number of 1s
0	1100111	5
0	1101011	5
1	1000010	3
1	0000011	3

At the receiving point the parity is checked to see that it is odd. if it is even, an error has been committed and the data is required to be transmitted again.

In some computer systems even parity is also used, i.e., parity bit is set so as to make the total number of 1s even.

Check Sums

The above discussed parity check cannot detect two errors in the same word. If 01000011 or 01010111 is transmitted instead of 01100111, the errors will not be detected. One method to detect such cases is the check sums. As each word is transmitted, it is added to the previous word and the sum is retained at the sending end. E.g.,

Word A	0 0 0 1 0 0 1 1
Word B	1 0 0 1 0 1 0 0
SUM	1 0 1 0 0 1 1 1

Each successive word is added to the sum of the previous words. At the end of transmission, the sum (known as check sum) is also sent and is checked at the receiving point. Check sum method is commonly used in tele-processing.

Parity Data Codes

Parity can be added within each character. Two of these methods are known as 2 out of 5, and biquinary and are shown in Table 2.9.

Table Parity data codes

Decimal	2 out of 5 code	Biquinary 5043210
0	00011	0100001
1	00101	0100010
2	00110	0100100
3	01001	0101000
4	01010	0110000
5	01100	1000001
6	10001	1000010
7	10010	1000100

8	10100	1001000
9	11000	1010000

The 2 out of code uses five bits to represent the 10 decimal digits. Each code word has two 1s. This facilitates decoding and easier error detection. If the number of 1s received is other than two, an error is indicated. It is used in communication systems.
The biquinary code has a 2 bit group and a 5 bit group. Each of these groups has a single 1. Its weight are 5043210. It is used in counters. The two bit group having weights 50 indicates whether the number is less than or equal to or greater than 5. The five bit group indicates the count below or above 5.

Error Correction Code

A method developed by RW Hammings and known as Hamming code is very commonly used for error correction. It contains parity bits located in proper positions.
To find the required number of parity bits, the following equation is used

$$2^p \geq m + p + 1 \qquad \text{...(2.3)}$$

where m = number of information bits
p = number of parity bits
If m = 4, p must have a minimum value of 3 for Eqn. (2.3) to be satisfied. If m = 11, p must have a minimum value of 4 to satisfy Eqn. (2.3). the parity bits are located at each 2^n bit, e.g., for a bit data, the parity bits are located at positions 2^0, 2^1, 2^2, i.e., 1, 2, 4th bit position starting with least significant bit (right most bit). Thus the format is
D7 D6 D5 P4 D3 P2 P1
where P_1, P_2, P_4 indicate parity bits and the remaining are data (information bits).
For 11 bit data, the format is :
D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 P4 D3 P2 P1
The assignment of parity bits in the four bit data is as under:
The bit P_1 is set so that it establishes even parity over bits 1, 3, 5 and 7 (i.e., data bits D_3, D_5, D_7 and itself P_1). P_2 is set for even parity over bits, 2, 3, 6 and 7 (i.e., D_3, D_6, D_7 and itself P_2). P_4 is set for even parity over bits 4, 5, 6, 7 (i.e., D_5, D_6, D_7 and itself P_4).
At the receiver end, each group is checked for even parity. If an error is indicated, it is located by forming a p bit binary number formed by the p parity bits. When the number of parity bits is 3 then binary number is 3 bit. The method is as discussed in examples 2.44 and 2.45.

Example: Data required to be transmitted is 1101. Formulate even parity Hamming code.
Solution : Since m = 4, p must be 3 to satisfy Eqn. (2.3). Parity bit positions are 2^0, 2^1, 2^2, i.e., bits 1, 2, and 4.

D7 D6 D5 P4 D3 P2 P1 1 1 0 0 1 1 0

P must be 0 so that there is even parity over bits 1, 3, 5, 7. P_2 must be 1 to create even parity over bits 2, 3, 6, 7 and P_4 must be 0 so that there is even parity over bits 4, 5, 6, 7. The values of P_1, P_2 and P_4 are indicated above.

Example : A seven bit Hamming code as received is 1111101. Check if it is correct. If not find the correct code if even parity is used.

Solution : D_7 D_6 D_5 P_4 D_3 P_2 P_1 1 1 1 11 0 1

Bits 4, 5, 6, 7 have even number of 1s. Hence no error

Bits 2, 3, 6, 7 have odd number of 1s. Hence error

Bits 1, 3, 5, 7 have even number of 1s. Hence no error

Evidently the error is in bit 2 position. The correct code is 1111111.

Example: The seven bit Hamming code as received is 0010001. Assuming that even parity has been used, check it is correct. If not find the correct code.

Solution : D_7 D_6 D_5 P_4 D_3 P_2 P_1 0 0 1 0 0 0 1

Bits 4, 5, 6, 7 hae odd number of 1s. Hence error

Bits 2, 3, 6, 7 have even number of 1s. Hence no error

Bits 1, 3, 5, 7 have even number of 1s. Hence no error.

Evidently the error in bit 4. The correct code is 0011001.

UNIT-4

Logic Gates and Families

The basic building block in a digital system is logic gate logic circuits have evolved into families each of which has its own advantages and disadvantages. Generally a digital system is designed with circuits from one family only. When circuits from more than one family are used. It is necessary to ensure that the output of one is compatible with input to the other.

The different logic functions are available in integrated circuit (IC) form. All digital systems are built with digital ICs. The present day ICs have many advantages in terms of size, power, reliability *and const over discrete circuits*. A monolithic integrated circuit is an electronic circuit constructed entirely on a single chip of silicon. All the components, i.e., transistors, diodes, resistors etc. are an integral part of this chip. Typical chip sizes range from 40 × 40 mils** to about 300 × 300 mils and it contains both active and passive components. Many processes like wafer preparation, impurity diffusion, ion implantation, oxide growth, photolithography are used in their manufacture.

CLASSIFICATION OF LOGIC FAMILIES

Classification as per Level of Integration

As per this classification digital integrated circuits can be classified as small scale integration (SSI), Medium scale integration (MSI), Large scale integration (LSI), Very large scale integration (VLSI) and ultra large scale integration (ULSI). This classification is given in Table

Table Levels of integration of digital ICs

Category	Numbers of equivalent basic gates on a single chip	Total number of components on single chip
SSI	Less than 12	Less than 100
MSI	12-99	100-999
LSI	100-999	1000-9999
VLSI	1000-9999	10,000 to 99,999
ULSI	10,000 and more	100,000 and more

SSI are the least complex and include basic gates and flip flops. They are available in dual in package (DIP) or flat package and 14 or 16 pin versions.

Small digital sub systems form the MSI category. The complex logic functions, e.g., adders, registers, comparators, code converters, counter, multiplexers etc., are fabricated in MSI. They are also available in dual in package (KIP), flat package and carrier package with 24 or 28 pins.

LSI chips are small digital systems, e.g., digital clocks, calculators, microprocessors, ROM, RAM*** etc.

VLSI is a digital system on a chip. Large memory chips and advanced microprocessors fall in this category.

ULSI is the most complex of digital ICs.

In 1960s, digital ICs manufactured were SSI and MSI only. LSI technology was developed in late 1960s and single chip calculators and memories become available. After this microprocessors were introduced in 1973. A microprocessor has the architecture of a computer on a single chip. In 1980s many sections of computers (central processing unit or CPU, RAM, ROM etc.,) were combined into a single chip. At present the latest version of microprocessor has equivalent of millions of transistors on a single chip.

Classification as per Technology

Digital ICS are manufactured by two technologies viz., Bipolar and MOS.

The bipolar family uses transistor fabricated on a chip. This family includes DTL (Diode transistor logic using diodes and transistor), TTL (transistor-transistor logic which uses transistors only) and ECL (emitter coupled logic). TTL is the most poplar family in SSI and MSI category.

MOS family includes PMOS (p-channel MOSFET), NMOS (n0channel MOSFET) and CMOS (complementary MOSFET). PMOS is almost obsolete. NMOS is dominating the LSI field. CMOS is the most commonly used technology for digital wrist watches, pocket calculators etc.

The technologies being used now-a-days are TTL, ECL and CMOS. Before discussing these technologies we discuss the important specifications of digital ICs.

CHARACTERISTICS OF DIGITAL ICs

With the widespread use of ICs in digital systems and with the development of various technologies for the fabrication of ICs, it has become necessary to be familiar with the characteristics of IC logic families and their relative advantages and disadvantages. Digital ICs are classified either according to the complexity of the circuit, as the relative number of individual basic gates (2-input NAND gates) it would require to build the circuit to accomplish the same logic function or the number of components fabricated on the chip.

The various characteristics of digital ICs used to compare their performances are:

1. Speed of operation,
2. Power dissipation,
3. Figure of merit,
4. Fan-out,
5. Current and voltage parameters,
6. Noise immunity,
7. Operating temperature range,
8. Power supply requirements, and
9. Flexibilities available.

Basic difference between positive and negative logic

There are two types of representations used in digital systems, the positive logic and the negative logic representations.

In positive logic representation Bit 1 represents Logic high and Bit 0 represent a Logic low as shown in fig 2 a and b. High is represented by +5 Volts and low is represented by -5 Volts or 0 Volts.

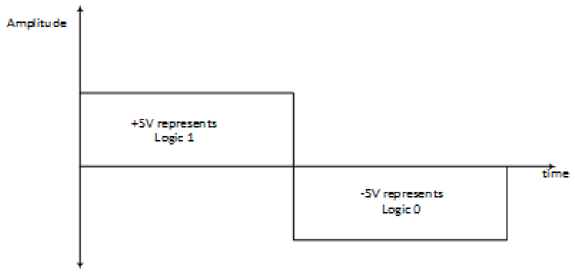


Fig. 2 a. Positive logic representation

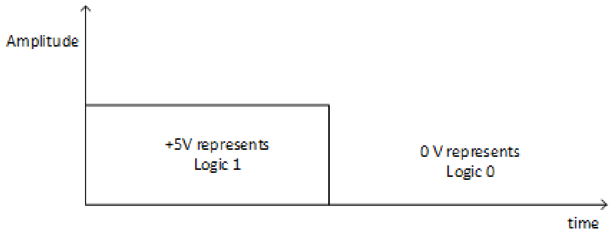


Fig. 2 b. Positive logic representation

In Negative logic representation Bit 1 represents logic low and Bit 0 represents logic high as shown in Fig 3 a and b. In terms of voltage level, bit 1 can be represented as +5V and bit 0 can be represented as 0 V or -5 Volts

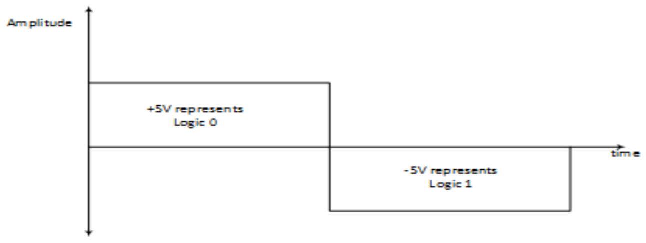


Fig. 3 a. Negative logic representation

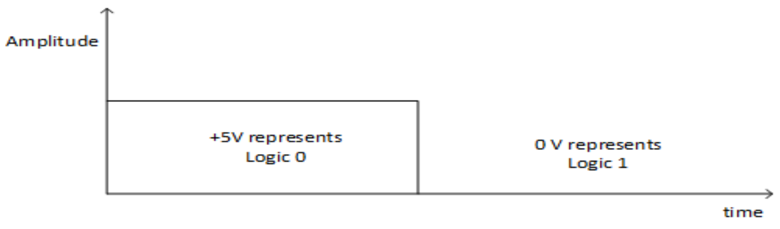


Fig. 3 b. Negative logic representation

Speed of Operation

The speed of a digital circuit is specified in terms of the propagation delay time. The input and output waveforms of a logic gate are shown in Fig. 7.1. The delay times are measured between the 50 per cent voltage levels of input and output waveforms. There are two delay times: t_{pLH} , when the output goes from the HIGH state to the LOW state and t_{pLH} , corresponding to the output making a transition from the LOW state to the HIGH state. The propagation delay time of the logic gate is taken as the average of these two delay times.

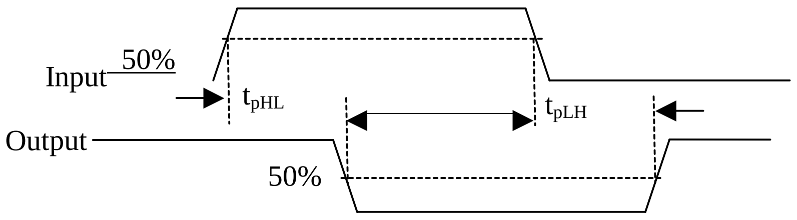


Fig. Input and output voltage waveforms to define propagation delay tines.

Power Dissipation

This is the amount of power dissipated in an IC. It is determined by the current, $1_{CC,}$ that it draws from the V_{CC} supply, and is given by $V_{CC} \times 1_{CC}$. 1_{CC} is the average value of 1_{CC} (0) and 1_{CC}(1). This power is specified in milliwatts.

Figure of Merit

The figure of merit of a digital IC is defined as the product of speed and power. The speed is specified in terms of propagation delay time expressed in nanoseconds.

Figure of merit = propagation delay time (ns) × power (mW) It is specified in pico joules (ns × mW = p^J)

A low value of speed-power product is desirable. In a digital circuit, if it is desired to have high speed, i.e. low propagation delay, then there is a corresponding increase in the power dissipation and vice-versa.

Fan-Out

This is the number of similar gates which can be driven by a gate, High fan-out is advantageous because it reduces the need for additional drivers to drive more gates.

Current and Voltage Parameters

The following currents and voltages are specified which are very useful in the design of digital system.

High-level input voltage, V_{IH}: This is the minimum input voltage which is recognized by the gate as logic 1.

Low-level input voltage, V_{IL}: This is the maximum input voltage which is recognized by the gate as logic 0.

High-level output voltage, V_{OH} : This is the maximum input voltage available at the output corresponding to logic 1.

Low-level output voltage, V_{OL}: This is the maximum input voltage available at the output corresponding to logic 0.

High-level input current, I_{IH}: This is the maximum current which must be supplied by a driving source corresponding to 1 level voltage.

Low-level input current, I_{IL}: This is the maximum current which must be supplied by a driving source corresponding to 0 level voltage.

High-level output current, I_{OH}: This is the maximum current which the gate can sink in 1 level.

Low-level output current, I_{OL}: This is the maximum current which the gate can sink in 0 level.

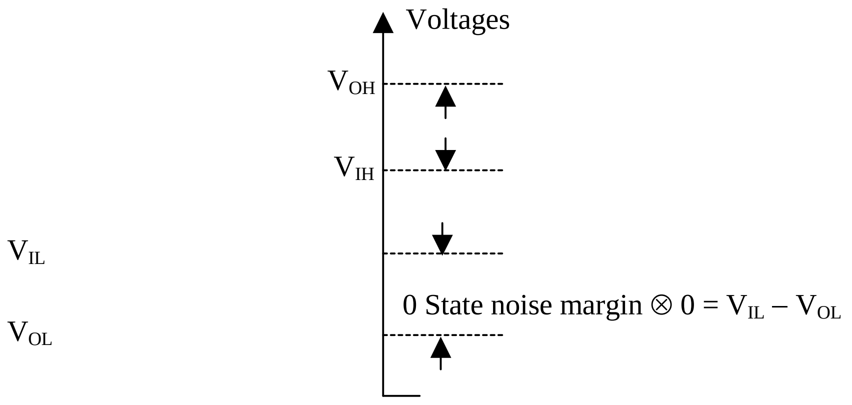
High-level supply current, I_{CC} (1): This is the supply current when the output of the gate is at logic 1.

Low-level supply current, I_{CC} (0) : This is the supply current when the output of the gate is at logic (0).

Noise Immunity

The input and output voltage levels defined above are shown in Fig. 4.3. Stray electric and magnetic fields may induce unwanted voltages, known as noise, on the connecting wires between logic circuits. This may cause the voltage at the input to a logic circuit to drop below V_{IH} or rise above V_{IL} and may produce undesired operation.

The circuit's ability to tolerate noise signals is referred to as the no8ise immunity, a quantitative measure of which is called noise margin. Noise margins are illustrated in Fig.. The noise margins defined above are referred to as dc noise margins. Strictly speaking, the noise is generally thought of as an a.c. signal with amplitude and pulse width. For high speed ICs, a pulse width of a few microseconds is extremely long in comparison to the propagation delay time of the circuit and therefore, may be treated as d.c. as far as the



response of the logic circuit is concerned. As the noise pulse width decreases and approaches the propagation delay time of the circuit, the pulse duration is too short for the circuit to respond. Under this condition, a large pulse amplitude would be required to produce a change in the circuit output. This means that a logic circuit can effectively tolerate a large noise amplitude if the noise is of a very short duration. This is referred to as ac noise margin and is substantially greater than the dc noise margin. It is generally supplied by the manufacturers in
the form of a curve between noise margin and noise pulse width.

Operating Temperature

The temperature range in which an IC functions properly must be known. The accepted temperature ranges are: 0 to + 70 ^{0}C for consumer and industrial applications and --55 ^{0}C to + 125 ^{o}C for military purposes.

Power Supply Requirements

The supply voltage (s) and the amount of power required by an IC are important characteristics required to choose the proper power supply.

TRANSISTOR-TRANSISTOR LOGIC (TTL)

Fig. shows a TTL NAND gate with a totem pole output. The totem pole output means that transistor T_4 sits atop T_3 so as to give low output impedance. The low output impedance implies a short time constant RC so that the output can change quickly from one state to 60 emitters have been developed. In Fig. 7.4, T_1 has 3 emitters so that there can be three inputs A, B, C. The transistor T_2 acts as a phase splitter because the emitter voltage is out of phase with the collector voltage. The transistors T_3 and T_4 form the totem pole output. The capacitance C_L represents the stray capacitance etc. The diode D is added to ensure that T_4 is cut off when output is low. The voltage drop of diode D keeps the base- emitter junction of T_4 reverse biased so that only T_3 conducts when output is low. The operation can be summed up as under:

Condition I : At least one input is low (i.e.,0). Transistor T_1 saturates. Therefore, the base voltage of T_2 is almost zero. T_2 is cut off and forces T_3 to cut off. T_4 acts like an emitter follower and couples a high voltage to load. Output is high (i.e., Y = 1).

Condition II : All inputs are high. The emitter base junctions of T_1 are reverse biased. The collector base junction of T_1 is forward biased. Thus, T_1 is in reverse active mode. The collector current of T_1 flows in reverse direction. Since this current is flowing into the base of T_2, the ansistorT_2 and T_3 saturate and output Y is low.

Condition II : The circuit

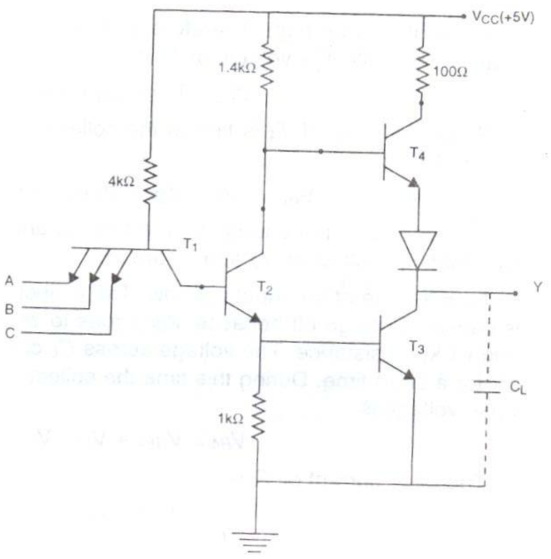


Fig. TTL NAND gate with totem pole output

is operating under condition II when one of the input becomes low. The corresponding emitter base junction of T_1 starts conducting and its base voltage drops to a low value. Therefore, T_1 is in forward active mode. The high collector current of T_1 removes the stored charge inT_2 and T_3 and therefore, T_2 and T_3 go cutoff and T_1 saturates and output Y returns to high.

TTL is the most popular logic in the SSI and MSI category. Table 7.2 summarises the input and output conditions.

Table Three input NAND gate

A	B	C	$Y = \overline{ABC}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Two important characteristics of a digital IC are power dissipation per gate and propagation delay time. TTL 5400/7400 series is the most popular and commonly used. The power dissipation of TTL series varies from 1 to 20mW per gate. The propagation delay time is the time taken by the output to change from one state to the other. Thus, this determines the speed of operation. For the different ICs in TTL series, the propagation delay time varies from about 1.5 ns to 10 ns.

An improved version of TTL logic uses Schottky transistors and is designated as 74 S series. In this series, bipolar junction transistors have been replaced by Schottky transistors. In this series, the propagation delay time is reduced by a factor of 3 but the power dissipation is double than that in 5400/7400 series.

Open Collector Connection

TTL gates without active pull up and with collector open and brought out can be connected for wired AND connection. Fig. shows one such connection. In this case all the open collector terminals share one common pull up resistance R. The size of this resistance R depends on the number of open collector gates, required noise margin, fan out etc.

In any of the input (say A) of TTL gate is not used (and left unconnected or open) the corresponding emitter base junction of T_1 will not be forward biased and behave as if logical 1 is applied to this input. Therefore, the unused input terminal of any TTL gate should preferably be

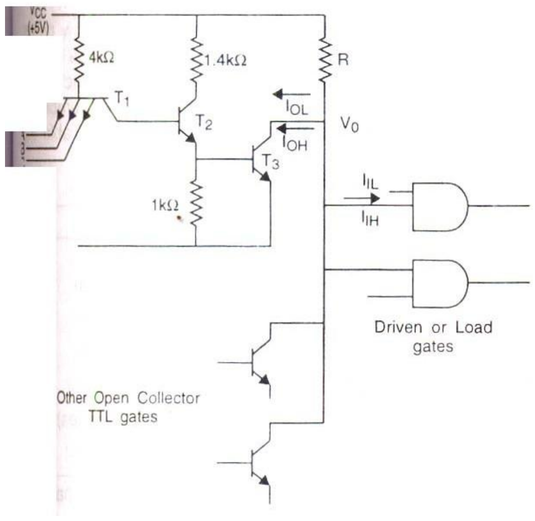


Fig. Open collector TTL gate

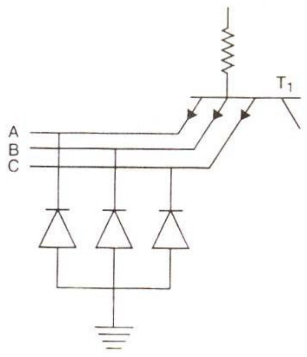


Fig. Clamping diodes connected at inputs to TTL gate

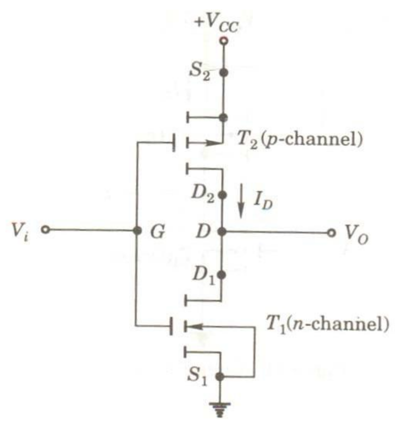
TTL gates are used with clamping diodes connected between each input terminal and round as shown in Fig.. These diodes clamp the input to – 0.7 V and minimize undesired negative noise transients.

CMOS LOGIC

A complementary MOSFET (CMOS) is obtained by connecting a p-channel and an n-channel MOSFET is series, with drains tied together and the output is taken at the common drain. Input is applied at the common gate formed by connecting the two gates together. In a CMOS, p-channel and n-channel enhancement MOS devices are fabricated on the same chip, which makes its fabrication more complicated and reduces the packing density. But because of negligibly small power consumption, CMOS is ideally suited for battery operated systems.

Its speed is limited by substrate capacitances. To reduce the effect of these substrate capacitances, the latest technology known as silicon on sapphire (SOS) is used in microprocessor fabrication which employs an insulating substrate (sapphire). CMOS is becoming very popular in MSI and LSI areas.

Fig. CMOS Inverter



CMOS Inverter

The basic CMOS logic circuit is an inverter shown in Fig.. For this circuit the logic levels are 0 V (logic 0) and V_{CC} (logic 1). When $V_i = V_{CC}$, T_1 turns On and T_2 turns OFF. Therefore $V_0 = 0$ V, and since the transistors are connected in series the current I_D is very small. On the other hand,
Then $V_i = 0$ V, T_1 turns OFF and T_2 turns ON giving an output voltage $V_0 = V_{CC}$ and I_D is again very small. In either logic state, T_1 or T_2 is OFF and the quiescent power dissipation which is the product of the OFF leakage current and V_{CC} is very low. More complex functions can be realized by combinations of inverters.

CMOS NAND and NOR Gates

A 2-input CMOS NAND gate is shown in Fig. and NOR gate in Fig.. In the NAND gate, the NMOS drivers are connected in series, where as the PMOS loads are connected in parallel. On the other hand, the CMOS NOR gate is obtained by connecting the NMOS drivers in parallel and PMOS loads in series. The operation of NAND gate can be understood from Table. The operation of the NOR gate can be verified in the similar manner.

Table Operation of CMOS NAND gate

Inputs		State of MOS devices				Output
A	B	T_1	T_2	T_3	T_4	Y
0	0	OFF	OFF	ON	ON	V_{CC}
0	V_{CC}	ON	OFF	ON	OFF	V_{CC}
V_{CC}	0	OFF	ON	OFF	OF	V_{CC}
V_{CC}	V_{CC}	ON	ON	OFF	OFF	0

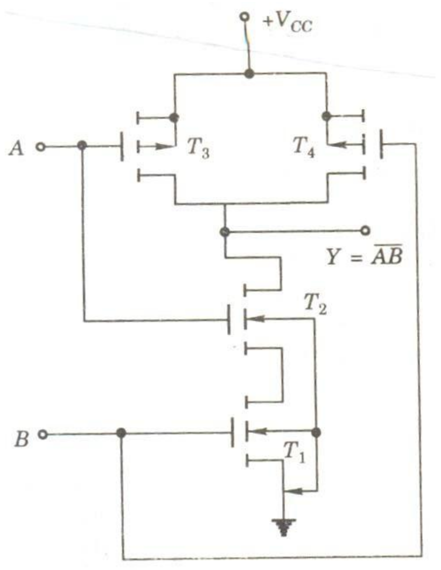


Fig. A 2-input CMOS NAND gate.

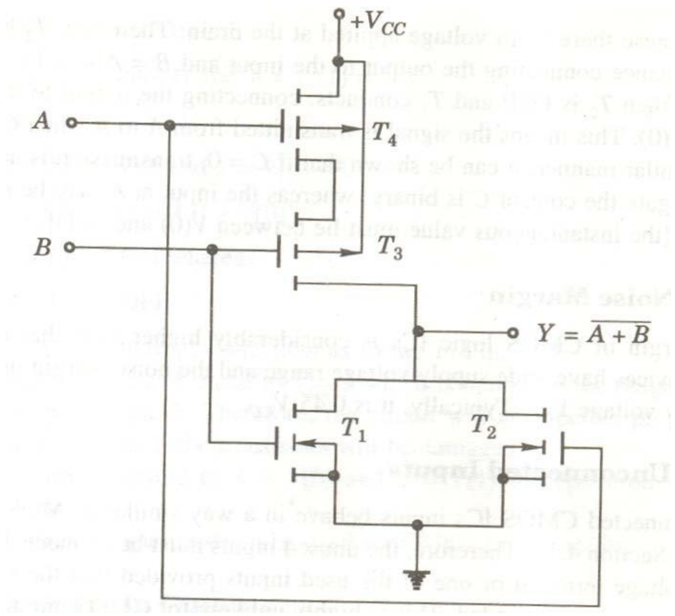
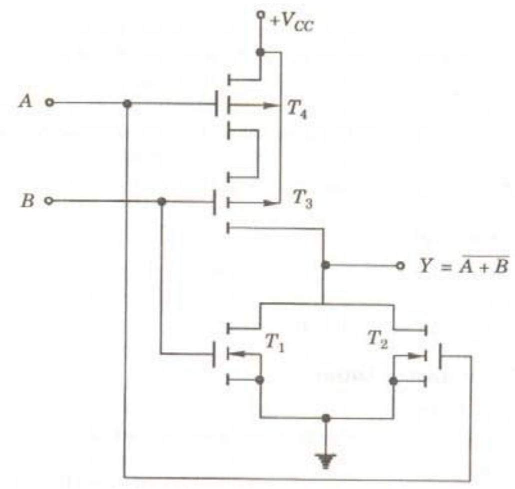


Fig. 2-input CMOS NOR gate.

CMOS Transmission Gate

A CMOS transmission gate controlled by gate voltages C and $\overline{C}$ is shown in Fig. Assume C – 1

.



If A = V(1), then T_1 is OFF and T_2 conducts in the ohmic

Fig. (a) A CMOS transmission gate.

region because there is no voltage applied at the drain. Therefore, T_2 behaves as a small resistance connecting the output to the input and B = A = V(1). Similarly, if A = V(0), then T_2 is

OFF and T_1 conducts, connecting the output to the input and B = A = V(0). This means the signal is transmitted from A to B when C = 1.

In a similar manner, it can be shown that if C = 0, transmission is not possible.

In this gate the control C is binary, whereas the input at A may be either digital or analog [the instantaneous value must lie between V(0) and V(1)].

Noise Margin

Noise margin of CMOS logic ICs is considerably higher than that of TTL ICs. CMOS devices have wide supply voltage range and the noise margin increases with the supply voltage V_{CC} Typically, it is 0.45 $V_{CC.}$

Unconnected Inputs

The unconnected CMOS ICs inputs behave in a way similar to MOS devices. Therefore, the unused inputs must be connected to either the supply voltage terminal or one of the used inputs provided that the fan-out of the signal source is not exceeded. This is highly unlikely for CMOS circuits because of their high fan-out.

Some CMOS ICs have Zener diodes connected at the inputs for protection against high input voltages. Wired-Logic

Figure shows two CMOS inverters with their outputs connected together. In this circuit,

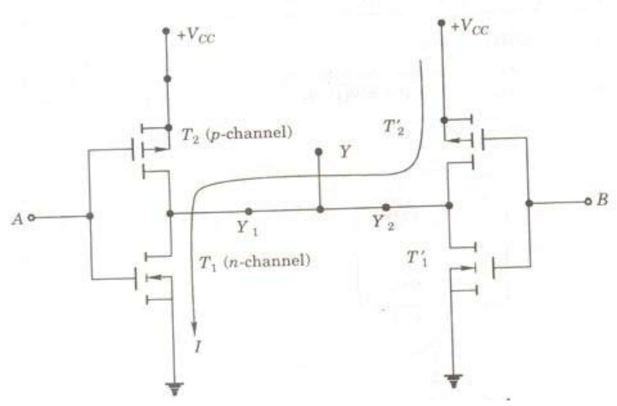


Fig. CMOS inverters with outputs connected.

(i) When A = B = V(0)

T_1 and T'_1 are cut-off and Y = V (1) = V_{CC}

(ii) When A = B = V(1)

T_1 and T'_1 are ON and Y = V(0) = 0

(iii) When A =V(1) and B = V(0)

T_1 and T'_2 are ON whereas

T'_1 and T_2 are OFF

Therefore, a large current I will flow as shown in Fig. 7.11.

This will make voltage at Y equal to $V_{CC}/2$ which is neither in the range of logic 0 nor in the range of logic 1. Therefore, the circuit will not operate properly. Also because of large current I, the transistors will be damaged.

Similarly, corresponding to A = V(0) and B = V(1) the operation will not be proper.

Therefore, wired-logic must not be used for CMOS logic circuits.

Open-Drain Outputs

CMOS gates with open-drain output are available which useful for wired-AND operation. In this the drain terminal of the output transistor (n-channel) is available outside and the load resistor is to be connected externally since p-channel load does not exist.

54C00/74C00 CMOS Series

There are two commonly used CMOS series ICs. These are the 4000 series and 54C/74C series. 54C/74C CMOS series is pin-for-pin, function-for-function equivalent to the 54/74 TTL family and has, therefore, become very popular. The temperature range for 54 series is -55 ^{o}C to + 125 ^{o}C and for 74C series is – 40 ^{o}C to 85 ^{o}C. It has a wide supply voltage range, 3 V to 15 V. A person can take full advantage of his knowledge of the 54?74 TTL series for the effective use of 54C/74C series.

There have been significant improvements in 54C/74C series. The 74HC/74HTC have higher speed and better current capabilities. 74HC is known as *high-speed* CMOS and 74HCTisknown as *high-speed*, TTL *compatible* CMOS series. 74AC/74ACT are very fast and have very high current sinking capabilities. There are known as *advanced* CMOS and *advanced*, *TTL compatible* CMOS, respectively. The 74 HC/74HCT/74AC/74ACT series can be operated at supply voltages in the range of 2—6 volts.

The voltage and current parameters of various 74 CMOS series with 5V supply voltage are given in Table 4.9. From the table, we observe that the output currents and voltages for 74HC/74HCT/75AC/74ACT are different when gates of these series are driving CMOS circuits and TTL circuits. 74 HCT and 74 ACT series are compatible with TTL series for input as well as output and therefore, can easily be used along with TTL ICs for optimum system design from the point of view of speed, power dissipation noise margins, cost, etc.

The fan-out of 74 HC/74HCT series is 20, whereas for 74AC/74ACT series it is 50 while driving these CMOS series. The fan-out of these gates while driving various TTL series gates can be determined using the specifications of TTL (Table 4.3) and CMOS (Table 7.4).

Table Specifications of CMOS IC families

Parameter	Load	74C	74HC	74HCT	74AC	74ACT	Units
V_{IH}		3.5	3.85	2.0	3.85	2.0	volts
V_{IL}		1.5	1.35	0.8	135	0.8	volts
V_{OH}	CMOS	4.5	4.4	4.4	4.4	4.4	volts
	TTL		3.84	3.84	3.76	3.76	volts
V_{OL}	CMO S	0.5	0.1	0.1	0.1	0.1	volt
	TTL		0.33	0.33	0.37	0.37	volt
			1	1	1	1	μA
I_{IH}		1	–1	–1	–1	–1	μA
I_{IL}		–1	–0.02	–0.02	–0.05	–0.05	mA
I_{OH}	CMOS	–0.1	–4.0	–4.0	–24.0	–24.0	mA
TTL	CMOS	0.36	4.0	4.0	24.0	24.0	mA
I_{OH}	CMOS	–0.1	–4.0	–4.0	–24.0	–24.0	mA TTL

TRI-STATE LOGIC

In normal logic circuits there are two states of the output, LOW and HIGH. If the output is not in the LOW state, it is definitely in the other state (HIGH). Similarly, if the output isnot in the HIGH state, it is definitely in the LOW state. In complex digital systems like microcomputers and microprocessors, a number of gate outputs may be required to be connected to a common line which is referred to as a bus which, in turn, may be required to drive a number of gate inputs. When a number of gate outputs are connected to the bus, we encounter some difficulties. These are:

1. Totem-pole outputs cannot be connected together because of very large current drain from the supply and consequent heating of the ICs which may get damaged.
2. Open-collector outputs can connected together with a common collector-resistor connected externally. This causes the problems of loading and speed of operation.

To overcome these difficulties, special circuits have been developed in which there is one more state of the output, referred to as the *third state* or *high-impedance state*, in addition to the LOW and HIGH states. These circuits are known as TRI-STATE, *tri-state logic* (TSL) or *three-state logic*. TRI-STATE, is a registered Trade Mark of National Semiconductor Corporation of USA.

There is a basic functional difference between wired-OR and the TSL. For the wired-OR connection of two functions Y_1 and Y_2 is

$Y = Y_1 + Y_2$ whereas for TSL, the result is not a Boolean function but an ability to multiplex many functions economically.

BASIC DIGITAL CIRCUITS: Logic Gates

In a digital system there are only a few basic operations performed, irrespective of the complexities of the system. These operations may be required to be performed a number of times in a large digital system like digital computer or a digital control system, etc. The basic operations are AND, OR, NOT, and FLIP-FLOP. The AND, OR, and NOT operations are discussed here and the FLIP-FLOP, which is a basic memory element used to store binary information (one bit is stored in one FLIP-FLOP).

The And Operation

A circuit which performs an AND operation is shown in Fig. 1.2. It has N inputs ($N \geq 2$) and one output. Digital signals are applied at the input terminals marked A, B, …, N, the other terminal being ground, which is not shown in the diagram. The output is obtained at the output terminal marked Y (the other terminal being ground) and it is also a digital signal. The AND operation is defined as : the output is 1 if and only if all the inputs are 1. Mathematically, it is written as

$$\begin{aligned} Y &= A \text{ AND } B \text{ AND } C \ldots \text{ AND } N \\ &= A \cdot B \cdot C \cdot \ldots \cdot N \\ &= ABC \ \ldots N \qquad \ldots(1.1) \end{aligned}$$

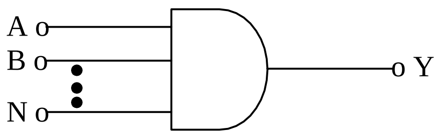


Fig. 1.2 The standard symbol for an AND gate

where A, B, C, … N are the input variables and Y is the output variable. The variables are binary, i.e. each variable can assume only one of the two possible values, 0 or 1. The *binary variables* are also referred to as *logical variables*.

Equation (1.1) is known as the *Boolean equation* or the *logical equation* of the AND *gate*. The term gate is used because of the similarity between the operation of a digital circuit and a gate. For example, for an AND operation the gate opens ($Y = 1$) only when all the inputs are present, i.e. at logic 1 level.

Truth Table Since a logical variable can assume only two possible values (0 and 1), therefore, any logical operation can also be defined in the form of a table containing all possible input combinations (2^N combinations for N inputs) and their corresponding outputs. This is known as a *truth table* and it contains one row for each one of the input combinations. For an AND gate with two inputs A, B and the output Y, the truth table is given in Table 1.1. Its logical equation is $Y = AB$ and is read as "Y equals A AND B". Since, there are only two inputs, A and B, therefore, the possible number of input combinations is four.

Truth table of a 2-input AND gate

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

The pattern in which the inputs are entered in the truth table may also be observed carefully, which is in the ascending order of binary numbers formed by the input variables.

The OR Operation

Figure shows an OR gate with N inputs (N ≥ 2) and one output. The OR operation is defined as: the output of an OR gate is 1 if and only if one or more inputs are 1. Its logical equation is given by

$$Y = A \text{ OR } B \text{ OR } C \ldots \text{ OR } N$$

$$= A + B + C + \ldots + N \qquad \ldots(1.2)$$

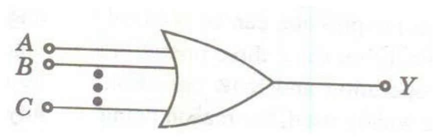


Fig. The standard symbol for an OR gate

The truth table of a 2-input OR gate is given in Table 1.2. Its logic equation is Y = A + B and is read as "Y equals A or B".

Truth table of a 2-input OR gate

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

The NOT Operation

Figure NOT gate, which is also known as an *inverter*. It has one input (A) and one output (Y). Its logic equation is written as

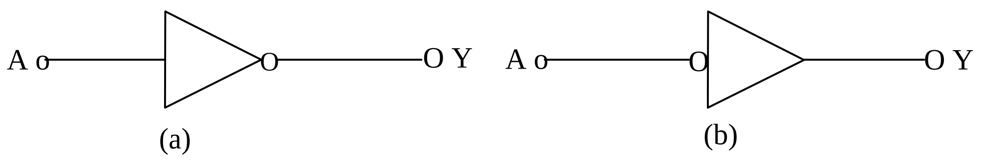


Fig. The standard symbols for a NOT gate

—

Y = NOT A

$$= A' \qquad \ldots(1.3)$$

and is read as "Y equals NOT A" or "Y equals complement of A". The truth table of a NOT gate is given in Table .

Table Truth table of a NOT gate

Input A	Output Y
0 1	1 0

The NOT operation is also referred to as an inversion or complementation. The presence of a small circle, known as the *bubble*, always denotes inversion in digital circuits.

NAND AND NOR OPERATIONS

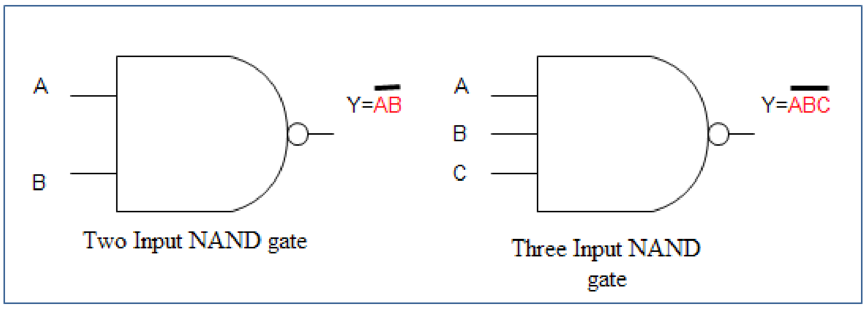
Any Boolean (or logic) expression can be realized by using the AND, OR and NOT gates discussed above. From these three operations, two more operations have been derived: the NAND operation and NOR operation. These operations have become very popular and are widely used, the reason being the only one type of gates, either NAND or NOR are sufficient for the realization of any logical expression. Because of this reason, NAND and NOR gates are known as *universal gates*.

NAND GATE

NAND gate is one of the basic logic gates to perform the digital operation on the input signals. It is the combination of AND Gate followed by NOT gate i.e. it is the opposite operation of AND gate where the Logic NAND gate is **complementary** of AND gate. The logic output of NAND gate **is low (FALSE) only when the inputs are high (TRUE).**

NAND Gate Symbol

NAND gate symbol is nothing but a round circle at the end of the AND Gate symbol to indicate the NOT gate fallowed by AND gate.



NAND Gate truth table

NAND gate truth table is shown in figure. In the truth table it is shown that when the tow inputs A & B is high then only the output Y is low and in all the remaining conditions the output is high. This property of NAND gate helps in detecting any signal or sensor failure in a group of sensors.

A	B	output
0	0	1
0	1	1
1	0	1
1	1	0

Let us suppose room temperatures sensed by the three sensors are fed to the NAND Gate to check the sensor failure. If any sensor reading is low or zero then the NAND gate output will high which alerts that one or more sensor failed in the circuit.

The NOT-AND operation is known as the NAND operation. Figure 1.5a shows and N input (N ≥ 2) AND gate followed by a NOT gate.

The operation of this circuit can be described in the following way: The output of the AND gate (Y′) can be written using Eq. (1.) Y′ = AB …N …(1.4)

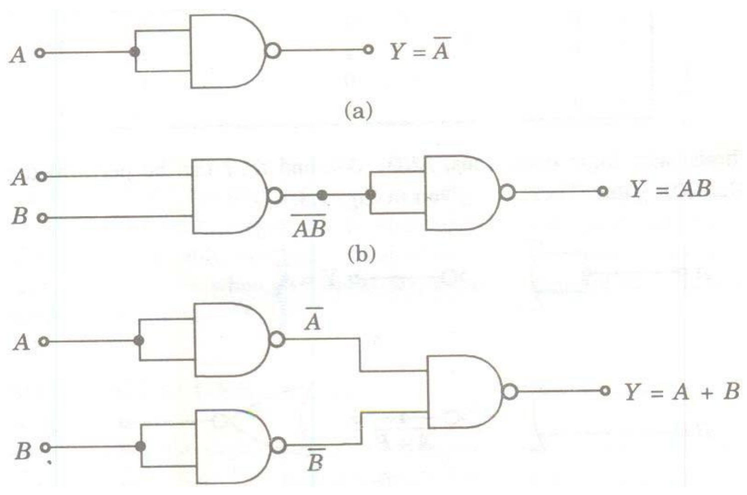
Now, the output of the NOT gate (Y) can be written using Eq. (1.3)

$$Y = \overline{Y'} = (\overline{AB...N}) \qquad \ldots(1.5)$$

The logical operation represented by Eq. (1.5) is known as the NAND operation. The standard symbol of the NAND gate is shown in Fig. 1.5b. Here, a bubble on the output side of the NAND gate represents NOT operation, inversion or complementation.

The truth table of a 2-input NAND gate is given in Table 1.4. Its logic equation is $Y = \overline{A \cdot B}$ and, is read as "Y equals NOT (A AND B)".

The three basic logic operations, AND, OR and NOT can be performed by using only NAND gates. These are given in Fig.



(a) (c)

Fig. Realization of basic logic operations using NAND gates (a) NOT (b) AND (c) OR.

The NOR Operation

The NOT-OR operation is known as the NOR operation. Figure 1.7a shows an N input (N ≥ 2) OR gate followed by a NOT gate. The operation of this circuit can be described in the following way:

The output of the OR gate Y′ can be written using Eq. (1.2) as

Y′. = A + B + … + N …(1.6)

and the output of the NOT gate (Y) can be written using Eq. (1.3) 1.7)

$\overline{Y = Y' = A + B + ... + N}$

The logic operation represented by Equ. (1.7) is known as the NOR operation.

The standard symbol of the NOR gate is shown in Fig. 1.7b. Similar to the NAND gate, a bubble on the output side of the NOR gate represents the NOT operation.

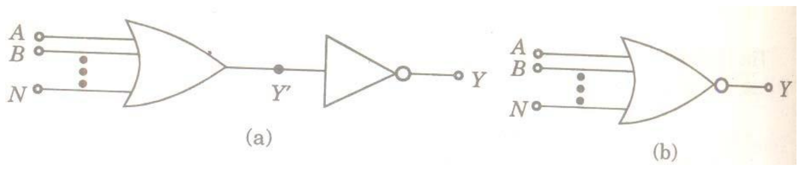


Fig. 1.7 (a) NOR operation as NOT-OR operation

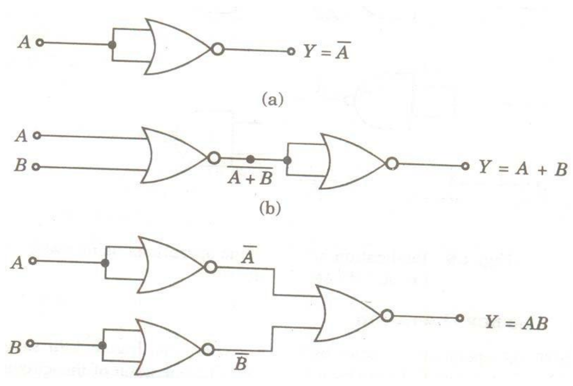
(b) Standard symbol for the NOR gate

Table gives the truth table of a 2-input NOR gate. Its logic equation is $Y = \overline{A + B}$ and is read as "Y equals NOT (A OR B)"

Table 1.5 Truth table of a 2-input NOR gate

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

The three basic logic operations, AND, OR, and NOT can be performed by using only the NOR gates.



(c)

Fig: Realization of basic logic operations using NOR gates (a) NOT (b) OR (c) AND

EXCLUSIVE–OR OPERATION

The EXCLUSIVE–OR (EX–OR) operation is widely used in digital circuits. It is not a basic operation and can be performed using the basic gates–AND, OR and NOT or universal gates NAND or NOR. Because of its importance, the standard symbol shown in Fig. 1.9 is used for this operation.

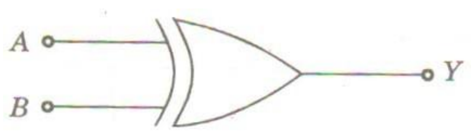


Fig. 1.9 Standard symbol for EX-OR gate.

The truth table of an EX–OR gate is given in Table 1.6 and its logic equation is written as Y = A EX – OR B = A ⊕ B …(1.8)

Table 1.6 Truth table of a 2-input EX–OR gate

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

If we compare the truth table of an EX–OR gate with that of an OR gate given in Table, we find that the first three rows are same in both. Only the fourth row is different. This circuit finds application where two digital signals are to be compared. From the truth table we observe that when both the inputs are same (0 or 1) the output is 0, whereas when the inputs are not same (one of them is 0 and the other one is 1) the output is 1.