PG CA S1-01 (DSC)

TWO YEAR MASTER'S DEGREE COURSE

SEMESTER - I

COMPUTER FUNDAMENTALS AND DIGITAL TECHNIQUES

MSc IT

Hiranya Chandra Bhuyan School of Science and Technology

KRISHNA KANTA HANDIQUI STATE OPEN UNIVERSITY

Subject Experts

Prof. Anjana Kakoti Mahanta, Gauhati University

Prof. Hiren Kumar Deva Sarma, GauhatiUniversity

Dr. Lipi B Mahanta, IASST, Guwahati

Dr. Sanjib Kr. Kalita, Gauhati University

Course Co-ordinator : **Dr. Nabankur Pathak**, KKHSOU

SLM Preparation Team

UNITS	CONTRIBUTORS
1,2	Arabinda Saikia, KKHSOU
3	Saurav Sutradhar, Royal Global University
4	Dr. Sruti Sruba Bharali, KKHSOU
5,6,7,8	Dr. Sangeeta Kakoty, KKHSOU
9,10	Dr. Jyotiprokash Goswami, Assam Engineering College
11,12,13,14,15,16	Dr. Chakradhar Das, Lakhimpur Polytechnic

Editors

Content Dr. Sanjib Kumar Kalita, Gauhati University (Former Faculty, KKHSOU)

Language Dr. Prof.(Retd.) Robin Goswami,Cotton College

Structure, Format & Graphics : Dr. Sruti Sruba Bharali, KKHSOU

July, 2023

Copyright © Krishna KantaHandiqui State Open University

This Self Learning Material (SLM) of the Krishna Kanta Handiqui State University is made available under a Creative Commons Attribution-Non Commercial-ShareAlike4.0 License (International): http.//creativecommons.org/licenses/by-nc-sa/4.0.

Printed and published by Registrar on behalf of the Krishna Kanta Handiqui State Open University.

Headquarters: Patgaon, Rani Gate, Guwahati-781017
City Office: Resham Nagar, Khanapara, Guwahati-781022 Web: www.kkhsou.in

MSc IT

COMPUTER FUNDAMENTALS AND DIGITAL TECHNIQUES

DETAILED SYLLABUS

		Page Nos.
Unit 1:	**Introduction to Computer** Block Diagram of Computer, Evolution, Generations, Classification and its Application	**7-20**
Unit 2:	**Basic Components of Computer** Concept of Bit and Byte; CPU, ALU, CU; Computer Memory: Primary and Secondary; Input/Output Devices; Buses: Address, Dataand Control.	**21-49**
Unit 3:	**Introduction to Computer Security** Computer Virus, Worm, Spyware, Malware, Trojanhorse, Antivirus Software	**50-65**
Unit 4:	**Cyber Threats and Legal Framework** Cyber Crimes against Individuals, Institution and State, Hacking, Digital Forgery, Cyber Stalking/Harassment, Cyber Pornography, Identity Theft & Fraud, Cyber terrorism, Cyber Defamation, Different offences under IT Act, 2000	**66-80**
Unit 5:	**Introduction to Number Systems** Decimal, Binary, Hexadecimal and Octal number system, Number system conversion	**81-92**
Unit 6:	**Binary Arithmetic** Complement: r's and (r-1)' s complement, Binary addition, Binary subtraction, Binary Multiplication, Binary division.	**93-104**
Unit 7:	**Data Representation** Fixed Point representation and Floating point representation	**105-112**
Unit 8:	**Code Conversion Technique** Gray code, BCD, Code conversion technique, ASCII, EBCDIC, Unicode, ISO	**113-120**

Unit 9: Boolean algebra **121-129**

Introduction, Properties of Boolean Identities, Properties of Positive and Negative Logic, De-Morgan's Theorem, Application of De-Morgan's theorem.

Unit 10: Logic Gates **130-149**

Logic Gates: AND, OR, NOT, NAND, NOR, XOR; Conversion of the Logic Gates, NAND and NOR Implementation of Basic Gates.

Unit 11: Reduction Techniques **150-168**

Boolean Expression, Simplification of Boolean expression by Algebraic method and Karnaugh Map, SOP, POS, Standard and Canonical form, Conversion of SOP to POS and vice-versa, Reduction upto 3-variables.

Unit 12: Combinational Circuit **169-185**

Definition and Properties of Combinational Circuit, Introduction to Multiplexer, 4-to-1multiplexer, Demultiplexer, Encoder, Decoder

Unit 13: Binary Arithmetic Circuits **186-195**

Half adder, Full adder, Binary Half subtractor, Full subtractor

Unit 14: Sequential Circuit **196-207**

Definition and Properties of Sequential Circuit, Block Diagram of Sequential Circuit, Types of Sequential Circuit, Introduction to Flip-Flop and its different types: RS, JK; Master-Slave JK Flip-Flop.

Unit 15: Registers **208-217**

Types of Shift Registers: SISO, PISO, SIPO, PIPO;

Unit 16: Registers **218-225**

Counters Synchronous and Asynchronous Counter, Limitations of Asynchronous Counter.

COURSE INTRODUCTION

Welcome to the course on Computer Fundamentals and Digital Techniques! In this course, you will learn the basics of computers and digital electronics. We will start by introducing you to the evolution, generations, and classification of computers. You will also learn about the various components of a computer such as the CPU, ALU, CU, and computer memory, both primary and secondary. In addition, you will be introduced to various input/output devices and buses such as address, data, and control.

Computer security is an essential aspect of this course, and we will cover various topics such as computer viruses, worms, spyware, malware, Trojan horses, and antivirus software. You will also learn about different cyber-crimes against individuals, institutions, and states, such as hacking, digital forgery, cyberstalking/harassment, cyber pornography, identity theft, fraud, cyber terrorism, and cyber defamation. Moreover, we will discuss the different offenses under the IT Act, 2000.

Number systems are an integral part of computing, and we will cover topics such as decimal, binary, hexadecimal, and octal number systems, along with number system conversions. You will learn about binary arithmetic, complement r's and (r-1)'s complement, binary addition, subtraction, multiplication, and division. Furthermore, you will be introduced to data representation, fixed-point representation, and floating-point representation.

Boolean algebra is another crucial topic in this course, and we will cover the introduction, properties of Boolean identities, positive and negative logic, De-Morgan's theorem, and its application. You will also learn about logic gates such as AND, OR, NOT, NAND, NOR, and XOR, along with their conversions and implementations. We will then move on to reduction techniques, such as simplification of Boolean expression by algebraic method and Karnaugh Map, SOP, POS, standard and canonical form, and conversion of SOP to POS and vice-versa.

The course will also cover combinational circuits, such as multiplexers, demultiplexers, encoders, and decoders, along with their properties and definitions. Binary arithmetic circuits such as half adders, full adders, binary half subtractors, and full subtractors will also be discussed. You will also learn about sequential circuits, such as the block diagram of sequential circuits, types of sequential circuits, and the different types of flip-flops such as RS, JK, and master-slave JK flip-flop. Additionally, we will cover registers and counters, such as shift registers, their properties, applications, and types.

By the end of this course, you will have gained a solid understanding of computer fundamentals and digital techniques that are essential for anyone interested in pursuing a career in computer science, electronics, or related fields.

Each unit includes some along-side boxes to help you know some of the difficult, unseen terms. Some “EXERCISES” have been included to help you apply your own thoughts. You may find some boxes marked with: “LET US KNOW”. These boxes will provide you with some additional interesting and relevant information. Again, you will get “CHECK YOUR PROGRESS” questions. These have been designed for you to self-check your progress of study. It will be helpful for you if you solve the problems put in these boxes immediately after you go through the sections of the units and then match your answers with “ANSWERS TO CHECK YOUR PROGRESS” given at the end of each unit.

UNIT 1 : INTRODUCTION TO COMPUTER

UNIT STRUCTURE

1.1 Learning Objectives

1.2 Introduction

1.3 Computer Evolution and Generation

1.4 Classification of Computer

1.5 Applications of Computer

1.6 Let Us Sum Up

1.7 Further Reading

1.8 Answers to Check Your Progress

1.9 Model Questions

1.1 LEARNING OBJECTIVES

After going through this unit, you will be able to :

- describe the evolution of computer
- classify the different types of computer
- describe applications of computer.

1.2 INTRODUCTION

Computer is one of the most powerful tools man has ever created. Computers have made a great impact on our everyday life.Today, computer technology has permeated every sphere of existence of modern man. With the growing information needs in the modern world, the computer has become one of the vital components for the survival of the business houses. The presence of computers is felt in almost every working place from the bus, air and railway ticket counters to satellite launching, from retail stores to medical diagnosis and from home use to research organizations. Everywhere, we witness the elegance, sophistication and efficiency made possible only with the help of computers. Computer has become a part and parcel of our daily life.

In this unit, we will introduce you to the computer fundamentals including its various components and functionality. Besides discussing the characteristics of computers including its evolution and generations, this unit also throws light on the categories of computers along with its application in the various fields of the modern world.

1.3 COMPUTER EVOLUTION AND GENERATION

The term ***computer*** is derived from the word ***compute***, which means ***to calculate.*** It can be defined as an ***electronic machine*** capable of performing calculations and other manipulations of various types of data, under the control of a stored set of instructions. In the simplest form, a computer can be defined as a ***programmable machine***. A computer accepts data from an input device and processes it into useful information which it displays on its output devices. A general purpose computer generally has an *input unit*, an *output unit*, a *central processing unit* and a *memory unit*.

Most of today's computer designs are based on the concepts developed by **John Von Neumann** refered to as the "**Von Neumann architecture**". According to this architecture, a computer must have *two* units - a *processing unit* and a single *separate storage unit*. The term "*stored program computer*" is generally used to represent a computer of this design. The processing unit is termed as **central processing unit (CPU)** which mainly comprises of two units, namely Arithmetic and Logic Unit (**ALU**) and Control Unit (**CU**). The ALU performs the arithmetic operations on data such as addition, subtraction, multiplication and division, and the logical operations such as, >, <, >=, <= etc. The control unit directs the ALU to perform specific arithmetic and logical functions on the data. The storage unit is used to store instructions and data temporarily. This component is referred to as memory. The memory unit stores all the information in a group of memory cells, also called memory locations. Each memory location has a unique address and can be addressed independently.

The following figure shows the basic structure of a conventional Von Neumann machine :

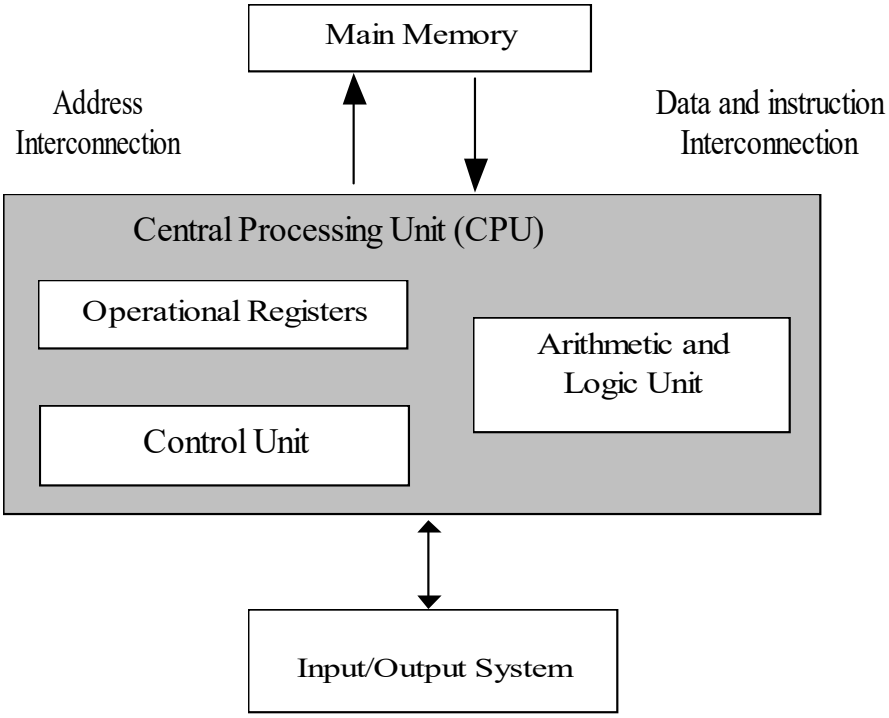


Figure 1.1 : Von Neumann Architecture

Depending on the major technological development, the whole evolution process of the computer can be categorised into five generations, where each generation has some special characteristics.

The *five* generations of computers are described as follows :

First Generation (1945 – 55) : The computers of this generation used vacuum tubes and consumed more power. Assembly languages were used for preparing programs. Mercury delay line memories and electrostatic memories were used for storing programs. Punched cards and paper tape were invented to feed programs and data to get results. The cost of the first generation computers was very high and they were mainly used for scientific computations.

Examples of computers of this generation are :

ENIAC – Electronic Numerical Integrator And Calculator

EDSAC – Electronic Delay Storage Automatic Calculator

EDVAC – Electronic Discrete Variable Automatic Computer

UNIVAC – Universal Automatic Computer IBM 701



NOTE :

All the arithmetic and logical operations are performed in the CPU in special storage areas called registers. The size of the register is one of the important considerations in determining the processing capabilities of the CPU. Register size refers to the amount of information that can be held in a register at a time for processing. The larger the register size, the faster may be the speed of processing CPU. Processing power is measured in Million Instruction Per Second (MIPS).

Second Generation (1955 – 65) : The second generation computers used transistors. They consumed less power and were smaller in size. Magnetic ferrite core memories were used as main memory and magnetic tapes and magnetic disks were used as secondary memory. High level languages such as FORTRAN, COBOL etc were used.

Examples of computers of this generation are– IBM 7030, Digital Data Corporation's PDP 1/5/8 Honeywell 400.

Third Generation (1965 – 75) : The third generation computers used IC(Integrated Circuits). Small Scale Integration and Medium Scale Integration technology were implemented during this generation. Semiconductor memories were introduced in this generation. The computers have became smaller in size with fast processors and the cost of the computers is less compared to second generation.

Examples of computers of this generation are - 360 Mainframe from IBM, PDP-8 Mini Computer from Digital Equipment corporation.

Fourth Generation (1975 – 89) : Microprocessors were introduced as CPU. Very large Scale Integration (VLSI) techniques were used during this generation. Semiconductor memory chips were used as main memory. Hard disks, floppy disks, magnetic tapes were used as secondary memory. Lots of techniques like parallelism, pipelining, cache memory, virtual memory and networking techniques were applied during this generation.

Examples of computers of this generation are - Intel's 8088, 80286, 80386, 80486 .., Motorola's 68000, 68030, 68040, Apple II, CRAY I/2/X/MP etc.

Fifth Generation (1989 to present) : Computers based on artificial intelligence belong to the fifth generation. ULSI (Ultra Large Scale Integration) technology used in fifth generation computer. For example, Intel's Pentium 4 microprocessor contains 55 million transistors, millions of components on a single IC chip. Superscalar processors, Vector processors, SIMD processors, 32 bit micro controllers and embedded processors, Digital Signal Processors (DSP) etc have been developed in fifth generation. Recent development of grid technology and nanotechnology in this generation will radically change the phase of computers.

Examples of computers of this generation are– IBM notebooks, Pantium PCs-Pentium 1/2/3/4/Dual core/Quad core.. SUN work stations, Origin 2000, PARAM 10000, IBM SP/2).

1.4 CLASSIFICATION OF COMPUTER

Computers can be classified according to their size and power as follows :

- Micro computer
- Mini computer
- Mainframe
- Supercomputer

(I) Micro Computers

- A microcomputer uses a microprocessor as its central processing unit. Microcomputers are tiny computers that can vary in size from a single chip to the size of a desktop model.
- They are designed to be used by only one person at a time.
- Small to medium data storage capacities 500MB – 2GB or more

The common examples of microcomputers are chips used in washing machines, TVs, cars, note book and personal computers.

Eg. : IBM PC, PS/2 and Apple Macintosh

Applications : Used in the field of desktop publishing, accounting, statistical analysis, graphic designing, investment analysis, project management, teaching, entertainment etc. The different models of microcomputers are given below :

- **Personal computers :** The name PC was given by the IBM for its microcomputers. PCs are used for word processing, spreadsheet calculations, database management etc.
- **Note book or Laptop :** It is small in terms of size and can be folded and carried around. Its monitor is made up of LCD and the keyboard and system units are contained in a single box. It has all the facilities of a personal computer (HDD, CDD, Sound card, N/W card, Modem etc) and a special connection to connect

to the desktop PC which can be used to transfer data.

- **Palm Top :** Smaller model of the microcomputer. Its size is similar to that of a pocket size calculator. It has a processor, memory and a special connection to connect to the desktop PC which can be used to transfer data.
- **Personal Digital Assistance :** A personal digital assistant is a small palm-sized hand-held computer which has a small colour touch screen with audio and video features. They are used as smart phones, web-enabled plamtop computers, portable media players or gaming devices. Most PDAs today have a touch screen for data entry, a memory card, bluetooth, Wi-Fi or an infra red connectivity and can be used to access the Internet and other networks.



a) Personal computer



b) Notebook computer



d) Palmtop computer



e) PDA (personal digital assistant)

Figure. 1.2 : Micro Computer

(II) Mini Computers

- Perform better than micros
- Large in size and costlier than micros
- Designed to support more than one user at a time

- Possess large storage capacities and operate at higher speed
- Support faster peripheral devices like high speed printers
- Can also communicate with main frames

Applications :

- These computers are used when the volume of processing is large. For example, data processing for a medium sized organization
- Used to control and monitor production processes
- Used to analyze results of experiments in laboratories
- Used as servers in LANs (Local Area Networks)

Examples of mini computers are - Digital Equipments PDP 11/45 and VAX 11 etc. The figure 1.3. shows a mini computer.

(III) Mainframe Computers :

- Are able to process large amount of data at a very high speed
- Support multi-user facility
- Number of processors varies from one to six.
- Cost: 3500 to many million dollars
- Are to be kept in air conditioned room to keep them cool
- Support many I/O and auxiliary storage devices
- Support network of terminals



Figure. 1.3 : Mini computer



Figure. 1.4 : Mainframe computer

Applications :

- Mainframe computers are used to process large amount of data at a very high speed such as in the case of banks insurance companies hospitals railways which need online processing of large number of transactions and requires massive data storage and processing capabilities
- Used as controlling nodes in WANs (Wide Area Networks)
- Used to mange large centralized databases

Examples of main frame computers are– IBM 370, IBM 3081, IBM 3000 series, Univac 1180, DEC IBM 3000 series, CDC Cyber - 2000V etc.

(IV) Super Computer

Supercomputer is a broad term for one of the fastest computers currently available. The main characteristics of a super computer are :

- Most powerful Computer system -it needs a large room
- Minimum world length is 64 bits
- CPU speed: 100 MIPS (Million instruction per second)
- Equivalent to 4000 computers
- High cost: 4 – 5 millions
- Able to handle large amount of data
- High power consumption
- High precision



Figure. 1.5 : PARAM 1000 Series Super Computer

Applications :

- In petroleum industry - to analyze volumes of seismic data which are gathered during oil seeking explorations to identify areas where there is possibility of getting petroleum products inside the earth.
- In Aerospace industry - to simulate airflow around an aircraft at different speeds and altitude. This helps in producing an effective aerodynamic design for superior performance.
- In Automobile industry – to do crash simulation of the design of an automobile before it is released for manufacturing – for better automobile design.

- In structural mechanics – to solve complex structural engineering problems to ensure safety, reliability and cost effectiveness. For example the designer of a large bridge has to ensure that the bridge must be proper in various atmospheric conditions and pressures from wind, velocity etc and under load conditions.
- Meteorological centers use super computers for weather forecasting.
- In Biomedical research – atomic nuclear and plasma analysis – to study the structure of viruses.
- For weapons research and development, sending rockets to space etc.

Examples of supercomputers are : Cray-1(1976), Cray-2(1985), Cray T3D(1993), NEC's SX-S/44(1991), Fujitsu VP 2600/10(1991), Hitachi 820/80(1987), C-DAC's PARAM Series of supercomputers etc. A PARAM 10000 series super computer of IIT Guwahati is shown in figure 1.5.

1.5 APPLICATIONS OF COMPUTER

Now a days, computers are used almost in every aspect of life. Every company, small or large, government offices, educational institutions are now directly or indirectly dependent on computers mainly for information processing. Computer based railway and airway reservation are common examples of computer applications. Computer system are helping in the efficient management of the banking sector, hospital records, payroll records and so on. Some of the areas where computers are being used mostly can be listed as below:

Science : Scientists are using computers to carry out their research works based on complex computations because of computer's fast speed and the accuracy.

Education : In schools and colleges, to make education much more interesting, computers are used now a days. Computer Aided Education (CAE) and Computer Based Training (CBT) packages are making learning much more interactive.

Health and Medicine : Starting from diagnosing the illness to monitoring a patient's status during a surgery, in pathological analysis ,in CAT scans or MRI scans etc. Some special purpose computers are available which can even be operated within the human body.

Engineering : Engineers and architects are using computers in designing machineries, drawing design layouts. Architects use computers to view objects that can be viewed from all the three dimensions by using techniques like virtual reality. In manufacturing industries, using computerized robotic arms hazardous jobs can be performed. The packages like Computer Aided Designing (CAD), Computer Aided Manufacturing (CAM) and so on are used in designing the product, ordering the parts and planning production.

Entertainment : With the use of multimedia facilities, computers are now greatly used in entertainment industry. Computers are used to control and bring special effects on image and sound.

Communication : Computer network and finally the Internet has brought a drastic change in communication system. Through E-mail or Electronic mail, it is possible to send messages and reports very fast from one person to another or a group of persons with the aid of computers and telephone lines.

Business and Banking : In order to deposit or withdraw cash from bank, people can use ATM (Automated Teller Machine) services 24 hours of the day. Through the computer networks among different branches of a bank, inter branch transactions can be carried out without delay.

Apart from the above mentioned applications, there are many other applications of computers that can be seen in our day to day activities.



CHECK YOUR PROGRESS

Q1. State True or False:

i) PARAM is a super computer.

ii) A laptop is a portable computer.

iii) Vacuum tubes were part of the second generation computers.

iv) Micro computers are more powerful than mini computers.

v) EDSAC is an example of a second generation computer.

Q2. Fill in the blanks :

i) Physical components of a computer are called _____.

ii) The basic components of first generation computers was _____.

iii) PDA stands for _____.

iv) _____ is a very small computer that can be held in the palm of the hand.

v) Analytical Engine was developed by _____.

Q3. What is a PDA?

.....

Q4. Name some storage device used in computer.

.....



ACTIVITY

- Create a table on various generations of computers with hardware and software details.
- Prepare a list of areas where computers are used. Mention the specific software for each application and its significance.



1.6 LET US SUM UP

- Computer is an electronic device that is used to perform diverse operations with the help of instructions to process the data in order to produce desired results.
- Computer development is divided into five generations.
- The first generation of computers used **vacuum tubes** for circuitry and **magnetic drums** for memory, and were often enormous, taking up entire rooms.
- In the second generation of computers, **transistors** replaced vacuum tubes.
- The **integrated circuits** were used in the third generation of computers.
- The fourth generation of computers are based on **microprocessors.**
- Fifth generation computers are based on ***artificial intelligence.***
- Micro computer is a small, low cost digital computer and includes desktop, laptop, PDA.
- A mini computer is a small digital computer which is more powerful than a micro computer.
- A mainframe computer consists of several processors that carry out most of the computations and have direct control over all the other computers in a network.
- Supercomputers are the most powerful computers used for processing complex scientific applications that involve task with highly complex calculations and solving problems with mechanical physics, such as weather forcasting and climate research systems, nuclear weapon simulation and simulation of automated aircrafts.



1.7 FURTHER READING

- Sinha, P. K., & Sinha, P. (2010). Computer fundamentals (Vol. 4). BPB publications.

- Ram, B. (2000). Computer fundamentals: architecture and organization. New Age International.

1.8 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : i) True, ii) True, iii) False, iv) False, v) False

Ans. to Q. No. 2 : i) Hardware, ii) Vacuum tubes, iii) Personal Digital Assistant, iv) Palmtop, v) Charles Babbage.

Ans. to Q. No. 3 : A personal digital assistant is a small palm-sized hand-held computer which has a small colour touch screen with audio and video features.

Ans. to Q. No. 4 : Hard Disk, CD-ROM, DVD-ROM, Pen Drive etc.

1.9 MODEL QUESTIONS

Q1. Who invented the concept of stored program ? Why is this concept so important ?

Q2. List the key hardware technologies used in building computer of each of the five generations.

Q3. What are the advantages of transistors over vacuum tubes?

Q4. What is an IC ? How does it help in reducing the size of computers

Q5. Discuss the important features of various generations of computers. Give some examples of the computers of each generation.

Q6. What is computer? What are the characteristics of a Computer? Do you think computers are superior to human being?

Q7. Describe the evolution of computers.

Q8. How are computers categorized into different categories?

****

UNIT 2 : BASIC COMPONENTS OF COMPUTER

UNIT STRUCTURE

2.1 Learning Objectives

2.2 Introduction

2.3 Concepts of Bits and Bytes

2.4 Block Diagram of a Computer

2.5 Computer Memory

- 2.5.1 Primary Memory
- 2.5.2 Secondary Memory

2.6 Input Devices

- 2.6.1 Keyboard
- 2.6.2 Mouse
- 2.6.3 Hand-held Devices
- 2.6.4 Optical Input Devices

2.7 Output Devices

- 2.7.1 Display Devices
- 2.7.2 Printers

2.8 Computer Bus

2.9 Let Us Sum Up

2.10 Further Reading

2.11 Answers to Check Your Progress

2.12 Model Questions

2.1 LEARNING OBJECTIVES

After going through this unit, you will be able to :

- describe the concept of bits and bytes
- understand the block diagram of a computer
- discuss the primary and secondary memory in computer
- analyse the different types of input and output devices
- describe the different types of buses

2.2 INTRODUCTION

A computer system is composed of various components which perform many functions. In this unit, we will introduce you to the basic components of a computer system where, you will first learn how the data is represented in the form of bits and bytes. Through this unit you will be acquinted with the types of computer memory. Various types of input and output devices and the basic functioning of a bus are also included as a part of discussion in this unit.

2.3 CONCEPTS OF BITS AND BYTES

You may be surprised to know that the computer is made up of millions of tiny electric circuits that are not visible to us. The memory elements are also built with these tiny circuits which are used for storing data. For every circuit in a computer chip, there are two possibilities:

- an electric current flows through the circuit, or
- an electric current does not flow through the circuit.

When an electric current flows through a circuit, the circuit is on. When no electricity flows, the circuit is off. An "on" circuit is represented by the number one (1) and an off circuit is represented by the number zero (0). The number 1 or 0 is called ***bit***. A number system that uses only two digits, 0 and 1, is called the ***binary number system***. The binary number system is also called the base two system. The two symbols 0 and 1 are known as binary digits. The word **bit** comes from "**bi**nary digi**t**".

Always remember that each time a computer reads an instruction, it translates that instruction into a series of bits, 1's and 0's. On most computers

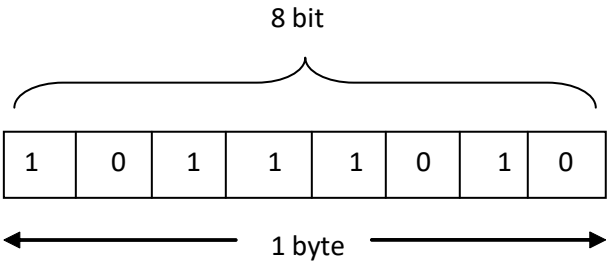


Figure 2.1 Byte representation

every character from the keyboard is translated into eight bits, a combination of eight 1's and 0's. Each group of 8 bits is called 1 byte as shown in figure 3.1. Each group of four bits is called a nibble.

Thus, 8 bits = 1 byte and 4 bits = 1 nibble

and 1024 byte = 1 kilo byte

We know that, all the quantities, physical or abstract, can be measured in some units. For example, time is measured in seconds; length is measured in metres, and mass in grams. Similarly, the computer memory can be measured in terms of bits, bytes, kilobytes, megabytes, gigabytes etc.

The followings are the measuring units given in terms of bytes :

1 kilobyte (KB) = 1,024 byte

1 megabyte (MB) = 1,048,576 bytes

1 gigabyte (GB) = 1,073,741,824 bytes

1 terabyte (TB) = 1,099,511,627,776 bytes

2.4 BLOCK DIAGRAM OF A COMPUTER

All general-purpose computers require the following hardware components:

Input Unit : This is the unit through which data and instructions are entered into a computer. Usually a keyboard and mouse are the common input devices.

Output Unit : This unit communicates the results to the user. A display screen and a printer are the common devices that let you see the results in the computer.

Central Processing Unit (CPU) : Execution of programs is the main function of the computer. The program is a set of instructions that is stored in the computer's memory. A task becomes completed when the instructions of the program are executed by the central processing unit. All the major calculations and comparisions are carried out inside the CPU. The CPU is responsible for activating and controlling the operations of various units (input & output) of the computer system.

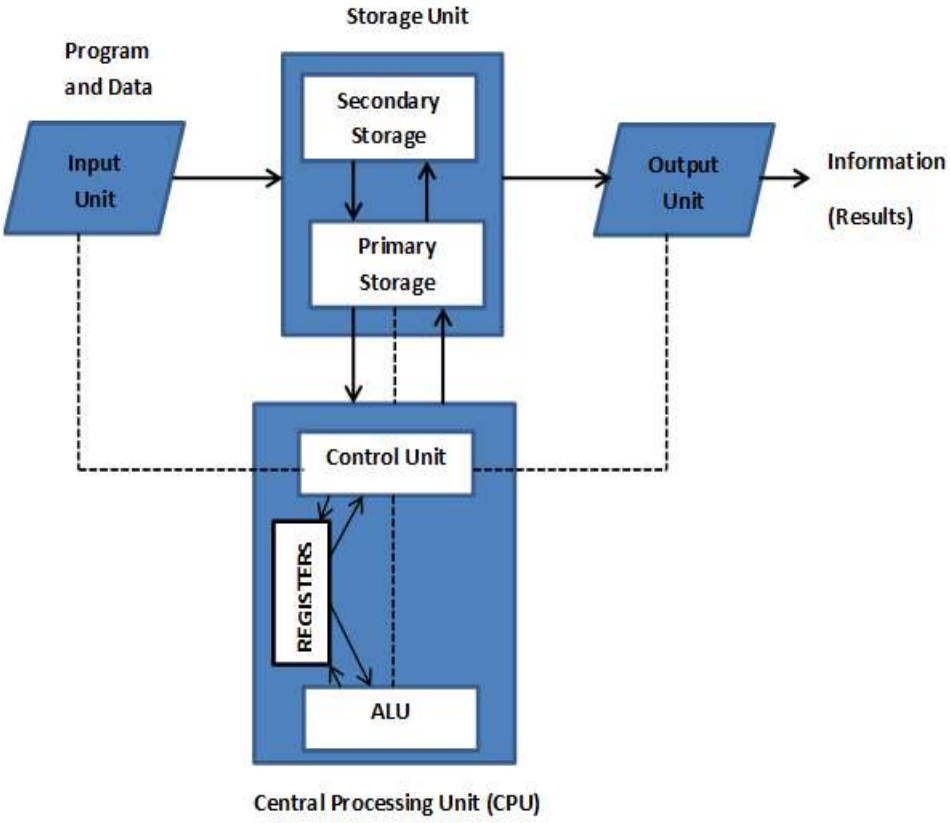


Figure 2.2: Central Processing Unit

The CPU is made up of three major components : the **registers,** the a**rithmetic and logic unit (ALU)** and the **control unit**. The registers are responsible for storing the intermediate data during the execution of program. The control unit supervises the transfer of information between the registers and instructs the ALU to perform the required operation.

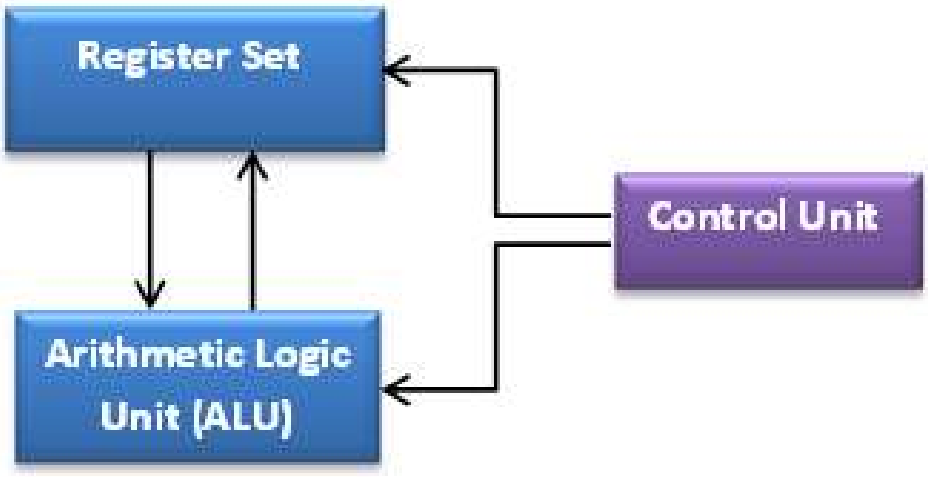


Figure 2.3 : Components of CPU

Control Unit (CU) : The control unit controls and guides the interpretation, flow and manipulation of all the data and information. It does not perform the actual processing of data, but manages and coordinates the entire computer system, including the input and output (I/O) devices. Control unit retrieves and interprets the instructions from the main memory and send signals to the other units to execute them one after the other. Some special purpose registers found inside the CPU like *instruction register, program counter* etc. help the CU to perform its function smoothly.

Arithmetic Logic Unit (ALU) : The ALU performs all the arithmetical functions +, -, *, / and some logical functions like <, >, =, <=, >=, <> etc. When two numbers are required to be added, these numbers are sent from memory to ALU where addition takes place and the result is sent back to memory. ALU has special circuits to carry out the logical operations.

Storage Unit : The storage unit allows a computer to permanently store large amount of data. The storage unit can be categorised into two types namely *primary storage* and *secondary storage,* which we will discuss in the next section. Common mass storage devices include disk drives and tape drives etc.

2.5 COMPUTER MEMORY

The term ***memory*** usually refers to a form of semiconductor storage and sometimes other forms of fast but temporary storage. In a computer, the data and instructions are stored and subsequently retrieved from the memory. We know that the CPU has some storage elements known as registers for storing data and instructions during the program execution. Every computer requires storage space for storing programs and data temporarily during the time of program execution or storing the data permanently for future use. Generally, a computer has three set of memories:

a) **Primary storage or main memory :** Primary memory, also known as *memory*, is the only one *directly accessible to the CPU*. The CPU continuously reads instructions stored there and executes them. Any data actively operated on is also stored there in uniform manner. It is a form of semiconductor storage known

as random access memory (**RAM**) and sometimes other forms of fast but temporary storage. It is small-sized, light, but quite expensive. This type of memory is divided into the following two types : *RAM (Random Accress Memory) & ROM (Read Only Memory)*.

b) **Secondary storage or auxiliary memory :** Secondary memory or *storage* provides the facility of storing information and programs permanently. It differs from primary storage in that it is *not directly accessible by the CPU*. The computer usually uses its input/output channels to access secondary storage and transfers the desired data using intermediate area in primary storage. Secondary storage does not lose the data when the device is powered down; it is non-volatile. It is less expensive than primary storage.

c) **Internal processor memory :** It consists of a small set of high-speed registers that are used as temoprary locations inside the processors during the execution of instructions.

Another type of memory, is the *cache memory*. Its logical position is between the main memory and the internal memory(registers). Its primary function is to store and cache part of the main memory's contents that the processor is currently using.

The computer system has a memory hierarchy consisting of the storage devices in it. A typical memory hierarchy is illustrated in the figure 2.4.

There are three key characteristics of the memory. They are cost, capacity and access time. On moving down the memory hierarchy, it is found that the cost of the storage devices decreases but their storage capacity as well as the memory access time increases. In other words, the smaller memories are more expensive and much faster. These are supplemented by the larger, cheaper and slower storage devices.

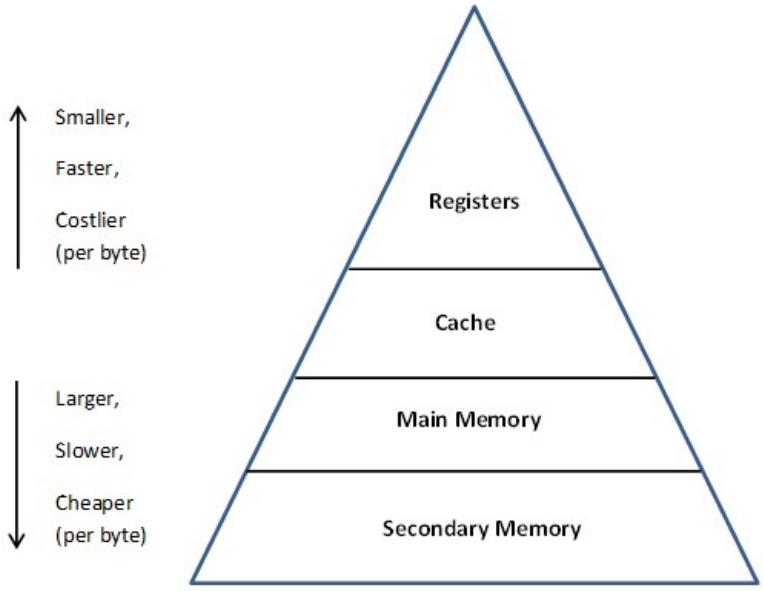


Figure 2.4 : Memory hierarchy

Thus, from the above figure it can be seen that the registers are at the top of the hierarchy and so provides the fastest, the smallest and the most expensive type of memory device for the CPU to access data.

Memory Capacity : Capacity in a computer system is defined in terms of the number of bytes that it can store in its main memory. This is usually stated in terms of kilobytes (KB) which is 1024 bytes or megabyte which is 1,048,576 bytes. The rapidly changing memory capacity of computer system has resulted in defining the capacity in terms of gigabytes (GB) which is 1,073,741,824 bytes. Thus, a computer system having a memory of 1GB is capable of storing (1024 X 1024 X 1024) bytes.

2.5.1 Primary Memory

Primary memory can be categorised into two types - RAM and ROM. We will now briefly describe RAM and ROM.

Random Access Memory (RAM) : A computer's **main memory** is often referred to as RAM - an area in the computer system unit that temporarily holds user data, operating system instructions and program instructions. Every time we turn on our computer, a set of operating system instructions is copied from hard disk into RAM.

These instructions, which help control basic computer functions, remain in RAM until we turn the computer off.

RAM features :

a) RAM is the place which holds the data that has to be processed by the computer's processor. Moreover, it holds the operating system and other application programs also.

b) RAM is made up of different ICs (Integreted Circuits). They are also called *semiconductor memory*.

c) Each element of RAM is a memory location in which data can be stored. Each location has a **unique address**. Using this address, data can be directly retrieved or stored.

d) Since RAM must hold both the data to be processed and the instructions for processing, its size or capacity is one of the measures of power of the computer.

e) RAM is a volatile memory. It loses its data when the computer is turned off.

There are generally two broad categories of random access memory :

i) **DRAM** memories (*Dynamic Random Access Memory*)

ii) **SRAM** memories (*Static Random Access Memory*)

DRAM : Dynamic RAM is a type of RAM in which data is stored in a storage cell, consisting of transistors and capacitors. A common property of a capacitor is its tendency to get discharge, it means the current goes down in the circuit and data cannot hold for long time. To overcome this effect, the DRAM needs to be continuously refreshed with power supply and thus it has one *refreshing circuit* which rewrites the data in certain time intervals. This circuit refreshes or rewrites the data several hundred times in a second.

SRAM : Static RAM is a type of RAM that holds its data as long as power is supplied to the circuit. The SRAM does not need any refreshing circuit. SRAMs are used for specific applications within the PC. The SRAM has the following advantages:

- **Simplicity :** SRAMs don't require external refresh circuits to keep their data intact.
- **Speed :** SRAM is faster than DRAM.

In contrast, SRAMs have the following weaknesses, compared to DRAMs:

- **Cost :** SRAM is more expensive than DRAM.
- **Size :** SRAMs take up much more space than DRAMs

Read Only Memory (ROM) : One major type of memory that is used in PCs is called *read-only memory*, (*ROM*). ROM is the memory that stores the data permanently i.e. the instructions in ROM are permanent whether the power is on or off. We have no way to change them, unless we remove the ROM chips from the main board and replace them with another set.

Read-only memory is most commonly used to store system-level programs that we want to be made available to the PC at all times. For example, the ROM-BIOS (ROM basic input and output services) programs are stored in ROM chips which are used to boot up (start up) the computer.

The following are the different types of ROMs:

Programmable ROM (PROM) : This is a type of ROM that can be programmed using special equipment; it can be written to, but only once.

Erasable Programmable ROM (EPROM) : *EPROM* is a ROM that can be erased and reprogrammed. Ultraviolet light of a specific frequency is used for erasing the EPROM and for allowing it to be reprogrammed again.

Electrically Erasable Programmable ROM (EEPROM) : The next level of erasability is the *EEPROM*, which can be erased under software control. This is the most flexible type of ROM, and is now commonly used for holding BIOS programs.

2.5.2 Secondary Memory

Secondary memory or *storage* provides the facility of storing information and programs permanently. It differs from primary storage in that it is not directly accessible by the CPU. The computer usually uses its input/output channels to access secondary storage and transfers the desired data using intermediate area in primary storage. Secondary storage does not lose the data when the device is powered down; it is non-volatile. It is also typically less expensive than primary storage. Consequently, modern computer systems typically have an order of magnitude more secondary storage than primary storage and the data is kept for a longer time there. The figure 2.5 shows the commonly used secondary storage devices and their classification.

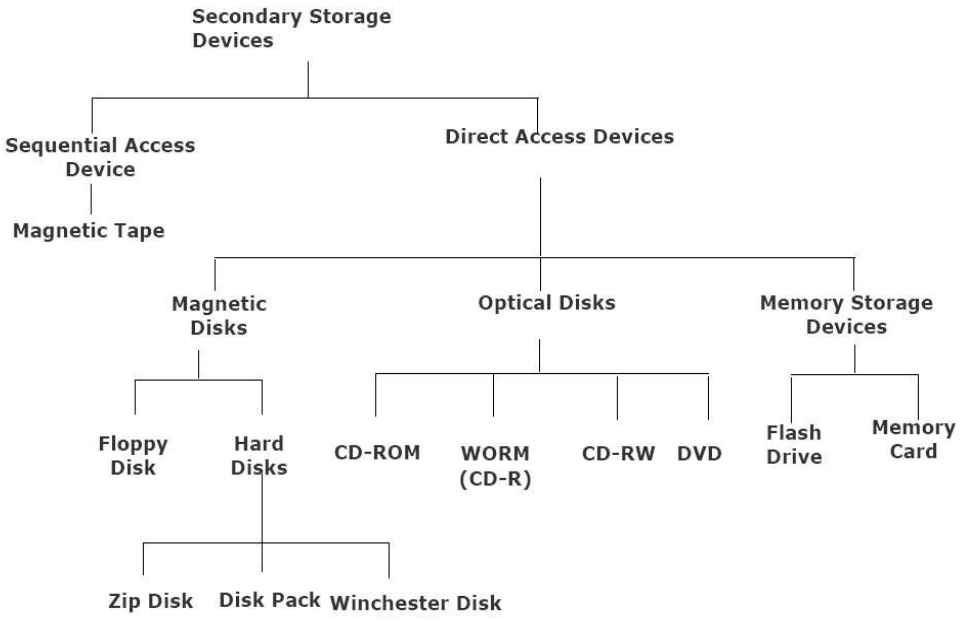


Figure 2.5 : Secondary storage devices and their classification

In modern computers, hard disks are usually used as secondary storage. The time taken to access a given byte of information stored on a hard disk is typically a few thousandths of a second, or milliseconds. By contrast, the time taken to access a given byte of information stored in random access memory is measured in

thousand-millionths of a second, or nanoseconds. Some other examples of secondary storage technologies are: Flash Memory (e.g. USB sticks or keys), Floppy Disks, Magnetic tape, Paper tape, Punch Cards, standalone RAM disks, and Zip drives.



CHECK YOUR PRPOGRESS

Q1. State True or False:

i) The secondary memory is slower than the main memory but has a large capacity.

ii) In random access memory any memory location can be accessed independently.

iii) Cache memory increases load on main memory.

iv) Flash memory is one kind of ROM.

v) High speed memories are needed to bridge the gap of speed between I/O device and memory.

vi) Registers are the fastest memory element in computer system.

vii) Main memory of a computer system cannot store very large data.

Q2. Which memory is known as the main memory of the computer system ?

.....

.....

Q3. What are the differences in :

a) Volatile versus Non-volatile memory

.....

.....

.....

b) Static RAM versus dynamic RAM

.....

.....

.....

.....

Q4. What is the similarity between ROM and PROM?

.....

.....

2.6 INPUT DEVICES

The main function of the input unit of a computer system is to provide information for execution. Programs and data must be entered into the computer memory for processing through the input devices, and the results obtained from the execution by the CPU must be displayed or recorded for the user's. The most commonly used input devices can be classified into the following categories :

- **Keyboard devices**
- **Point and draw devices** (mouse, track ball, joystick, light pen, touch screen)
- **Scanning devices** (optical mark recognition, magnetic ink ch aracter recognition, optical bar code reader, digitizer, electronic card reader)
- **Voice recognition devices**
- **Vision-input devices** (Webcam, video camera etc)

2.6.1 Keyboard

It is the main input device used in most of the computer systems. It has letter and number (alphanumeric) keys, function keys, computer specific task keys, that allow the user to use an English-like language to issue instructions to an electronic environment. It uses a cursor to keep your place on the screen and to let you know where to begin typing. You are able to input commands, type data into documents, compose documents, draw pictures with use of certain keys, pull down menus, and respond to prompts issued by

the computer. The keyboard contains s*pecial function keys* labelled F1, F2, ..., F12 to manipulate the user interface. When a key is touched, an electrical impulse is sent through the device which is picked up by the operating system software, and sent through the computer to be processed.

The keyboard operates as a typical typewriter and uses a standard “QWERTY” keyboard. QWERTY is the way the keyboard is set up for typing. If you look at the keyboard under the top number row, you will see that the alphabet top row begins with QWERTY.

Some special purpose keyboards are also found which are used for specific applications. An example of such kind of keyboards can be seen at the Automatic Teller Machines or the ATMs where the keyboard is required for limited functionality by the customers.

2.6.2 Mouse

The mouse is a hand held pointing device that lets you to point and select items on your screen. In any mouse there are mostly 2-3 buttons and on a Mac there is one. A ball under the mouse senses movement. The mouse is included in almost every computer that is sold today. Besides becoming an important input tool, it has provided access to the computer for many individuals with disabilities that might not otherwise have the opportunity to use the computer.

Variants of a Mouse :

Touchpad : A touchpad is a touch sensitive input device which takes user input to control the onscreen pointer and perform other functions similar to that of a mouse. Instead of having an external peripheral device such as a mouse, the touchpad enables the user to interact with the device through the use of one or multiple fingers being dragged across the relative positions on a sensitive pad. These touch pads are mostly found in Notebook and Laptops.Touchpads are pressure and touch sensitive.



Figure 2.6 : Touchpad of a Laptop

2.6.3 Hand-held Devices

Trackball : The trackball is an upside-down mouse. It uses the same principle as the mouse except that the rollers are reversed and the ball is on top. This ball does not need as much attention as the normal mouse because the only thing that touches it is your hand as the normal mouse touches a surface.

Joystick : The joystick is a vertical stick that moves the graphic cursor in the direction the stick is moved. It consists of a spherical ball, which moves within a socket, and has a stick mounted on it.The user can move the ball with the help of the stick in any direction. Video games, training simulators and control panel of robots are some common uses of a joystick.



Figure 2.7 : Trackball



Figure 2.8 : Joystick

Light pen : The light pen is a pen shaped tool that allows the user to touch the computer screen with a lighted pen to activate commands and make selections. Light pens are typically used in

CAD (Computer Aided Design) applications to directly draw on screen.

Touch Screen : A touch screen is probably one of the simplest and most intuitive of all input devices. It uses optical sensors in, or near, the computer screen that can detect the touch of a finger on the screen.

2.6.4 Optical Input Devices

Scanner : Scanning devices are input devices used for direct data entry from the source document into the computer system. Scanners capture and store information in a graphical format, and then display it on the graphical screen. A scanner consists of two parts, one that illuminates the page for capturing the optical image, and the other that converts and stores the graphical image in a digital format. Once this is done, the computer can then see and directly process this scanned graphical image. A scanner's sophistication is the ability to translate an unlimited number of analog voltage levels to digital values. The scanner takes the information it sees on a page and converts into code that the computer can use. A picture once scanned can be edited, printed, or used in an application. Capturing information by using scanners reduces the possibility of human error typically seen during large data entry.

Optical Mark Recognizer (OMR) : Optical mark recognization devices sense the marks on computer readable paper. They are usually used by academic institutions for grading aptitude tests where the correct option is marked from multiple options given on special kinds of answer sheets. The OMR device directly reads this paper, and the computer can then use it for further processing.



Figure 2.9: Magnetic Ink Character Recognizer

Magnetic Ink Character Recognition (MICR) : MICR is similar to optical mark recognition and is used exclusively by the banking industry. The banking industry makes use of MICR devices for directly reading the account numbers written on cheques and for processing them further as required. MICR characters are printed in special typefaces with a magnetic ink or toner, usually containing iron oxide.

Optical Barcode Reader (OBR) : Data coded in the form of small vertical lines forms the basis of barcoding. Alphanumeric data is represented using adjacent vertical lines called barcodes. These are of varying width and the spacing between them is used to uniquely identify books, merchandise in stores, postal packages etc. A barcode reader uses laser beam technology. The laser beam is moved across the patterns of bars in a barcode. These bars reflect the beams in different ways. The reflected beam is then sensed by a light-sensitive detector, which then converts the light patterns into electrical pulses, thereby transmitting them to logic circuits for further conversion to alphanumeric value.



Figure 2.10: Optical Bar Code

Digitizer : The digitizer is another kind of input device used for converting (digitizing) picture,maps and drawings into digital form for storage in computers. These devices are commonly used in the

area of Computer Aided Design (CAD) by architects and engineers to design car, buildings, medical devices, robots and mechanical devices.

Electronic Card Reader : Card readers are devices that allow direct data input into computer system. The electronic-card reader is connected to a computer system and reads the data encoded on an electronic card and transfers it to the computer system for further processing. Typical example of electronic cards are the plastic coated cards provided by the banks to its customers for use in Automatic Teller Machines or ATMs.

2.7 OUTPUT DEVICES

The outputs produced by a computer system can be in two forms : *hard copy and soft copy*.

Hard copy : The output produced in a physical form is called hard copy output. This type of output is permanent and a relatively stable form of output. Paper is one of the most commonly used hard copy output media. The most widely used hard copy output device is the printer.

Soft copy : The output that remains in electronic form, stored in computer memory or on disk are called soft copy outputs. This kind output is not permanent and is usually displayed on monitor or screen. It includes both audio and visual forms of output.

A wide range of output devices are available today which are :

- Display device (monitors, multimedia projectors, terminals)
- Plotters (flatbed, drum)
- Voice response system (voice response system, speech synthesizer)
- Printers

2.7.1 Display Devices

The display device shows the processed output from a computer system. The increasing demand for displaying the graphics and

pictures for presenting information in a visual form has led to the popularity of graphics display device. Display screen technology may belong to one of the following categories :

a) **Cathod Ray Tube(CRT) :** The CRT is a vacuum tube containing an electron gun (a source of electrons) and a fluorescent screen, with internal or external means to accelerate and deflect the electron beam. It is used for creating images in the form of light emitted from the fluorescent screen. A CRT display can be either of the following two types : *Raster scan display and Vector display*.

b) **Liquid Crystal Display(LCD) :** The LCD technology is a thin, flat electronic visual display that uses the light modulating properties of liquid crystals. They are common in consumer devices such as video players, gaming devices, clocks, watches, calculators, and telephones. LCDs have replaced cathode ray tube (CRT) displays in most applications. They are usually more compact, less energy consuming, lightweight, portable, less expensive, more reliable, and easier on the eyes.

Monitor : The monitor is one of the most commonly by used output devices. It can be thought of as a high resolution TV set. The monitor displays text and images (pictures) converted to output from the video adapter. The video adapter is an expansion card that plugs into the motherboard. The adapter changes the instructions from the central processing unit (CPU) into a way that the monitor can understand it.

Most computer monitors are based on Cathode Ray Tube (CRT) technology. The basic operation of these tubes is similar to that in television sets. With the widespread use of smaller computers like PDAs and laptops now-a-days, a new type of display screen Liquid Crystal Display (LCD) has been used.

2.7.2 Printers

It is a widely used output device. A printer prints information and data from the computer onto a paper. Some printers can produce only textual information but some others can produce graphics also.

Printers can be classified into categories based on : *printing technology, printing speed, printing quality.*

Printing technology : Printers can be divided into two types - *(i)impact printers* and *(ii)non impact printers*.

(I) Impact printers : The printers that work by physically striking a head or needle against an ink ribbon to make mark. Impact printers are divided into *Dot matrix printers , Daisy-Wheel printers* and *Drum printers.*

Dot Matrix Printer : This printer is the cheapest but loudest of all the printers. This printer prints one character at a time. The printer uses a fine stripe of carbon and depending on what pin it is, will determine the quality of the printing. The pin sizes range from 9-24. The print head hits the ink ribbon with one of its many pins causing it to form a letter number or character.



Figure 2.11 : A Dot-Matrix Printer

With the pins hitting the ribbon this printer becomes quite loud. The ppm (page per minute) ranges from 1 to 3. But with new changes in the technology it is being more and more used in offices today and it can also print 3 - 6 ppm. The drawback in most dot matrix printers is the on-roll paper which has a track which feeds it through

the printer. Also, in most new dot matrix printers there is a paper feed which allows most forms of paper.

Daisy Wheel Printer : Daisy-wheel printers operate in much the same fashion as a typewriter. A hammer strikes a wheel with petals (the daisy wheel), each petal containing a letter form at its tip. The letter form strikes a ribbon of ink, depositing the ink on the page and thus printing a character. By rotating the daisy wheel, different characters are selected for printing.You can change the daisy wheel to print different fonts. This printer is also called the letter quality printer.

Drum printer : Drum printer is one of the most commonly used line printers.

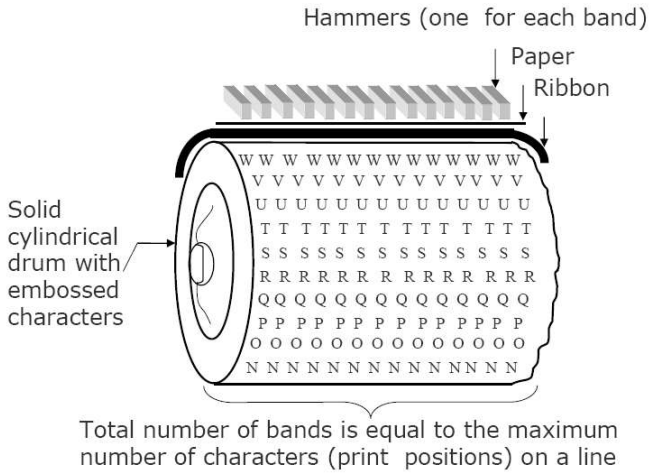


Figure 2.12 : Printing Mechanism of a Drum Printer

In a typical drum printer design, a fixed font character set is engraved onto the periphery of a number of print wheels, the number matching the number of columns (letters in a line) the printer could print. The wheels, joined to form a large drum (cylinder), spin at high speed and the paper and an inked ribbon are stepped (moved) past the print position. As the desired character for each column passes the print position, a hammer strikes the paper from the rear and presses the paper against the ribbon and the drum, causing the desired character to be recorded on the continuous paper. Because the drum

carrying the letterforms (characters) remains in constant motion, the strike-and-retreat action of the hammers has to be very fast. Typically, they are driven by voice coils mounted on the moving part of the hammer.

(II) Non impact printers : These printers work by not physically striking a head or needle against an ink ribbon to make a mark. Inkjet, Laser and thermal printers fall under this category of printers.

The ***non impact printers*** commonly used are :

Inkjet Printer : This printer is generally used in homes. It is a non impact printer and it does not touch the paper while printing. Instead, the print head sprays ink onto paper forming an image or text.

For business letters, a injet printer is the most common as it is used for its high quality printing. The common speed on an ink jet printer is about 3 ppm and 300 dpi (dots per inch).

Laser Printer : Among the different types of printers, laser printers remain the fastest, and the most efficient way to reproduce text and images, offering high resolution and quick speeds at a relatively low cost. A laser printer uses an electrostatic process similar to a photocopier. The laser charges the areas of a photosensitive image drum. Toner particles are then attracted to these areas of the drum. By rolling this drum against a piece of paper, the printer creates an image on a page.In the final stage, heat is used to fuse the toner to the paper.



Figure 2.13 : Inkjet printer and Laser Printer

Plotter : A plotter is a hard copy output device which is used to draw high-resolution charts, graphs, maps, and other line based diagrams. It is a pen based output device, attached to a computer for making vector graphics, that is, images created by a series of many straight lines. A plotter is a very slow and expensive output device. It is mainly used for Computer Aided Design(CAD) and Computer Aided Manufacturing applications. Plotters are classified as Drum plotters and Flatbed plotters.



Figure 2.14 : Plotter

Audio Output : Voice response systems enable the computer to talk to its users. It consists of an audio-response device that produces the audio output. Voice response systems are of *two* types : *voice reproduction system* and *speech synthesiser.*

Voice reproduction system produce an audio output by selecting the appropriate response from a predefined set of responses. These responses may be in the form of speech (words or phrases spoken by human beings), musical sounds, alarms, or other sounds. Text is converted into spoken words by speech access systems.



CHECK YOUR PROGRESS

Q5. State true or false

a) One of the most common input device is keyboard.

b) Plotters are devices used for outputting pictures, drawings.

c) Digitizers are used as output devices.

d) In case where graphical user interfaces are common mouse should not be used.

e) Speech recognition devices are input devices.

Q6. Answer the followings :

a) What are electronic card readers?

b) Compare and contrast the Laser and Dot Matrix Printers.

c) What is an OMR device?

d) On what basis are the printers classified?

2.8 COMPUTER BUS

Bus is an electrical pathway through which the processor communicates with the internal and external devices that are assembled with the computer. Buses resposible to carry memory address as well as data to the control location in memory. Computer system contains a number of different buses that provide pathways between components at various levels of the computer system hierarchy. The name of the bus is determined by the type of the signals it carries and the operations it performs. A bus that connects major computer components (CPU, memory, I/O) is called a *system bus*.

The interaction between the CPU and memory and CPU and I/O takes place very frequently. To enable this interaction, some type of connectivity is needed between the units of a computer system. This connectivity channel is known as *bus*. A key characteristic of a bus is that it

is a shared transmission medium. Physically, a bus is set of wires, which carries a group of bits in parallel and has an associated control scheme. The bus width is defined as the number of parallel lines in the bus. Every computer has three types of buses for interconnecting the CPU and memory and CPU and I/O. The functions performed by bus are as follows :

- Bus transfers data between the computer subsystem. It sends instructions and commands to and from the processors.
- It connects all internal computer components to the main memory and the CPU.
- It logically connects various peripheral devices which can easily communicate with the CPU.
- It has clock speed which is measured in MHz.

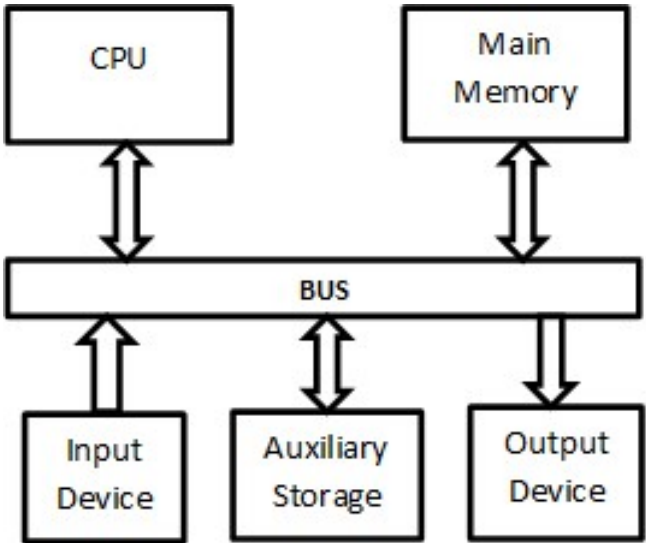


Figure 2.15: Computer Bus

Data Bus : The data bus is used to transfer data between the CPU and memory and CPU and I/O. The one that you hear most about is the data bus. The bus width of the data bus is an important parameter that affects the overall speed of a computer system. This is because each wire of a bus can transfer one bit at a time. Hence, a 8-bit bus can move 8 bits data time, a 16 bit bus can transfer two bytes, and 32 bit bus can transfer four bytes at a time. This is similar to a multi-lane highway. The wider the highway, the more traffic can flow simultaneously. Similarly the wider the bus enables more bits of data to travel, simultaneously resulting in faster

exchange of data. The data bus is also referred to as a *memory bus* which is often called *system bus.*

Address Bus : Every location in the memory has a unique address. The address of a location does not change, but the data stored in it can change. In order to retrieve some data from memory, it is necessary to specify the address of the location where the data is stored. The address bus is used to carry the address of a memory location whenever data is to be transferred to or from memory.

Again, a computer system normally has multiple I/O devices like disk, tape, network etc simultaneously connected to it. Each I/O device has a unique identifier associated to it. The address bus is used to carry the address of the I/O device to be accessed by the CPU.

Control Bus : In addition to sending address and exchanging data with the memory, the CPU also needs to send control signals to the memory to specify whether the data is to be read from or written to the specified address location. The control bus carries such signals, which are in the form of READ/WRITE commands. A typical bus structue is shown in the figure 3.16.

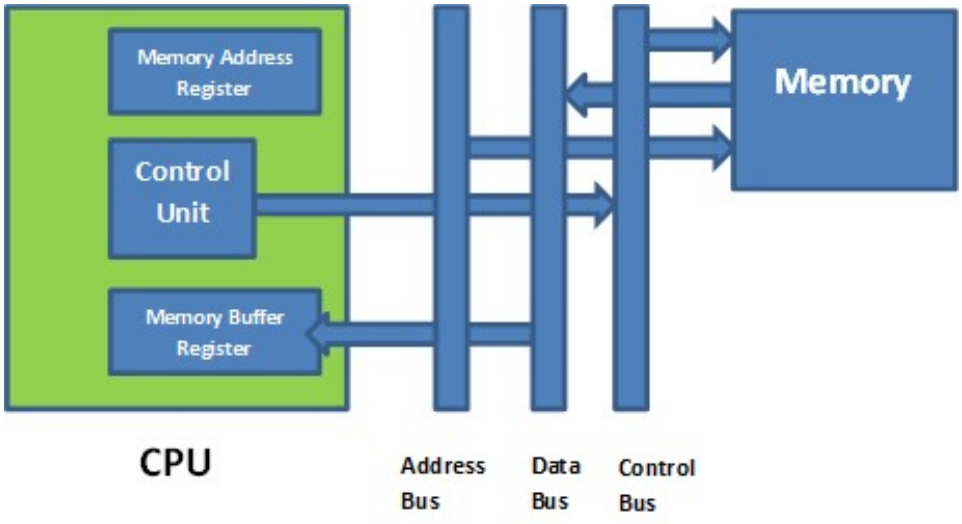


Figure 2.16: Different types of Bus



ACTIVITY

- Find out the capacity and structure of the memory types used in a computer system.
- Prepare a list of all old and new input and output devices with illustrations.



2.9 LET US SUM UP

- Data inside a computer is stored in the form of 0s and 1s. These 0 and 1s are called bit. 4 bit constitute a nibble and 8 bit constitute a byte.
- The ***memory*** usually refers to a form of semiconductor storage i.e. primary memory and ***storage*** refers to mass storag i.e. secondary memory.
- Random access memories are the semiconductor memory. They are volatile in nature. RAMs are of two types - dynamic RAM and static RAM.
- ROMs are also the semiconductor memory. The instructions in ROMs are permanent whether the power is on or off. PROM, EPROM, EEPROM are the types of ROMs.
- Cache memory is a special type of RAM which is the faster memory used in a computer system in between the CPU and main memory.
- Secondary memory also called secondary storage stores information and programs permanently. In personal computer, secondary memory typically consists of hard disk or any removable media, if present, such as a CD or DVD etc.
- Magnetic tapes are the magnetic-coated plastic tapes, which are used for storing data and audio or video files.
- Input device is an electromechanical device that accepts data or information from the user in user readable form, and translates them into computer readable form.

- An output device is an electromechanical device, which produces the output of processing by the computer to the user. Computer produces the output in a machine readable form, and the output device converts it to a human readable form.
- Bus is an electrical pathway through which the processor communicates with the internal and external devices that are assembled with the computer. Basically ,the buses are of three types : data bus, address bus and control bus.



2.10 FURTHER READING

- Ram, B. (2000). *Computer Fundamentals: Architecture and Organization*. New Age International.
- ITL Education Solutions Limited, & Sargunar, J. (2011). *Introduction to Computer Science*. Pearson Education India.
- Morris, M. M. (1987). Digital Logic and Computer Design.
- Sinha, P. K., & Sinha, P. (2010). *Computer Fundamentals* (Vol. 4). BPB publications.



2.11 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : i) True, ii) True, iii) False, iv) True, v)True, vi)True, vii) True

Ans. to Q. No. 2 : Random Access Memory is known as the main memory of the computer system.

Ans. to Q. No. 3 :

a)	**Volatile Memory**		**Non-volatile memory**
i)	Power is required all the time	i)	Power is not required once data is stored
ii)	Temporary storage	ii)	Permanent storage
iii)	Normally faster than non-volatile types	iii)	Slower than volatile memories.

b)	**Static memory**	**Dynamic memory**
i)	Loses its signal	i) Doesnot lose signal strength
ii)	Periodic refreshing of memory is needed	ii) No refreshing
iii)	Expensive	iii) Cheap
iv)	Used as Cache	iv) Used as main memory

Ans. to Q. No. 4 : The similarity in ROM and PROM is that both write operations which can be performed only once and whatever has been written cannot be changed.

Ans. to Q. No. 5 : a) True, b) True, c) False, d) False, e) True

Ans. to Q. No. 6 : a) Card readers are devices that allow direct data input into a computer system. The electronic-card reader is connected to a computer system and reads the data encoded on an electronic card and transfers it to the computer system for further processing.

b)	**Laser Printer**	**Dot Matrix Printer**
i)	High cost	i) Low cost
ii)	Forms character using laser beam	ii) Use Dot Matrix to print character
iii)	Non-impact printer	iii) Impact printer
iiv)	High quality output	iv) Output is of low quality
v)	Noiseless operation	v) Noisy
vi)	Cannot produce multiple copies of a single print	vi) Can produce multiple copies in a single print with the help of carbon paper.

c) An input device (a scanner) is capable of recognizing a pre-specified type of mark made by pencil or pen. Any input data that is of a choice or selection nature can be recorded for OMR input.

d) Printers can be classified into categories based on : printing technology, printing speed, printing quality. Printing technology : Printers can be divided into two types - *impact printers* and *non impact printers*.



2.12 MODEL QUESTIONS

Q1. Differentiate between a bit and a byte.

Q2. What is computer memory? List the two different categories of memory.

Q3. What is a RAM? What are its two types? Differentiate between them.

Q4. What is a ROM? Why is it called so? List the differenet types of ROMs and their key features.

Q5. What is a cache memory? How is it different from a primary memory?

Q6. Differentiate between

a) PROM and EPROM.

b) Static RAM and Dynamic RAM

Q7. Why are input and output devices necessary for a computer system?

Q8. What are pointing devices? Discuss some of the commonly used pointing devices.

Q9. Explain the different types of printers. Make a clear difference among them in terms of speed, cost, and method of operation.

Q10. What is a bus? List out the main functions performed by buses in a computer system.

Q11. Write short note :

a) RAM b) Light pen

c) MICR d) Scanner

e) Computer Bus

****

UNIT 3: COMPUTER SECURITY

UNIT STRUCTURE

3.1 Learning Objectives

3.2 Introduction

3.3 Computer Security

3.4 Malware

3.5 Computer Viruses

3.6 Worms

3.7 Other Threats

- 3.7.1 Trojan Horses
- 3.7.2 Spyware
- 3.7.3 Adware
- 3.7.4 Scareware

3.8 Counter Measures

- 3.8.1 Antivirus
- 3.8.2 Deception Techniques
- 3.8.3 Working of Antivirus

3.9 Let Us Sum Up

3.10 Further Reading

3.11 Answers to Check Your Progress

3.12 Model Questions

3.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn about computer security and its goals
- describe different threats related to computer security
- describe the classification of the threats
- describe measures of threats

3.2 INTRODUCTION

The advancement of computer networking and information technology has been explosive in recent times. This technology has been propagated around the world rapidly and has affected every aspect of day-to-day life. Most computer facilities continue to protect their physical machine better than their data, but in reality the value of the data is greater than the value of the hardware. Throughout this chapter, the discussion of computer security emphasizes the problem of protecting information from unauthorized disclosure.

3.3 COMPUTER SECURITY

The state of being free from danger or threat is called security. So computer security means protection of the computer systems from theft or damage to its hardware or to its software or to the information it stores, as well as from disruption or misdirection of the services it provides. More precisely we can explain with the CIA triad of information security, which maintains these three security goals:

- **Confidentiality:** It means protecting the information from disclosure to unauthorized parties.
- **Integrity:** It means keeping the data pure and trustworthy by protecting system data from intentional or accidental changes
- **Availability:** It refers to ensuring that authorized parties are able to access the information when needed.

Let us describe a day-to-day example to understand this CIA triad. Suppose I own a bank account and I am not willing to share my ATM card number, PIN and bank statements. So, me and my bank is protecting the information, this is confidentiality. Suppose, one day I make a online transaction to a 3rd party for rupees 1000.00, but in the meantime somehow, someone tampered the information and changed the amount to 10000.00. I don't want to have such tampering with my confidential data. That is integrity. I want my bank account to be accessed and manipulated only by me not by others, my bank should protect my account from unauthorized attempt of access. This is an example of availability.

3.4 MALWARE

If we can ensure these goals for our system then we can say our system is protected and any threat to these goals is a threat to computer security. The threats to computer security can be in the form of unauthorized access or deception or disruption to the services or data. Generally these violations to security are caused by some malicious software. A software is potentially malicious if it can be used to harm either the computer on which it is hosted or another computer. Software may also be considered malicious if it is designed to install itself on a computer without the permission of the owner of that computer, particularly if it does so in a way that may compromise the security of the computer. Malicious may be loosely interpreted. A piece of software may be considered malicious even though it may have been launched with the intention of providing an arguable benefit. These types of software are referred as malware (= malicious + software) in general.

Malware can be classified as

- **Transient -** They need a host to exist (Viruses, Trapdoors)
- **Resident -** They can exist on their own (Worms)

Most malicious software takes advantage of the vulnerability-exploit cycle in software development if it is not managed properly. Vulnerability is a system susceptibility or flaw, and vulnerability-exploit cycle is the continuous process of identifying, classifying, remediating, and mitigating vulnerabilities as they are discovered.

3.5 COMPUTER VIRUSES

A computer virus is an executable program that replicates itself or attached to a file where it infects the computer when the file is executed. Sometimes it may also replace an existing program. There are many techniques used by virus to spread or replicate or attach to a file, but the self replicating nature is the main criterion that distinguish a virus from other software. And not all viruses show harmful activities on system.

But why is such malicious software is termed with the biological term "Virus"? This is so because most computer viruses are parasitic like biological virus i.e.: they work by attaching themselves to a carrier object (e.g.: a file). And when this carrier object is transmitted to other systems the virus is also transmitted and it is activated when the host program is executed. After activation the computer virus like its biological one search for another host and gets attached to it.

Working of a Computer Virus

A virus does not appear as an object in itself. A virus always resides hidden in some useful object and it may perform the main parts of a virus code which are the replication routine and the payload routine. A virus is not a virus without the replication routine, but the pay load routine for a virus is optional.

- **Virus replication:**

  The replication routine contains a code that locates suitable objects to attach the virus to and to copy the virus to these elements. There are different techniques for this purpose.

  Virus code can be executed when the object is executed. Viruses that infect program files may attach the virus code at the start or at the beginning of the program and patch the entry point so that when the program is run the virus code is executed first. The virus usually transfers control to the original program when it has finished its tasks. This ensures that the original program works properly and the virus avoids detection.

- **Payload Execution:**

  Payload routine of a virus is optional, some of the viruses may not have the payload routine which give them the efficiency to replicate more quickly. Payload part of the virus code generally contains the malicious activity such as slowing down the PC, deleting system files, making short cuts of every folder, swiping out the memory etc. The payload routine usually contains some activation criteria that must be met before the payload activates. This gives the virus some time to spread without

detection. The activation criteria may be any data that can be found in computer like a particular date after infection or a lapse of time period or execution of certain application etc.

From the above discussion one thing is clear: a virus always need a host to spread and it cannot spread on its own. It needs some activation criterion to be fulfilled. Then only it can be executed or it can take over the computer resources. But there are some other types of malicious software commonly mistaken as virus but unlike the viruses. They can survive without a host. These malware are called worms.

3.6 WORMS

A worm is more independent than a virus. A pure worm works by itself as an independent object. The main feature that separates worms from viruses is that worms are self replicating. A worm can initiate network communication by itself and send copies to other computers. So there is no need for activation criteria or host process. As worms can spread rapidly across a network infecting every PC so they are the most notorious of malwares. There are some malicious software that resemble both viruses and worms.

One common example of this kind is mass mailing worms. They are attached to documents or other object like a virus, but uses email clients to mass mail themselves in a worm like way. This type of worm needs some kind of activation as for example, in mass mail worms can only activated when a receiver opens an attachment. It makes the worm slow but as it initiate the network by its own they are much faster transmitted than viruses. Worms and viruses are the most common wide-spread malicious software. There are some other programs also which work may not be so vicious like the virus or worm, but still they are malicious.



CHECK YOUR PROGRESS

Q1. What are the three security goals?

Q2. What is computer virus?

Q3. Differentiate between worm and virus.

3.7 OTHER THREATS

In this section we will briefly introduce malicious software other than virus and worm which may not exploit or damage the system like virus or worms do, but may be a potential threat in point of view of security.

3.7.1 Trojan Horses

The term trojan horse comes from the tale of the Trojan horse from the Trojan war described in Greek mythological epic. In the Trojan War, the Greeks were unable to enter the city of Troy despite 10 years long siege and finally they constructed a large wooden horse and hid some soldiers in it. The citizens of Troy thought it was just a mere wooden horse and they let it in their city and in the night the soldiers inside the horse opened the city gates, which eventually became the downfall of their city. The wooden horse is coined as Trojan horse in the epic. The Trojan horses in computer security also serve the same purpose.

Trojan horses are applications which appear to be useful, legitimate applications, but run malicious, hidden code. The application may appear to perform tasks, such as scanning for spyware, but actually steal personal data from your system. Executing Trojans may result in changing desktop wallpaper, creating popups, or even deleting files. One of their most popular uses is creating backdoors to allow malicious users to gain access to files and information. Unlike viruses or worms, Trojans cannot infect other systems.

3.7.2 Spyware

Spyware are the software that collects user sensitive information like browsing and searching habit, password, social interactions without your knowledge and uses such information to gain profit. Information collection by spyware is done using technique like key logging, browser activity monitoring. Spyware is meant to make money by using personal information so it does not exploit the PC like other malware do. In fact, many spyware may be running in our PC without our knowledge.

3.7.3 Adware

Adware are the software that supports advertising. In general it is the entire program that downloads or displays unwanted banner advertisements in the software being used. Adware can be also used to collect user personal data and based on these data the adware shows up with classified advertisement. This type of software doesn't harm the system but collecting information and running advertisement without user knowledge is a threat.

3.7.4 Scareware

Scareware is a new type of attack, where a user is tricked into downloading a software that appears to be a useful one. After installing, it shows that the machine is infected and it can clean all the infection if you purchase the full license version of the software. These scareware applications are malware that hold the PC hostage until people don't pay the ransom. In most of cases, these scareware can't be uninstalled also.

3.8 COUNTER MEASURES

"Prevention is better than cure". To detect and eliminate security threat is always tougher than to prevent it. We can say a standalone system is more secure than any other. But it is not ideal scenario that a PC will

never expose to security threats. A threat to computer security may be in many forms and hence we have to deploy different technique also to encounter them. But in this unit we will mainly focus on counter techniques for malware.

3.8.1 Antivirus

Malware uses different strategies for deception and replication. Hence, for detection and elimination of different malware involved, different techniques and tools are used which is impossible to be done manually. A special kind of software is developed for this purpose commonly known as antivirus. So, antivirus software is a program or set of programs designed for searching, preventing, detecting and eliminating malware. The detection techniques of antiviruses are generally based on malware deception strategies. So let's go through a brief overview of different deceptions strategies used by the malware.

3.8.2 Deception Techniques

Different malicious software uses different strategies to spread or replicate them and exploit the system. Deception helps malware to stay in the system and replicate for a longer time by disguising as a part of the host program. Malware employ different kinds of deception to hide them such as–

- **Overwriting:** In this technique, the virus overwrites files with their own copy. It's a primitive technique but it is simple and easy to implement. This type of viruses can be disinfected from system simply by deleting the infected files.
- **Companion Infection:** In this approach the virus disguises itself as a companion of .exe file by naming it with same base name as the targeted program, but use a .COM extension instead of .EXE, because Windows gives priority to a file with the .COM extension over a file with the .EXE extension.
- **Appending:** In this method, a jump (JMP) or goto instruction is encoded at the start of the of the host program that will point at

the end of the original host. And the starting of virus code is laid there.

- **Pre-appending:** In this strategy the virus code has been inserted at the front of host programs. This is a simple efficient method.
- **Cavity filling:** Here the virus tries to install itself in the empty space of the host while not damaging the actual program itself. An advantage of this technique is that the virus does not increase the length of the program but it also minimizes a number of possible hosts. Cavity viruses are rare.
- **Compressing:** Generally after infection, size of the affected files increase, but by using compression techniques in this method size of the affected files has reduced so that change in file become negligible.
- **Encryption:** In this scenario the virus consists of a constant decryptor or set of decryptor, followed by the encrypted virus code, after decryption of the encrypted code with different decryptor yield different virus. This type of malware is called oligomorphic virus.
- **Boot Sector Execution:** Due to the executable nature of master boot record (MBR) and partition boot sector (PBS), if any executable code is present in these sections, it will execute when the machine is booted and hence bypass all the security measures.
- **Macros:** Macros are generally targeted for Microsoft word or similar products and damage only such documents. They are embedded in such application and while activation triggered some other application or insert unreadable character in the documents.
- **Malicious mobile code (MMC):** Mobile codes are lightweight programs that are downloaded from a server and executed with minimal or no user intervention on your local machine. JavaScript scripts, VB Script, and ActiveX controls are some of examples

of mobile code that we encounter while browsing. These code are used to monitor browsing activities, getting unauthorized access to local file system, showing pop ups, adding unwanted browser add-ons etc. Spyware, adware, Trojan are such malware that uses MMC.

3.7.3 Working of Antivirus

Antivirus software scans all the file system and search for virus definition in there and tries to disinfect it by removing the malware code or deleting the infected file. Virus definition of a malware is signature by which the malware can be identified. The signature is determined based on the strategy adopt by the malware to replicate themselves. These definitions are provided by the antivirus company. Antivirus program try to match the files with these definition. The basic searching methods used by the antivirus can be broadly classified into five categories. These are:

1. **Signature based detection:** Once a virus is detected, it is analyzed precisely and a unique sequence of bytes extracted from the virus code. This string is the signature of that virus. The antivirus engine scans the binary code of files to find these strings. If it encounters with a known pattern, it alerts detection of the matching virus. Some of commonly known signature based techniques are:

- **String Scanning method:** Searches for sequence of bytes (strings) that are unique to a specific virus but can't be found in other programs. In this process, a known virus is analyzed precisely and a unique sequence of bytes extracted from the virus code. For example a virus Die.448 has a unique bit string like B440 B9E8 0133 D2CD 2172 1126 8955 15B4 40B9 0500 BA5A 01CD in its code by which we can detect it.
- **Wildcards method:** Including a wild card character like ? or * allows skipping bytes or byte ranges of code in the signature code and try to match next which will enable the antivirus engine

to detect slightly modified virus. For example: presence of a wildcard character * in the malware code means that scanner will try to match the next byte in the file.

- **Mismatches method:** This method allows a negligible number of bytes in a signature string to be of random values, regardless of their position. For example, virus X has signature string of 2A 80 4D 01 F6 and following strings contains the same signature values in different places and also mismatched by 2.

  11 2A 32 01 17 80 4D 24

  0A D9 1E 17 80 F6 24 AA

  But using mismatched method we can still say that these two strings contains the virus X.
- **Generic Detection method:** This technique uses one common generic string as a signature string to detect several or all known variants of a family of viruses. When a virus has lots of its variant then the variants are analyzed to extract a common string that can identify the whole genre. This type of technique often uses the wildcard and mismatched technique to determine a common unique string.
- **Bookmarks method:** Bookmark method can ensure more reliable detection of viruses while it is used along with string scanning by negating the false positive. The byte distance between the start of the virus code and start the signature string vary in different viruses. So it can be used to bookmark particular viruses when string based scanning is used.
- **Smart Scanning:** Smart scanning optimizes the signature based detection. Some new variants of viruses try to hide their signature by inserting junk code or instruction in their code which have no effect in execution. Smart scanning skip junk instructions, such as NOPs (No Operation) in the host file to obtain the actual virus body.
- **Skeleton Detection:** The scanner parses the statements of a file line by line and drops all nonessential statements like

statements which are not part of the virus or any blank gaps before actual procedure start and try to find essential virus code for execution of the virus which is called skeleton of the virus. An antivirus named Kaspersky uses this particular method very efficiently to detect macro viruses.

2. **Heuristic Based Detection:** This type of detection helps in detecting new unknown or altered version of malware, but it produces false positive often. So is used with along with signature based scanner. As the name suggest this detection mainly focus on heuristics (experimental trial and error methods) like analysis of file structure and the code organization of the virus or CPU emulation of virus code. Some common heuristic based detection methods are:

- **Heuristics Analysis:** It is an expert based analysis that determines the susceptibility of a malware code by running it on virtual runtime environment and analyzing system towards particular risk using various decision rules or weighing methods. Multi Criteria Analysis (MCA) is one of the means of weighing.
- **Virus specific detection:** There are cases when the standard algorithm of the virus scanner cannot detect a particular malware. In such cases, a new detection code must be introduced to implement a virus specific detection algorithm to detect only specific malware. This method includes-

➢ **Filtering:** A virus only effects a particular type of known objects. So viruses can be classified according to its infection type. For example, executable viruses infect only programs such as .EXE and .COM, macro viruses only attack to files or that can perform macro statements. So whenever a particular set of files are scanned we have to check only the specific virus signature that is relevant to those files. Such filtering minimizes the searching time.

➢ **Static Decryptor Detection:** Some viruses encrypt their body to bypass string matching so they have a decryptor in the virus

code. Detection of this decryptor code also result in efferent detection but it also produces false negative and false positive if not used with specific types of viruses which used that particular decryption. So this method is used after filtering of encrypted virus like oligomorphic virus.

- **XRay scanning:** This method tries to find the weakness in encryption algorithm of viruses. It is also virus specific and can be only used with encrypted viruses and give more suitable result than static decryptor detection as its tries to find the encryption strategy.

3. **Behavioral Based Detection:** When a malware attach to its host or execute its code some particular behavioral pattern or dynamic signature they exhibit is not common to other programs and therefore by observing such signature we may detect them. But as these signatures are dynamic we need a continuous process for monitoring such abnormal behavior. A malware may produce dynamic signature like

- Opening an executable file, with both read and write permission.
- Reading the portion of the file header containing the executable's start address
- Writing the same portion of the file header.
- Seeking to the end of the file.
- Appending to the file.

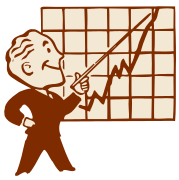
This detection method monitors a running program's behavior in real time, watching for suspicious activity like the dynamic signature mentioned above. If any such activities are detected, it report to the antivirus engine that as a response prevents such operations from being executed or terminate the program. This detection method can only work when malware perform some malicious activity during runtime.

4. **Sandbox Detection or Emulation Detection:** This type of detection is also based on behavioral traits of the malware. Here the malware code under analysis is executed in an emulated

environment or sandbox and the antivirus try to detect it during runtime based on behavioral traits. But unlike the Behavioral Based detection malware code, it is run in a virtual environment and not in real system and hence, minimizes the risk of being affected.

5. **Data mining techniques:** It is the recent technology used for detecting malware which uses data mining techniques like analyzing very huge sets of infected data and try to find out particular behavioral pattern or try to find out a common signature strings from a large number of different variants of malware and using this signature try to detect a known malware or to classify a code for malware.

   After detection, antivirus program try to remove the malware code from host program. If it is not possible to remove the viral code and the host program is not so significant than, the antivirus delete the infected file itself. On the other hand, if the infected file is important from system point of view then it pushes it to quarantine.



CHECK YOUR PROGRESS

Q4. What are Trojan horses?

Q5. Differentiate between spyware and adware.

Q6. Define antivirus.



3.9 LET US SUM UP

- The basic goal of computer security is Confidentiality, Availability and Integrity.
- Violation of these goals leads to a security treats.
- Malware is a term for all malicious programs, such as viruses, worms, Trojan horses, spyware, adware and scareware which are designed to infect and perform malicious activity on computers.

- Antivirus software is designed to protect system against malware. It is available as a standalone product and is also included in most security software packages.
- They use different static and dynamic method to identify malware and neutralize them.



3.10 FURTHER READING

- Stallings, W. (2006). Cryptography and network security: principles and practices. Pearson Education India.
- Canavan, J. E. (2001). Fundamentals of network security. Artech House.



3.11 ANSWERS TO CHECK YOUR PROGRESS

Answer 1: The three security goals are

- **Confidentiality:** It means protecting the information from disclosure to unauthorized parties.
- **Integrity:** It means keeping the data pure and trustworthy by protecting system data from intentional or accidental changes
- **Availability:** It refers to ensuring that authorized parties are able to access the information when needed.

Answer 2: A computer virus is an executable program that **replicates itself** or attached to a file where it infects the computer when the file is executed.

Answer 3: A worm is more independent than a virus. A pure worm works by itself as an independent object. The main feature that separates worms from viruses is that worms are self replicating. A worm can initiate network communication by itself and send copies to other computers. So there is no need for activation criteria or host process. As worms can spread rapidly across a network infecting every PC so they are the most notorious of malwares.

Answer 4: Trojan horses are applications which appear to be useful, legitimate applications, but run malicious, hidden code. The application may appear to perform tasks, such as scanning for spyware, but actually steal personal data from your system. Executing Trojans may result in changing desktop wallpaper, creating popups, or even deleting files.

Answer 5: Spyware are the software that collects user sensitive information like browsing and searching habit, password, social interactions without your knowledge and uses such information to gain profit. Adware are the software that supports advertising. In general it is the entire program that downloads or displays unwanted banner advertisements in the software being used.

Answer 6: Antivirus software is a program or set of programs designed for searching, preventing, detecting and eliminating malware.



3.12 MODEL QUESTIONS

Q1. Define computer security?

Q2. What are the goals of computer security?

Q3. Give an example of computer virus. Explain its working principle.

Q4. What is the difference between computer virus and worms?

Q5. Why is scareware considered as malware?

Q6. Name any antivirus and explain how it works.

Q7. Briefly describe different detection mechanism used by antivirus software.

Q8. Why deception is needed for malware? Explain various deception strategy used by malwares.

Q9. Is deception strategy of malware and detection method of antivirus related to each other? Explain.

Q10. List out some of the common malware we encounter and list out whether they are viruses, worms, Trojan or any other malicious software.

****

UNIT 4: CYBER THREATS AND LEGAL FRAMEWORK

UNIT STRUCTURE

4.1 Learning Objectives

4.2 Introduction

4.3 Cyber Crime

4.4 Hacking

4.5 Digital Forgery

4.6 Identity Theft

4.7 Cyber Stalking

4.8 Cyber Terrorism

4.9 Cyber Defamation

4.10 Software Piracy

4.11 Information Technology Act, 2000

4.12 Let Us Sum Up

4.13 Further Reading

4.14 Answers to Check Your Progress

4.15 Model Questions

4.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- learn what constitutes cyber crime
- describe hacking, digital forgery, cyber stalking
- define identity theft, cyber terrorism and defamation
- learn about different types of software piracy
- describe different offences under IT Act 2000

4.2 INTRODUCTION

In the previous unit you have been introduced to the concept of computer security. We have learned about different types of computer virus and worms. Different threats like malware, spyware, adware and Trojan

horse were explored. Counter measures to deal with these threats like antivirus software were also explored.

In the present unit you will be introduced to the concepts of cyber crime and how cyber crimes are committed against individuals, institutions and state. Cyber crimes like hacking, digital forgery, cyber stalking, and cyber terrorism are some of the cyber crimes introduced in this unit. The IT Act 2000 which criminalizes cyber crime is also discussed in this unit.

You will be made familiar with the different cyber crimes and laws against cyber crime in this unit, and then in the next unit you will be made familiar with the basics of computer hardware configuration. Different hardware configuration of computers, motherboards and processors will be covered in the next unit.

4.3 CYBER CRIME

Computers have become an essential part of our life. And along with the widespread growth of computers, internet use is also expanding. Although the use of internet and technology has made the life of people easier, it has also given rise to a new type of crime - cyber crime. The term cyberspace is used to represent the electronic medium of computer networks, in which online communication takes place. This term not only represents the internet but also includes computers, computer networks, the internet, data, software etc. Cyber crime can be defined as any crime that is performed using a computer and network either as a tool or as a target. Cyber crimes are often referred to as computer crimes.

According to a definition in the book ‘‘Cyber Crime and the Victimization of women, Laws, Rights and Regulations’’ Cyber crime can also be defined as "Offences that are committed against individuals or groups of individuals with a criminal motive to intentionally harm the reputation of the victim or cause physical or mental harm, or loss, to the victim directly or indirectly, using modern telecommunication networks such as Internet (Chat rooms, emails, notice boards and groups) and mobile phones (SMS/MMS)".

Some of the common cyber crimes are mentioned below :

- computer hacking
- digital forgery
- cyber stalking
- identity theft
- identity fraud
- cyber terrorism
- cyber defamation
- software piracy
- cyber pornography
- cyber bullying
- unwarranted cyber surveillance
- cyber warfare

We will look at some of these cyber crimes in detail in the next few sections.

Cyber crimes are generally divided into two categories depending on the use or activity of the computer. In the first category, the computer is the target of the cyber crime. Some examples of this category of cyber crime include hacking, virus attacks and denial of service attacks. In the second category, the computer is the tool used to perform the cyber crime. Some examples of this category include cyber frauds, cyber stalking and cyber terrorism.

Cyber crimes can also be divided into different categories depending on the target of the crime. The cyber crime performed can be committed against an individual, an institution or against a state. An example of cyber crime committed against an individual can be cyber stalking. Similarly, an example of cyber crime committed against an institution can be hacking their websites. An example of cyber crime committed against a state is cyber terrorism or cyber warfare.

4.4 HACKING

Hacking is one of the most common cyber crimes. It can be defined as an illegal intrusion into a computer system or network. This is a type of

crime where an individual or institution's computer is broken into so that personal or sensitive information can be accessed. The act that is committed towards breaking into a computer system or network is known as hacking.

The person who performs the act of hacking is known as a hacker. The hacker tries to find the vulnerabilities in a person's or in the institutions computer system or passwords to exploit the weakness and gain access into the system. The hackers are usually highly skilled computer professionals. Depending on the intent of the hackers, the hackers are classified into different groups.

The first category of hackers are ethical hackers. Ethical hackers are persons who hack into the computer system with an aim to identify the weaknesses present in the system and not for any personal gain. Because of this they are also called white hat hackers. Mostly, ethical hackers hack into computer systems on a request from the organization itself to test vulnerabilities present in the system.

The second category of hackers are crackers. Crackers are persons who hack into the computer system with an aim to gain unauthorized access to the system or network for personal gain. They aim to steal sensitive and important personal data, or corporate data or bank information. Both types of hackers use a variety of software to enter a person's computer. Most of the victims are sometimes not even aware that they had been hacked.

4.5 DIGITAL FORGERY

Digital forgery is the digitization of the process of forgery. Forgery in traditional systems is the crime of falsely and fraudulently making or altering a document and even possessing such a forged document. Digital forgery is the crime of falsely and fraudulently making or altering a documents and pictures in the digital domain. Examples of digital forgery include counterfeit currency notes, forgery of postage and revenue stamps, forgery of mark sheets etc. All of these acts of forgery are done by using computers and high quality scanners and printers.

There are different types of digital forgery. The most common types of digital forgery are mentioned below :

- **Document Forgery:** This type of forgery occurs when a legal document is forged or replicated by another person using digital means. This also includes forgery of mark sheets and revenue stamps.
- **Counterfeit Money Forgery:** This type of forgery occurs when the legal currency of a nation is forged or replicated by using digital means.
- **Digital Signature Forgery:** This type of forgery occurs when the signature of an individual is forged or replicated by another person using digital means.
- **Art Forgery:** This type of forgery occurs when a piece of art is replicated and sold as the original art piece. Sometimes the whole painting is forged and sometimes only the signature of a famous painter is forged into the painting with an intention to get more money.

4.6 IDENTITY THEFT

Identity theft is one of the oldest crimes in human history. Identity theft can be defined as stealing personal information of a person and impersonating as that person. The terms identity theft and identity fraud are often used interchangeably. But they refer to different crimes. In identity theft, a criminal impersonates another person who actually exists. In identity fraud, a criminal creates a fictional person and assumes this fictional person's identity. The criminal in identity fraud creates a whole new person, with both personal and financial details.

Identity theft is a much more common crime than identity fraud. Most of the information stolen is in identity theft is financial in nature regarding bank account or credit card details. But according to the victim's need different personal or financial information may be stolen. For example information like name, address, passport number, PAN number, bank account details, bank passwords, and debit or credit card details.

While most of the identity thefts are done for financial gain, there are many identity thefts done with other intentions too. According to a non-profit

Identity Theft Resource Center, identity theft can be categorized into five types depending on the intent. These categories are mentioned below:

- **Criminal Identity Theft:** In this type of identity theft, the criminal impersonates as another person as he/she is held accountable for a crime. The criminal identifies himself as another person to the state authority by using simple documents such as fake identity cards. This is usually done to get the police or court authorities off their back. The victim's of this identity theft may never come to know of these crimes done using their false impersonations.
- **Financial Identity Theft:** This is the most common type of identity theft. In this type of identity theft the criminal uses another person's identity to obtain money, loans or any other financial services using the victim's identity. These types of crimes are usually found out only when the debts mount or when it is time to pay off the loan installments.
- **Identity Cloning:** In this type of identity theft, the criminal impersonates another person to conceal their own identity. The criminals are mostly persons hiding from debt collectors or who want to became someone else for personal reasons.
- **Medical Identity Theft:** This is a new type of identity theft where the criminals steal the identity of the victim's for medical benefits. The criminals pose as another person to get medical care like medical insurances and high cost drugs.
- **Child Identity Theft:** This is another type of identity theft where the criminals target minor children for their personal gain. The criminals steal the social security numbers of children and use is information for their own benefit.



CHECK YOUR PROGRESS

Q1. Define cyber crime?

Q2. Differentiate between identity fraud and identity theft.

Q3. Who are the crackers?

4.7 CYBER STALKING

Stalking can be defined as an act of harassment targeting the victim. Stalking includes various acts such as following the victim, making harassing phone calls, vandalizing victim's property, leaving written messages or objects etc. Cyber stalking can be defined as the act of stalking done by using the means of computer systems or internet. The terms cyber stalking and cyber bullying are often used interchangeably. But there is a difference between these two terms, even though in essence both are crimes. The term cyber bullying is generally used when the parties involved are not adults and includes teenagers or children.

In most of the cases of cyber stalking, the victim is harassed by the criminal by sending repeated storm of online messages and emails. The criminals often harass the victims by posting messages on websites or discussion groups. The criminals often try to track every moment of the victim's daily activity using the help of social media. Cyber stalkers use social media sites like Facebook, Twitter, MySpace to keep track of the victim's activities and location. In most of the cyber stalking cases, the stalkers are known to the victims and they use internet to stalk the victims. Also, in many cases, if the criminals find that online stalking does not have their desired effect, they then resort to both online and offline stalking. This crime is punishable by law in India and in the first conviction for cyber stalking the person was sent to jail for three months.

4.8 CYBER TERRORISM

Cyber terrorism can be defined as an act of terrorism that is committed using computer systems and networks. Spreading terrorism through the means of internet has become an attractive option for modern terrorists. This is due to the fact that the terrorists prefer using the internet and other advanced technology in comparison to traditional methods because these machines can be accessed remotely. Another term that frequency comes when we speak of cyber terrorism is cyber warfare. Cyber warfare

can be defined as the action by a nation or a state to penetrate another nation's computers or networks for the purpose of causing damage.

In cyber terrorism, the terrorists use the internet to conduct violent acts that may cause loss of life or bodily harm. The terrorists not only target human property and life but also try to cause deliberate failure of functioning of government services by targeting communication networks. Government services like electric and solar power plants, air and rail traffic systems, telecommunication systems, medical services, police and military services are targeted mostly to paralyze the victim country. Large scale disruptions of these services are caused by using tools like computer viruses, worms and other malicious software. Most of the government websites are brought down by performing denial-of-service (DOS) attacks. In 1998, cyber terrorism attack was done by ethnic Tamil guerrillas by disrupting the Sri Lankan embassies using DOS attacks. The Sri Lankan embassies were attacked by sending around 800 e-mails a day for over a two week period.

4.9 CYBER DEFAMATION

Cyber defamation can be defined as defamation done using digital media usually through the internet. When a person publishes defamatory matter about someone on a website or sends e-mails containing defamatory information to all of that person's friends, it is termed as cyber defamation. The defamation can be done using words spoken orally or in written form or by signs or similar visible representations.

In most cases of cyber defamation when it is done with no serious grudge, it can be stopped by a cease and desist letter and by removing the messages from the digital media. But, if it is done with serious intent then the law can be used to punish the criminal. Cyber defamation not only harms the individual victim but also his family, the individual's organization and sometimes even the state or country of the victim. This depends on who the victim is and on what position the individual is currently working on. In such cases, the target for cyber defamation is often chosen not for personal grudge but for damaging the image of the organization or the country of the individual. The harm that is caused in defamation by using digital media is

widespread and often irreparable as the information is already out over the whole internet. Some of the digital media that is used to perform cyber defamation are mentioned below :

- world wide web
- social networking sites
- email and messaging
- discussion groups
- bulletin boards

In India, there are two types of liability for defamation. The first category involves cases in which the liability is of the primary publishers of the defamatory content material. This category includes persons who has written the defamatory content material and then published this material in the cyber space. The second category involves cases in which the liability is of the internet service providers or bulletin board operators. This category includes persons who had authorized publication of the defamatory materials.

4.10 SOFTWARE PIRACY

Software piracy can be defined as the unauthorized copying or illegal distribution of copyrighted materials. Software piracy causes loss of revenue for the makers which in turn lads to the increased prices of software for the licensed consumers. Most of the software has an end user license agreement with the consumers. Violation of this agreement leads to software piracy.

Software piracy has been categorized into six categories. Let us look at each of these six software piracies.

- **Softlifting:** Softlifting or softloading is one of the most common types of software piracy. Softlifting is defined as sharing a program or software with someone who is not authorized to use that software in accordance with the licensing agreement. Most often a single licensed copy of the software is brought and but then it is loaded into multiple computer systems. Softloading is quite common in college campuses and businesses.

- **Internet piracy:** With the increasing number of internet users every year the number of persons committing internet piracy has grown tremendously. Internet piracy occurs when software is downloaded from the internet. There are a lot of websites where software is available for free unlimited download without following any licensing agreements.
- **Software Counterfeiting:** Software counterfeiting is the illegal duplication, distribution and sale of copyrighted software material. In software counterfeiting it is common to make duplicate copies of the programs, the documentation and also the licensed agreements. Microsoft products are one of the most common products to be counterfeited.
- **Hard Disk Loading:** Hark disk loading software piracy occurs when new computers are sold with unauthorized software. This is done by hardware dealers to make their products more attractive to the buyers.
- **Renting:** Renting involves the process of renting software to another person temporarily who does not have the authorized permission or license to use the software.
- **Original Equipment Manufacturer (OEM) Unbundling:** In this type of software piracy, sellers sell standalone software which are originally meant to be included with some other licensed product.

4.11 INFORMATION TECHNOLOGY ACT, 2000

The Information Technology (IT) Act, 2000 is an Act of the Indian Parliament enacted with the prime objective to create an enabling environment for commercial use of information technology. The IT Act specifies the acts which have been made punishable by law regarding cyber crime and electronic commerce in India. The IT Act has amended various sections of the Indian Penal Code (IPC) of 1860, the Indian Evidence Act of 1872, the Bankers Book of Evidence Act of 1891 and the Reverse Bank of India Act of 1934 to include cyber crimes done using new technologies. A major amendment of the IT Act was made in the year 2008 to include penalties for

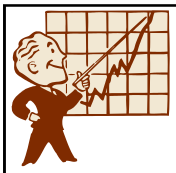
offensive messages, child pornography, cyber terrorism and voyeurism. In 2013, another amendment was made to include the offence of cyber stalking. Let us look at the various offenses that have been made punishable under the IT Act.

Some of the cyber crimes under the IT Act are mentioned below :

- Tampering with Computer source documents - Sec.65
- Hacking with Computer systems, Data alteration - Sec.66
- Sending any message that is offensive - Sec 66 A
- Publishing obscene information in electronic form - Sec.67
- Un-authorised access to protected system Sec.70
- Breach of Confidentiality and Privacy - Sec.72
- Publishing false digital signature certificates - Sec.73

Some of the cyber crimes under IPC are mentioned below :

- Cyber stalking - Sec 354D IPC
- Web-Hijacking - Sec. 383 IPC
- Bogus websites, cyber frauds - Sec 420 IPC
- Email spoofing - Sec 463 IPC
- Forgery - Sec 463 IPC
- Cyber defamation - Sec 499 IPC
- E-Mail Abuse - Sec.500 IPC
- Sending threatening messages by email - Sec 503 IPC



CHECK YOUR PROGRESS

Q4. What are the different types of software piracy?

Q5. Define cyber stalking.

Q6. What are the different types of medium used for cyber defamation?

Q7. What is the IT Act, 2000?



4.12 LET US SUM UP

- Cyber crime can be defined as any crime that is performed using a computer and network either as a tool or as a target.
- Hacking can be defined as an illegal intrusion into a computer system or network. This is a type of crime where an individual or institution's computer is broken into so that personal or sensitive information can be accessed. The person who performs the act of hacking is known as a hacker.
- Ethical hackers are persons who hack into the computer system with an aim to identify the weaknesses present in the system and not for any personal gain. Crackers are persons who hack into the computer system with an aim to gain unauthorized access to the system or network for personal gain.
- Digital forgery is the digitization of the process of forgery. Forgery in traditional systems is the crime of falsely and fraudulently making or altering a document and even possessing such a forged document. Digital forgery is the crime of falsely and fraudulently making or altering a documents and pictures in the digital domain.
- Identity theft can be defined as stealing personal information of a person and impersonating as that person. In identity theft, a criminal impersonates another person who actually exists. In identity fraud, a criminal creates a fictional person and assumes this fictional person's identity.
- Cyber stalking can be defined as the act of stalking done by using the means of computer systems or internet. The term cyber bullying is generally used when the parties involved are not adults and includes teenagers or children.
- Cyber terrorism can be defined as an act of terrorism that is committed using computer systems and networks. Cyber warfare can be defined as the action by a nation or a state to penetrate another nation's computers or networks for the purpose of causing damage.

- Cyber defamation can be defined as defamation done using digital media usually through the internet. When a person publishes defamatory matter about someone on a website or sends e-mails containing defamatory information to all of that person's friends, it is termed as cyber defamation.
- Software piracy can be defined as the unauthorized copying or illegal distribution of copyrighted materials.
- The Information Technology (IT) Act, 2000 is an Act of the Indian Parliament enacted with the prime objective to create an enabling environment for commercial use of information technology. The IT Act specifies the acts which have been made punishable by law regarding cyber crime and electronic commerce in India.
- A major amendment of the IT Act was made in the year 2008 to include penalties for offensive messages, child pornography, cyber terrorism and voyeurism.



4.13 FURTHER READING

- Chander, H. (2012). Cyber laws and IT protection. PHI Learning Pvt. Ltd..
- Singh, K. K. (2011). Information security and cyber law. kk singh.
- Godbole, N., & Belapure, S. (2011). Cyber Security: Understanding Cyber Crimes, Computer Forensic and Legal Perspective.



4.14 ANSWERS TO CHECK YOUR PROGRESS

Answer to 1: Cyber crime can be defined as any crime that is performed using a computer and network either as a tool or as a target.

Answer to 2: In identity theft, a criminal impersonates another person who actually exists. In identity fraud, a criminal creates a fictional person and assumes this fictional person's identity. The criminal in identity

fraud creates a whole new person, with both personal and financial details whereas in identity theft, the criminal impersonates an already existing person.

Answer to 3: Crackers are persons who hack into the computer system with an aim to gain unauthorized access to the system or network for personal gain.

Answer to 4: There are six types of software piracy. They are - softloading, internet piracy, renting, software counterfeiting, hard disk loading and OEM unbundling.

Answer to 5: Cyber stalking can be defined as the act of stalking done by using the means of computer systems or internet. In most of the cases of cyber stalking, the victim is harassed by the criminal by sending repeated storm of online messages and emails. The criminals often harass the victims by posting messages on websites or discussion groups.

Answer to 6: The different types of media used for cyber defamation are world wide web, social networking sites, discussion groups, bulletin boards and using emails and messages.

Answer to 7: The Information Technology (IT) Act, 2000 was enacted with the prime objective to create an enabling environment for commercial use of information technology. The IT Act specifies the acts which have been made punishable by law regarding cyber crime in India.



4.15 MODEL QUESTIONS

Q1. What are the different types of cyber crime?

Q2. Define digital forgery. What are the different types of digital forgery?

Q3. Differentiate between cyber trolling and cyber bullying.

Q4. What is cyber warfare? Give an example.

Q5. Differentiate between identity fraud and identity theft?

Q6. Describe the different types of identity theft?

Q7. Describe the different categories of cyber defamation.

Q8. Define hacking. How are the ethical hackers different from crackers?

Q9. Define cyber stalking. How is the crime of cyber stalking performed? Whch section of IT Act deals with cyber stalking?

Q10. Describe the laws related to cyber crime in India?

****

UNIT 5 : INTRODUCTION TO NUMBER SYSTEMS

UNIT STRUCTURE

5.1 Learning Objectives

5.2 Introduction

5.3 Number System

- 5.3.1 Decimal Number System
- 5.3.2 Binary Number System
- 5.3.3 Octal Number System
- 5.3.4 Hexadecimal Number System

5.4 Number System Conversion

- 5.4.1 Binary to Decimal Conversion
- 5.4.2 Decimal to Binary Conversion
- 5.4.3 Octal to Decimal Conversion
- 5.4.4 Decimal to Octal Conversion
- 5.4.5 Hexadecimal to decimal Conversion
- 5.4.6 Decimal to Hexadecimal Conversion

5.5 Let Us Sum Up

5.6 Further Reading

5.7 Answers to Check Your Progress

5.8 Model Questions

5.1 LEARNING OBJECTIVES

After going through this unit, you will be able to :

- define and describe number systems like decimal, binary, octal, and hexadecimal.
- convert from one number system to another number system.

1.2 INTRODUCTION

Computer system uses different number systems to represent data. In this unit, we will be able to understand how numbers are represented in different number systems.

In this unit, we will learn about decimal, binary, octal and hexadecimal number systems. In addition to this, we will also discuss the ways for converting from one number system to one another number system. Once, we have the knowledge of number systems, then in the next units, we will discuss about binary arithmetic and data representation methods.

5.3 NUMBER SYSTEM

Number system is a fundamental concept used in micro computer system. They are of different types and can be represented by digit symbols. The knowledge of binary, octal and hexadecimal number system is essential to understand the operation of a computer. This unit deals with all these number systems. In this unit we will discuss about all the representation of number system and their conversion from one number system to another number system.

We are familier with the decimal number system which is used in our day-to-day work. Ten digits are used in decimal number system. To represent these decimal digits, ten separate symbols 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9 are used. But a digital computer stores, understands and manipulates information composed of only zeros and ones. So, each decimal digits, letters, symbols etc. written by the programmer (an user) are converted to binary codes in the form of 0's and 1's within the computer. The number system is divided into different categories according to the base (or radix) of the system as binary, octal and hexadecimal. If a number system of base r is a system, then the system have r distinct symbols for r digits. The knowledge of the number system is essential to understand the operation of a computer.

5.3.1 Decimal Number System

Decimal number system have ten digits represented by 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. So, the base or radix of such a system is 10.

In this system, the successive position to the left of the decimal point represent units, tens, hundreds, thousands etc. For example, if we consider a decimal number 1257, then the digit representations are :

1	2	5	7
↓	↓	↓	↓
thousands positions	hundreds position	tens position	units position

The weight of each digit of a number depends on its relative position within the number.

Example 5.1 : Find the weight of each digit in 6472.

The weight of each digit of the decimal no. 6472

$6472 = 6000 + 400 + 70 + 2 = 6 \times 10^3 + 4 \times 10^2 + 7 \times 10^1 + 2 \times 10^0$

The weight of digits from right hand side are :

Weight of 1st digit = 2×10^0

Weight of 2nd digit = 7×10^1

Weight of 3rd digit = 4×10^2

Weight of 4th digit = 6×10^3

The above expressions can be written in general forms as the weight of n^{th} digit of the number from the right hand side :

$= n^{th}$ digit $\times 10^{n-1}$

$= n^{th}$ digit $\times$ (base)$^{n-1}$

The number system in which the weight of each digit depends on its relative position within the number is called positional number system. The above form of general expression is true only for positional number system.



Decimal Number System uses 10 digits from 0 to 9.

5.3.2 Binary Number System

Only two digits 0 and 1 are used to represent the binary number system. So the base or radix is two (2). The digits 0 and 1 are called bits (Binary Digits). In this number system the value of the digit will be two times greater than its predecessor. Thus, the value of the places are :

$\leftarrow 32 \leftarrow 16 \leftarrow 8 \leftarrow 4 \leftarrow 2 \leftarrow 1$

The weight of each binary bit depends on its relative position within the number. It is explained by the following example--

Example 1.2 : Find the weight of binary number 10110.

The weight of bits of the binary number 10110 is :

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

$= 16 + 0 + 4 + 2 + 0 = 22$ (decimal number)

The weight of each bit of a binary no. depends on its relative pointer within the no. and explained from right hand side as :

Weight of 1st bit = 1st bit X 2^0

Weight of 2nd bit = 2nd bit X 2^1

.....

.....

and so on.

The weight of the n^{th} bit of the number from right hand side

$= n^{th}$ bit $\times 2^{n-1}$

$= n^{th}$ bit $\times$ (Base)$^{n-1}$



A ***Binary Number System*** uses only digit 0 and 1

It is seen that this rule for a binary number is same as that for a decimal number system. The above rule holds good for any other positioned number system. The weight of a digit in any positioned number system depends on its relative positon within the number and the base of the number system.

Table 5.1 shows the binary equivalent numbers for decimal digits.

Table 5.1 : Binary equivalent of decimal numbers

Decimal Number	Equivalent Binary Number
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001

Binary Fractions : A binary fractions can be represented by a series of 1s and 0s to the right of a binary point. The weight of digit positions to the right of the binary point are given by 2^{-1}, 2^{-2}, 2^{-3} and so on.

Example 5.3 : Show the representation of binary fraction 0.1101.

Solution : The binary representation of 0.1101 is :

$$0.1101 = 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4}$$
$$= 1 \times 0.5 + 1 \times 0.25 + 0 \times 0.125 + 1 \times 0.0625$$
$$= 0.8125$$

So, $(0.1101)_2 = (0.8125)_{10}$

5.3.3 Octal Number System

A commonly used positional number system is the Octal Number System. This system has eight (8) digit representation as 0,1,2,3,4,5,6 and 7. The base or radix of this system is 8. The values increase from left to right as 1,8,64,512, 4096 etc. The decimal value 8 is represented in octal as 10,9 as 11,10 as 12 and so on. As $8=2^3$, an octal number is represented by a group of three binary bits. For example, 3 is represented as 011, 4 as 100 etc.

Table 5.2 The octal number and their binary representations.

Decimal Number	Octal Number	Binary Coded Octal No.
0	0	000
1	1	001
.	.	.
.	.	.
7	7	111
8	10	100 000
15	17	001 111

5.3.4 Hexadecimal Number System

The hexadecimal number system is now extensively used in computer industry. Its base (or radix) is 16, and the digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. The hexadecimal numbers are used to represent binary numbers incase of conversion and compactness.



Octal Number System uses 8 digits from 0 to 7.

As $16 = 2^4$, hexadecimal number is represented by a group of four binary bits. For example, 5 is represented by 0101. Table 1.3 shows the binary equivalent of a decimal number and its hexadecimal representation.

Table 5.3 : Hexadecimal number and their Binary representation

Decimal No.	Hexadecimal No.	Binary coded Hex. No
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
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

5.4 NUMBER SYSTEM CONVERSION

As the computer uses different number systems, there is a process of converting generally used decimal number systems to other number systems and vice-versa.

5.4.1 Binary to Decimal Conversion

To convert a binary number to its decimal equivalent we use the following expression.The weight of the n^{th} bit of the number from right hand side can be represented as-

$= n^{th}$ bit $\times 2^{n-1}$

First we mark the bit position and then we give the weight of each bit of the number depending on its position. The sum of the weight of all bits gives us the equivalent number.

Example 5.4 : Convert binary $(100101)_2$ to its decimal equivalent.

Solution : $(100101)_2 = 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$= 32 + 0 + 0 + 4 + 0 + 1$

$= 37$

So, $(100101)_2 = (37)_{10}$

Mixed number contain both integer and fractional parts and can convert to its decimal equivalent is as follows :

Example 5.5 : Convert $(11011.101)_2$ to its equivalent decimal number.

Solutaion :

$(11011.101)_2 = (1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0) + (1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3})$

$= (16 + 8 + 0 + 1) + (0.5 + 0 + 0.125) = 27.625$

So, $(11011.101)_2 = (27.625)_{10}$

5.4.2 Decimal to Binary Conversion

There are different methods used to convert decimal number to binary numbers. The most common method is to repeatedly divide the decimal number by 2, then the remainder 0's and 1's obtained after division, is read in reverse order to obtain the binary equivalent of the decimal number.

Example 5.6 : Convert $(75)_{10}$ to its binary equivalent.

Solution :

		Remainder
2 | 75		
2 | 37	LSB	1
2 | 18		1
2 | 9		0
2 | 4		1
2 | 2		0
2 | 1	MSB	0
0		1

Read in reverse order

So, $(75)_{10} = (1001011)_2$

Example 5.7 : Convert decimal fraction $(25.625)_{10}$ to its equivalent binary number.

Solution : 2|25 Remainder MSB 0.625

2 |12	1	↑	× 2
2 |6	0		1.250
2 |3	0		× 2
2 |1	1		0.500 ↓
0	1		× 2
			1.000

$(25)_{10} = (11001)_2$

$(0.625)_{10} = (0.101)_2$

So, $(25.625)_{10} = (11001.101)_2$

5.4.3 Octal to Decimal Conversion

The method of converting octal numbers to decimal numbers is simple. The decimal equivalent of an octal number is the sum of the numbers multiplied by their corresponding weights.

Example 1.8 : Find decimal equivalent of octal number $(153)_8$

Solution : $1 \times 8^2 + 5 \times 8^1 + 3 \times 8^0 = 64 + 40 + 3 = 107$

So, $(153)_8 = (107)_{10}$

The fractional part can be converted by multiplying it by the negative powers of 8 as shown in the following example.

Example 1.9 : Find decimal equivalent of octal number $(123.21)_8$

Solution : $(1X8^2 + 2X8^1 + 3X8^0) + (2 \times 8^{-1} + 1 \times 8^{-2})$

$= (64 + 16 + 3) + (0.25 + 0.0156) = 83.2656$

So, $(123.21)_8 = (83.2656)_{10}$

5.4.4 Decimal to Octal Conversion

The procedure for conversion of decimal numbers to octal numbers is exactly similar to the conversion of decimal number to binary numbers except replacing 2 by 8.

Example 1.10 : Find the octal equivalent of decimal $(4121)_{10}$

Solution :

8 |4121		
8 |515	1	↑ read from MSB
8 |64	3	↑ to LSB
8 |8	0	↑
8 |1	0	↑
0	1	↑

So, $(4121)_{10}=(10031)_8$

The fractional part is multiplied by 8 to get a carry and a fraction as shown in the following example.

Example 5.11 : Find the octal equivalent of $(.123)_{10}$

Solution : Octal equivalent of fractional part of the decimal number is :

8 × 0.123 = 0.984	0	↓ read from LSB
8 × 0.984 = 7.872	7	↓ to MSB
8 × 0.872 = 6.976	6	↓
8 × 0.976 = 7.808	7	↓

Read the integer to the left of the decimal point.

The calculation can be terminated after a few steps if the fractional part does not become zero.

The octal equivalent of $(0.123)_{10}= (0.0767)_8$

NOTE : The octal to binary and binary to octal conversion is very easy. Since, 8 is the third power of 2, we can convert each octal digit into its three-bit binary form and vice versa.

Example 5.12 : Convert $(567)_8$ to its binary form.

Solution :

5	6	7
↓	↓	↓
101	110	111

So, $(567)_8 = (101\ 110\ 111)_2$

Conversion from binary to octal is just opposite of the above example.

5.4.5 Hexadecimal to Decimal Conversion

The method of converting hexadecimal numbers to decimal numbers is simple. The decimal equivalent of an hexadecimal number is the sum of

the numbers multiplied by their corresponding weights.

Example 5.13 : Find the decimal equivalent of $(4A8C)_{16}$

Solution :

$$(4A8C)_{16} = (4 \times 16^3) + (10 \times 16^2) + (8 \times 19^{1)} + (12 \times 16^0)$$
$$= 16384 + 2560 + 128 + 12$$
$$=(19084)_{10}$$
$$(4A83)_{16} = (19084)_{10}$$

Example 1.14 : Find the decimal equivalent of $(53A.0B4)_{16}$

Solution :

$$(53A.0B4)16 = (5 \times 16^2) + (3 \times 16^1) + (10 \times 16^0) + (0 \times 16^{-1}) + (11 \times 16^{-2}) + (4 \times 16^{-3})$$
$$= 1280 + 48 + 10 + 0 + 0.04927 + 0.0009765$$
$$= (1338.0439)_{10}$$
$$(53A.0B4)_{16} =(1338.0439)_{10}$$

5.4.6 Decimal to Hexadecimal Conversion

The procedure for conversion from decimal no. and its fraction part to hexadecimal equivalent is exactly similar to the conversion of decimal to binary number except replacing 2 by 16.

Example 5.15 : Convert decimal $(1234.675)_{10}$ to hexadecimal.

Solution : First, we consider $(1234)_{10}$:

Remainder

	Decimal	Hexadecimal
16 |1234	2	2
16 |77	13	D
16 |4	4	4

$(1234)_{10} = (4D2)_{16}$

Then, conversion of $(0.675)_{10}$:

	Decimal	Hexadecimal
0.675 × 16 = 10.8	10	A
0.800 × 16 = 12.8	12	C
0.800 × 16 = 12.8	12	C
0.800 × 16 = 12.8	12	C

$(0.675)_{10}= (0.ACC)_{16}$

Hence $(1234.675)_{10}= (4D2.ACC)_{16}$

If the decimal number is very large, it is tedious to convert the number to binary directly. So it is always advisable to convert the number into hexadecimal first, and then convert the hexadecimal to binary.



CHECK YOUR PROGRESS

Q.1. What is the largest numbar that can be represented using 8 bits?

.....

Q.2. What is the weight of 1 in $(10000)_2$.

.....

Q.3. Convert the following:

a) $(565.25)_{10}$ to its equivalent binary number.

b) $(256.24)_8$ to decimal equivalent.

c) $(A3B.BB)_{16}$ to decimal equivalent.

d) $(10010.110)_2$ to decimal equivalent.

e) $(3964.63)_{10}$ to octal equivalent.



5.5 LET US SUM UP

- We have learnt four different number systems used in digital systems in this unit.
- Conversion of one number system to another number system can be done.
- In our day-to-day life, we use the decimal number system. In this system, base is 10 and we use 10 digits from 0 to 9.
- The binary number system has two digits and so its base is 2.
- The octal number system has 8 digits from 0 to 7 and the base of the system is 8.
- The hexadecinal number system has 16 digits from 0 to 9 and A to F and the base of the system is 16.



5.6 FURTHER READING

- *Ram, B. (2000), Computer Fundamentals Architecture and Organization*, New Age International
- *Mano, M.M. (2017), Digital Logic and Computer Design*, Pearson Education India
- *Sinha, P. K. and Sinha P. (2010), Computer Fundamentals*, BPB Publication.
- *Talukdar, P. (2010), Digital Techniques*, N.L. Publications.



5.7 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : $(1111\ 1111)_2$

Ans. to Q. No. 2 : Weight is $2^4 = 16$

Ans. to Q. No. 3 : a) $(1000110101.01)_2$ b) $(174.44)_{10}$ c) $(2620.8)_{10}$
d) $(18.75)_{10}$ e) $(7574.50)_8$



5.8 MODEL QUESTIONS

Q.1.What is binary number system?

Q.2.What is meant by 'base' or 'radix' of a number system?

Q.3.Convert the following to its binary equivalent:

(i) $(67)_8$ (ii) $(A8D)_{16}$ (iii) $(81B6.F)_{16}$
(iv) $(64.3)_{10}$ (v) $(765.45)_8$ (vi) $(1725.23)_8$

Q.4. Convert the following to its decimal equivalent:

(i) $(11011)_2$ (ii) $(1111.11)_2$ (iii) $(45)_8$
(iv) $(671.23)_8$ (v) $(ABCD)_{16}$ (vi) $(12AF.E4)_{16}$

Q.5.Explain the process of converting a decimal number to binary.

Q.6.How can a binary number be converted into octal and hexadecimal numbers?

****

UNIT 6 : BINARY ARITHMETIC

UNIT STRUCTURE

6.1 Learning Objectives

6.2 Introduction

6.3 Complement of Numbers

6.3.1 (r–1)'s Complement

6.3.2 r's Complement

6.4 Binary Arithmetic

6.4.1 Addition

6.4.2 Subtraction

6.4.3 Multiplication

6.4.4 Division

6.5 Let Us Sum Up

6.6 Further Reading

6.7 Answers to Check Your Progress

6.8 Model Questions

6.1 LEARNING OBJECTIVES

After going through this unit you will be able to:

- describe r's and (r-1)'s complement
- describe binary addition and subtraction
- describe binary multiplication and division.

6.2 INTRODUCTION

In the previous unit, we have been introduced to the different number systems Decimal, Octal, Binary and Hexadecimal. We have also learnt how to convert from one number system to another number system in the previous unit. In this unit, we will learn about the complement systems. We will learn about R and (R-1)'s complement systems. We will also learn to perform binary arithmetic functions like addition, subtraction, multiplication and division in this unit. In the next few units are will learn more about data representation and code conversion methods.

6.3 COMPLEMENT OF NUMBERS



The 7's and 15's complement of a number is found by subtracting each digit of the number from 7 and 15 respectively.

Complements are used in digital computers for simplifying the substraction operation and for logical manipulation.The complement of a binary number is obtained by inverting all of the bits.

For example, the complement of 10011 is 01100 and 00101 is 11010 etc. The complement again depends on the base of the number.

There are two types of complements for a number of base r. These are :

- r's complement and
- (r-1)'s complement,

For example, for decimal numbers the base is 10. Therefore, complements will be 10's complement and (10–1)=9's complement. For binary numbers, the complement are 2's complement and 1's complement since base is 2.

6.3.1 (r–1)'s Complement

Given a number N in base r having n digits, the (r–1)'s complement of N is defined as $(r^n–1) – N$.

9's Complement : For decimal numbers, r=10 and r-1=9, so the 9's complement of N is $(10^n–1) – N$.

For example, with n = 4 we have $10^4 = 10000$ and $10^4–1 = 9999$. It follows the rule that the 9's complement of a decimal number is obtained by subtracting each digit from 9.



Like 8's complement and 16's complement of a number is found by adding 1 to the LSB of the 7's and 15's complement of an octal and hexadecimal number respectively.

For example, 9's complement of 49 is (99–49) = 50

9's complement of 127 is (999–127) = 872

1's Complement : For binary numbers, r = 2 and (r–1) = 1, so the 1's complement of N is $(2^n–1) – N$. Again, 2^n is represented by a binary number that consists of a 1 followed by n 0's. 2^n-1 is a binary number represented by n 1's. For example,with n = 4, we have $2^4 = (10000)_2$ and $2^4–1 = (1111)_2$. Thus the 1's complement of a binary number is obtained by subtracting each digit from 1.

However, the subtraction of a binary digit from 1 causes the bit to change from 0 to 1 or from 1 to 0. Therefore, the 1's complement of a binary number is formed by changing 1's into 0 and 0's into 1's. For example, 1's complement of 1010111 is 0101000.

6.3.2 R's Complement

The r's complement of a n-digit number N in base r is defined as $r^n - N$ for N =0 and 0 for N = 0. Comparing with the (r–1)'s complement, we note that the r's complement is obtained by adding 1 to the (r–1)'s complement.

10's Complement : The 10's complement of a decimal number is equal to the 9's complement of the number plus 1

i.e. 10's complement of decimal number = Its 9's complement +1

So, 10's complement of 49 is (99–49) + 1 = 50 + 1 = 51

10's complement of 127 is (999–127) + 1 = 872 + 1 = 873

2's Complement : It is obtained by adding 1 in the 1's complement form of the binary numbers.

i.e. 2's complement of binary number = 1's complement of that number + 1

For example, 2's complement of 1010111 is 0101000 + 1 = 0101001



CHECK YOUR PROGRESS

Q.1. Find 9's complement 10's complement of decimal numbers 44 and 182.

.....

.....

Q.2. Find 1's complement and 2's complement of binary numbers 1101001 and 0000.

.....

.....

6.4 BINARY ARITHMETIC

Like decimal arithmetic operations such as addition, subtraction we can perform arithmetic operation in binary number system too.

2.4.1 Addition

Binary addition is performed in the same manner as decimal addition. Since,

in binary system only two digit 0 and 1 are used, the addition follows the following rules.

0 + 0 = 0

0 + 1 = 1 = 1+0

1 + 1 = 0, Carry 1 to the next left column

1 + 1 +1 = 1, Carry 1 to the next column.

Carry overs are performed in the same manner as in decimal arithmetic.

Example 2.1 : Add the binary numbers

(i) 1011 and 1001

(ii) 10.011 and 1.001

Solution :

	Binary no.	**Equivalent decimal no.**
(i)	11	
	1011	11
	+1001	9
	10100	20
(ii)	10.011	2.375
	1.001	1.125
	11.100	3.500

Since the circuit in all digital systems actually can handle two numbers to performs addition, it is not necessary to consider the addition of more than two binary numbers. When more than two numbers are to be added, the first two are added first and then their sum is added to the third and so on.

The complexity may rise when we add combination of positive and negative binary numbers. In this case, the arithmatic addition is dependent on the representation of

a) Signed magnitude

b) Signed 1's complement

c) Signed 2's complement

This will be more clear if we discuss using the following example:

Example 2.2 : Add 25 and -30 in binary using 7 bit register in signed magnitude representation

a) Signed 1's complement representaion

b) Signed 2's complement representaion

Solution : Here, 25 is + 25 = 0011001 in binary system

–30 = 1011110 in binary system

To do arithmatic addition with one negative number, we have to check the magnitude of the numbers. The number having smaller magnitude is subtracted from the bigger number and the sign of bigger number is selected. To implement such a scheme in digits, hardware will require a long sequence of control decisions as well as circuits that will add, compare and subtract numbers. The better alternative of arithmatic addition with one negative number is signed 2's complement.

In signed 2's complement representation :

We get that +30 is 0 011110

–30 is 1 011110

Now, 2's complement of –30 (including sign bit) 1 100010

+25 is 0 011001

Addition

+25	0	011	001
–30	1	100	010
–05	1	111	011 (Just add the numbers)

The result for negative number will store in signed 2's complement form. So the above result in signed 2's complemnt form including sign bit is:

1 000 100 +1 = 1 000 101

Which is -05 in decimal system.

From the above example, it can be noticed that, signed 2's complement representation is simpler than signed magnitude representaion. This procedure requires only one central decision and only one circuit for adding the two numbers. But it puts additional condition that the negative numbers should be stored in signed 2's complment form in the register. This can be achieved by complementing the positive number bit by bit then incrementing the resultant by 1 to get signed 2's complement.

In signed 1's complement representation : This method is also simple. Here, we add the two numbers including the sign bit. If carry of the most significant bit or sign bit is one, then we increment the result by 1 and discard the carry over.

Addition :

+25 = 0 011 001

–30 = 1 100 001 (1's complement of -30)

–5 = 1 111 010

The result will store in 1's complement format. So, 1111 010 in 1's complement format including the sign bit is 1 000 101 which is the required result.

Example 6.3 : Add -25 and +30 using 7-bit register.

Solution :

–25 1 100 110 (1's complement of 25)

+30 0 011 110

+5 1 0 000 100



Carry bit, so add 1 to the sum and discard the carry.

This sum is now = 0 000 101 which is +5

Example 6.4 : Add –25 and –30 using 7-bit register.

Solution :

–25 1 100 110 (1's complement of 25)

–30 1 100 001 (1's complement of 30)

–55 1 1 000 111



Carry bit, so add 1 to sum and discard the carry.

Now the sum is = 1 001 000, which is –55

Since, +55 is 0 110 111

So, –55 is in 1's complement 1 001 000

The interesting feature about these representation is the representation of 0 in signed magnitude and 1's complement. There are two representations for zero :

Signed magnitude +0 –0

	0	000000	1	000000
Signed 1's complement	0	000000	1	111111

But in signed 2's complement, there is just one zero and there are no positive or negative zero.

+0 000000

–0 in 2's complement is +0 = 1 111111

1

1 0 000000

↑

discard this carry

Thus, both +0 and –0 are same in 2's complement notation. This is an *added advantage* in favour of 2's complement notation. The maximum number which can be accomodated in registers also depends on the type of representation. In general, in a 8 bit register, 1 bit is used as sign. Therefore, the rest of 7 bits are used for representing the value. The value of maximum and minumum number which can be represented are :

For signed magnitude representation $2^7 - 1$ to $-(2^7 - 1)$

= 128 –1 to – (128–1)

= 127 to – 127,

which is for signed 1's complement representation.

For signed 2's complement representation is from + 127 to –128. The –128 is represented in signed 2's complement notation as 10000000.

6.4.2 Subtraction

Though there are many other methods for performing subtraction, we will consider the method of subtraction known as complementary subtraction. This is a more efficient method of subtraction while using electronic circuits. We have to follow the following three steps to subtract binary numbers.

In 1's complement method :

1. Find the 1's complement of the number which is subtracting.
2. Add the number which is subtracting from with the complement value obtained from step 1.

3. If there is a carry of 1, add the carry with the result of addition. Else, take complement of the result again and attach a negative sign with the result.

Example 6.5 : Subtract 5 – 6 by 1's complement method.

Solution : Binary equivalent of 5 is 101

Binary equivalent of 6 is 110

Step 1 : 1's complement of 6 is 001

Step 2 : Adding 001 with 101 gives the result :

$$\begin{array}{r} 001 \\ +\ 101 \\ \hline 110 \end{array}$$

Step 3 : Since there is no carry in step 2, we take the complement again which will be 001 and after attaching negative sign, the required result will be –001 which is -1.

In 2's complement method : It is same as 1's complement method except step 3. The steps are :

Step 1 : Find the 2's complement of the number which is subtracting.

Step 2 : Add the number which is subtracting from, with the complement value obtained from step 1.

Step 3 : If there is a carry of 1, ignore it. Else, take 2's complement of the result again and attach a negative sign with the result.

Example 6.6 : Subtract 5 – 7 by 2's complement method.

Solution : Binary equivalent of 5 is 101

Binary equivalent of 7 is 111

Step 1 : The 2's complement of 7 is 000 + 1 = 001

Step 2 : Adding 001 with 101 will give result as :

$$\begin{array}{r} 001 \\ 101 \\ \hline 110 \quad \text{(No carry)} \end{array}$$

Step 3 : Since there is no carry, the 2's complement of 110 is 001 + 1 = 010 and attaching a negative sign the required result is –10 i.e.,-2.

Overflow : An overflow is said to have occured when the sum of two n digit number occupies (n+1) digits. This definition is valid for both binary as well

as decimal digits. But what is the significance of overflow for binary numbers? Well, the answer lies in the representation of numbers. Every computer employs a limit for representation of numbers eg. in our examples we are using 8 bit registers of calculating the sum. But what will happen if the sum of the two numbers can be accommodated only in 9 bits? Where are we going to store the 9th bit? The problem will be more clear by the following example. In case of a +ve no. added to a –ve no., the sum of result will always be smaller than the two numbers. An overflow always occurs when the added numbers are both +ve or both -ve.

Example 6.7 : Add the numbers 65 and 75 in 8 bit register in signed 2's complement notation.

Solution :

65	0	1000001
75	0	1001011
140	1	0001100

This is a -ve number and the 2's complement of the result is equal to -115 which is obviously a wrong result. This has occured because of overflow.

Detection of Overflow : Overflow can be detected as :

If the carry out of the MSBs of number (or, carry into the sign bit) is equal to the carry out of the sign bit, then overflow must have occured. For example

–65	1	0111111	–65	1	0111111
–15	1	1110001	–75	1	0110101
–80	1 1	0110000	–140	1 0	1110100
Carry	= 1		Carry	= 10	
Carry from MSB	= 1		Carry from MSB	= 0	
Carry from sign bit	= 1		Carry from sign bit	= 1	
Sign bit is	= 1		Sign bit is	= 0	
No overflow			Overflow		

Thus, overflow has occured, i.e. the arithmatic results so calculated have exceeded the capacity of the representation. This overflow also implies that the calculated results might be errorneous.

6.4.3 Multiplication

Multiplication in binary follows the same rules that are followed in the decimal system. The rules to be remembered are:

$0 \times 0 = 0$ $1 \times 0 = 0$

$0 \times 1 = 0$ $1 \times 1 = 1$

For example, multiplying 10101 × 11001

```
   10101
 × 11001
 -------
   10101
  10101
 10101
 -------
1000001101
```

6.4.4 Division

The process of binary division is same as the decimal division. In binary division we have two rules:

$0/1 = 0$ $1/1 = 1$

The steps for binary division are :

1. Start from the left of the divided.
2. Perform subtraction in which the divisor is subtracted from the divedend.
   a) If subtraction is possible put a 1 in the quotient and subtract the divisor from the corresponding digits of the dividend. Else put a 0 in the quotient
   b) Bring down the next digit to the right of the remainder.
3. Execute step 2 till there are no more digits left to bring down from the dividend.

Example 6.8 : Divide 1011001 by 110

Solution :

```
                 0111      (Quotient)
(Divisor)  110 | 1011001   (Dividend)
                 110
                  1010
```

110

1000

110

0101

Here 111 is the quotient and 101 is the remainder.



CHECK YOUR PROGRESS

Q.3. Add 35 and -40 in binary using 7 bit register in 2's complement representation.

.....

Q.4. Add binary no. 110011.010 and 1000.10

.....

Q.5. Subtract $(1010101)_2$ -$(1001001)_2$ by 2's complement method.

.....

Q.6. Multiply $(11001)_2$ by $(101)_2$.

.....

Q.7. Divide $(101101.101)_2$ by $(110)_2$.

.....



6.5 LET US SUM UP

- The 1's complement of a binary number is formed by changing 1's into 0 and 0's into 1's.
- The 2's complement of a binary number is obtained by adding 1 to the 1's complement of the binary number.
- Binary addition is performed in the same manner as decimal addition.
- Complementary subtraction method is considerd for binary subtraction.
- An overflow is said to have occured when the sum of two n digits number occupies (n+1) digits.



6.6 FURTHER READING

- *Ram, B. (2000), Computer Fundamentals Architecture and Organization*, New Age International
- *Mano, M.M. (2017), Digital Logic and Computer Design*, Pearson Education India
- *Sinha, P. K. and Sinha P. (2010), Computer Fundamentals*, BPB Publication.
- *Talukdar, P. (2010), Digital Techniques*, N.L. Publications.



6.7 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : 9's complement of 44 is 55 and 182 is 817
10's complement of 44 is 56 and 182 is 818

Ans. to Q. No. 2 : 1's complement of 1101001 is 0010110 and 0000 is 1111
2's complement of 1101001 is 0010111 and 0000 is 10000

Ans. to Q. No. 3 : $(000101)_2$

Ans. to Q. No. 4 : $(111011.110)_2$

Ans. to Q. No. 5: $(1100)_2$

Ans. to Q. No. 6 : $(1111101)_2$

Ans. to Q. No. 7 : $(111.10011)_2$



6.8 MODEL QUESTIONS

Q.1. Find 8's and 16's complement of the following:
(i) 67 (ii) 5672

Q.2. Find the 7's and 15's complement of the following:
(i) 643 (ii) 15AB

Q.3. Perform the following by 2's complement method:
(i) 10101 – 11011 (ii) 100011 – 1111

Q.4. Multiply 1101 to 11110.

Q.5. Divide 110011 by 110.

****

UNIT 7 : INTRODUCTION TO DATA REPRESENTATION

UNIT STRUCTURE

7.1 Learning Objectives
7.2 Introduction
7.3 Data Representation
7.4 Fixed Point Representation
7.5 Floating Point Representation
7.6 Let Us Sum Up
7.7 Further Reading
7.8 Answers to Check Your Progress
7.9 Model Questions

7.1 LEARNING OBJECTIVES

After going through this unit, you will be able to :

- identify how data is represented in computers
- describe fixed point representation
- describe floating point representation

7.2 INTRODUCTION

In the previous units, we have learnt about different number systems like binary, octal, decimal and hexadecimal. We have also learnt how to do binary arithmetic operations like addition, subtraction, multiplication and division.

In this unit, we will discuss about the concept of data representation. Fixed and floating point representation are introduced in this unit. Floating point representation will be covered in detail in unit 7 later. In the next unit we will discuss about the different types of computer codes.

7.3 DATA REPRESENTATION

Data are usually represented by using the alphabets A to Z, numbers 0 to 9 and various other symbols. This form of representation is used to formulate problem and fed to the computer. The processed output is required

in the same form. This form of representation is called *external data representation*. However, the computer can understand data only in the form of 0's and 1's. The method of data representaion in a form suitable for storing in the memory and for processing by the CPU is called the internal data representation on digital computer.

Data, in general, are of two types : Numeric and non-numeric (Character data). The numeric data deals only with numbers and arithmetic operations and non numeric data deals with characters, names, addresses etc. and non-arithmetic operations. Numeric data can be represented using fixed point or floating point representations. Let us look at these in detail.

7.4 FIXED POINT REPRESENTATION

A fixed point number in binary system uses a sign bit. A positive number has a sign bit 0 while the negative number has a sign bit 1. A negative number can be represented in one of the following ways.

- Signed magnitude representaion
- Signed 1's complement representaion
- Signed 2's complement representaion

Assume that the size of the register is 7 bit and the 8th position bit in used for error checking and correction or other purposes.

a) Signed magnitude representation

	+6	–6	
0	000110	1	000110
↑		↑	No change in the
Sign bit		Sign bit	magnitude, only the sign bit changes

b) Signed 1's complement representation

	+6	–6	
0	000110	1	111001

Here 0 and 1 are sign bits. We get 1's complement for the –ve integer is by taking complement of all the bits of +ve no. including sign bit.

c) Signed 2's complement representation

	+6	–6	
0	000101	1	111011
↑		↑	2's complement of the
Sign bit		Sign bit	positive number
			including sing bit

The signed magnitude system is easier to interpret but computer arithmetic with this system is not efficient. The circuits for handling numbers are simplified if 1's or 2's complement systems are used and as a result one of these two is almost always adopted.

Note 1 : In 1's and 2's complements, all positive integers are represented as sign magnitude system.

Note 2 : When all the bits of the computer word are used to represent the number and no bit is used for signed representation, it is called unsigned representation of the number.

7.5 FLOATING POINT REPRESENTATION

A number which has both an integer part as well as a fractional part is called real number or floating point number. A floating point number is either positive or negative. Examples of real decimal numbers are 156.65, 0.893, –235.75, –0.253 etc. Examples of binary real numbers are 101.101, 0.11101, –1011.101, –0.1010 etc.

The first part of the number is a fixed point number which is called mantissa. It can be an integer or a fraction.

The second part specifies the decimal or binary point position and is termed exponent. It is not a physical point. Therefore, whenever we are representing a point it is termed as an exponent. It is only the assumed position. For example, for decimal 0. +15.37, the typical floating point notaion is :

$51.47 = 0.5147 \times 10^2$ or 5147×10^{-2}

Now, the floating point representation of 0.5147×10^3 is :

Sign Sign

|0 .5147| |0 02|

Mantissa (fraction) Exponent

The floating point representation of 5147 X 10^{-2} is

Sign Sign

|0 5147| |0 02|

Mantissa (Integer) Exponent

Similarly, for example a floating point binary number 1011.1010 can be represented as : 1011.1010 = 0.10111010 × 2^4

This can be represented in a 16 bit register as follows

Sign Sign

|0 .10111010| |0 000100|

Mantissa (fraction) Exponent

The mantissa occupies 9 bits (1 bit for sign and 8 bits for value) and the exponent 7 bits (1 bit for sign and 6 bits for value). The binary point (.) is not physically indicated in the register, but it is only assumed (position) to be there.

In general form, the floating point numbers is expressed as :

$N = M \times R^e$

Where, M – Mantissa

R – Radix (or base)

e – Exponent

The mantissa M and exponent e are physically present in register. But the radix R and the point (decimal or binary point) are not indicated in the register. There are only assumed for computation and manipulation.

Normalized Floating point Number : Floating point numbers are often represented in normalized forms. A floating point number where mantissa does not contain zero as the most significant digit of the

number is considered to be in normaliged form. For example, 0.00038695×10^5 and 0.0589×10^{-4} are not normaliged numbers. But 0.38695×10^2 and 0.589×10^{-5} are normaliged numbers. Similarly, for binary number also, 0.0011001×2^8 and 0.0001011×2^{-5} are not non-normaliged binary numbers. But 0.11001×2^6 and 0.1011×2^{-8} are normaliged binary numbers.

A zero cannot be normaliged as all the digits in the mantissa is zero.

Arithmetic operations involved with floating point numbers are more complex. It takes larger time for execution and requires complex hardware. But floating point representaion is frequently used in scientific calculations.

Overflow and Underflow : When the result is too small to be presented by the computer, an overflow or underflow condition exists. When two floating-point numbers of the same sign are added, a carry may be generated out of high-order bit position. This is known as mantissa overflow. In case of addition or subtraction floating point numbers are aligned. The mantissa is shifted right for the alignment of a floating point number. Sometimes, the low order bits are lost in the process of alignment. This is referred as mantissa underflow. To perform the multiplication of two floating point numbers, the exponents are added. In certain cases the sum of the exponents maybe too large and it may exceed the storing capacity of the exponent field. This is called exponent overflow. In case of division the exponent of the divisior is subtracted from the exponent of the dividend. The result of subtraction may be too small to be represented. This is called exponent underflow.

Overflow or underflow resulting from a mantissa operation can be corrected by shifting the mantissa of the result and adjusting the exponent. But the exponent overflow or underflow can not be corrected and hence, an error indication has to be displayed on the computer screen.



CHECK YOUR PROGRESS

Q.1. Represent 10 by the following representation method.

a) Signed magnitude representation.

b) Signed 2's complement representation.

Q.2. Represent $(1010.1010)_2$ with floating point representation method.

Q.3. Fill in the blanks

a) When all the bits of the computer word are used to represent the number, it is called _____ of the number.

b) A _____ cannot be normalized as all the digits in the mantissa is zero.

c) In general, _____ of a number is used to represent sign bit in signed magnitude representation.

d) _____ takes longer time for execution and requires complex hardware.

e) An _____ occurs when the result is too small to be represented by the computer.



7.6 LET US SUM UP

- A fixed point numbers in binary system uses a sign bit. A positive number has a sign bit 0 while the negative number has a sign bit 1.
- For fixed point we have discussed three different methods, they are:
  - Signed magnitude representation method
  - Signed 1's complement representation method
  - Signed 2's complement representation method
- **Signed magnitude representation:** It uses one bit, usually the leftmost bit, to indicate the sign where '0' indicates the positive integer

and the '1' indicates the negative integer. The rest of the bits are used as magnitude.

- A number which has both an integer part and a fractional part is called real number or floating point number. A floating point number can be either positive or negative.
- The first part of the number is a fixed point number which is called ***Mantissa***. It can be an integer or a fraction.
- The second part specified the decimal or binary point position and is termed ***exponent***.
- A floating point number where mantissa does not contain zero as the most significant digit of the number is considered to be in normalized form.
- When two floating-point numbers of the same sign are added, a carry may be generated out of high-order bit position. This is known as mantissa overflow. In case of division the exponent of the divisor is subtracted from the exponent of the dividend. The result may be too small to be represented and is calaled exponent underflow.



7.7 FURTHER READING

- Morris, M.M. (1987). DIgital Logic and Computer Design.
- Sinha, P.K., & Sinha, P. (2010). Computer Fundamentals (Vol. 4). BPB publications.
- Ram, B. (2000). Computer Fundamentals: Architecture and Organisation. New Age International.



7.8 ANSWERS TO CHECK YOUR PROGRESS

Answer to Q1: a) 001010

b) 110110

Answer to Q2: Mantissa is 10101010, fraction is 000100 (in 16 bit representation)

Answer to Q3: a) unsigned representation

b) zero

c) leftmost bit
d) Arithmetic operation
e) overflow or underflow



7.9 MODEL QUESTIONS

Q.1. Explain the fixed point representation method with example.
Q.2. How can decimal number be represented?
Q.3. What is mantissa and exponent of decimal representation?
Q.4. How can a floating point number be normalized for computerized representation?
Q.5. What is overflow and underflow?
Q.6. What is data representation?

****

UNIT 8 : CODE CONVERSION

UNIT STRUCTURE

8.1 Learning Objectives

8.2 Introduction

8.3 Computer Code

8.4 BCD number

8.5 ASCII Code

8.6 EBCDIC

8.7 Gray Code

8.8 Let Us Sum Up

8.9 Further Reading

8.10 Answers to Check Your Progress

8.11 Model Questions

8.1 LEARNING OBJECTIVES

After going through this unit, you will be able to :

- describe different computer code systems
- describe BCD and ASCII code
- describe EBCDIC and Gray code

8.2 INTRODUCTION

In the previous units we have learned about different number systems like decimal, binary, octal and hexadecimal. We have also covered binary arithmetic operations like addition, subtraction multiplication and division. We have also discussed about the methods for representing data. Fixed point representation and floating point representation were also introduced in the previous unit.

In this unit, we will introduce the concept of computer code and code conversion. Different computer codes like BCD, ASCII, EBCDIC and gray codes are discussed in this unit. We will also learn how to convert binary to gray code and vice versa. In the next units, we will discuss about boolean algebra and logic gates.

8.3 COMPUTER CODE

A **code** is a symbol or group of symbols that represents discrete elements. Coding of characters has been standardised to enable transfer of data between computers. *Numeric data* is not the only form of data handled by a computer. We often require to process *alphanumeric data*. An alphanumeric data is a string of symbols, where a symbol may be one of the letters A, B, C, ..., Z, or one of the digits 0, 1, 2, ..., 9, or a special character, such as + – */, . () = (space for blank) etc. However, the bits 0 and 1 must represent any data internally. Hence, computers use binary coding schemes to represent data internally. In binary coding, a group of bits represent every symbol that appears in the data. The group of bits used to represent a symbol is called a *byte*. To indicate the numbers of bits in a group, sometimes a byte is referred to as “n-bit byte”, where the group contains n bits. However, the term “byte” commonly means an 8-bit byte because most modern computers use 8 bits to represent a symbol.

8.4 BCD NUMBER

In computing and electronic systems, **Binary-Coded Decimal** (**BCD**) is a way to express each of the decimal digits with a binary code. Its main advantage is that it allows easy conversion to decimal digits for printing or display and faster decimal calculations. Decimal numbers with their BCD equivalent are given in the Table 4.1 :

Table 8.1 : BCD Code

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

Unlike binary encoded numbers, BCD encoded numbers can easily be displayed by mapping each of the nibbles(4-bits) to a different character. Examples of conversion of decimal to BCD and BCD to decimal are included below:

Example 4.1 : Convert the decimal numbers 47 and 180 to BCD.

Solution : 4 = 0100 and 7 = 111

47 = 0100111

Similarly, 180=0001 1000 0000

Example 4.2 : Convert each of the BCD code 1000111 to decimal.

Solution : First we have to divide the whole BCD code by a set of 4-bits from right to left. Then from left to right we can put the corresponding decimal numbers.

0100	0111
4	7

Thus, 1000111(in BCD) = 47 (in Decimal)

BCD addition : BCD is a numeric code and can be used in arithmetic operations. Addition is the most important operation because the other three operations (subtraction, multiplication, and division) can be accomplished by the use of addition. Here is how to add two BCD numbers:

Step1 : Add the two BCD numbers, using the rules for binary addition.

Step 2 : If a 4-bit sum is equal to or less than 9, it is a valid BCD number.

Step 3 : If a 4-bit sum is greater than 9, or if a carry out of the 4-bit group is generated, it is an invalid result. Add 6(0110) to the 4-bit sum in order to skip the six invalid states and return the code to 8421. If a carry results when 6 is added, simply add the carry to the next 4-bit group.

Example 4.3 : Add the following BCD numbers :

a) 0001 + 0100

b) 10000111 + 01010011

Solution : The decimal number addition are shown for comparison.

(a)

0001	1
+0100	+4
0101	5

$\therefore$ 0001 + 0100 = 0101 Which is valid BCD no. (Value < 9)

(b)

1000	0111	87
+0101	0011	+ 53

	1101	1010	Both groups are invalid (>9) +140
	+0110	+0110	Add 6 (i.e.,0110) to both groups
0001	0100	0000	Valid BCD number which is 140 in decimal.

8.5 ASCII CODE

ASCII stands for American Standard Code for Information Interchange. It is a very well-known fact that computers can manage internally only 0s (zeros) and 1s (ones). By means of sequences of 0s and 1s the computer can express any numerical value as its binary translation, which is a very simple mathematical operation.

However, there is no such evident way to represent letters and other non-numeric characters with 0s and 1s. Therefore, in order to do that, computers use *ASCII tables*, which are tables or lists that contain all the letters in the roman alphabet plus some additional characters. In these tables each character is always represented by the same order number. For example, the ASCII code for the capital letter “A” is always represented by the order number 65, which is easily representable using 0s and 1s in binary: 65 expressed as a binary number is 1000001. ASCII has 128 character codes(from 0 to 127) and symbols represented by a 7-bit binary code.

ASCII is the common code for microcomputer equipment. The first 32 characters in the ASCII-table are unprintable control codes and are used to control peripherals such as printers. Examples of control characters are “NULL”, “line feed”, “start of text” and “escape”. The other characters are graphic symbols that can be printed or displayed and that include the letters of the alphabet (lowercase and upper case), the ten decimal digits, punctuation signs, and other commonly used symbols.

8.6 EBCDIC

EBCDIC is the abbreviation for **Extended Binary-Coded Decimal Interchange Code**. EBCDIC is an IBM code for representing characters as numbers. It uses 8 bits per character. Thus 256 charaters can be represented with 8 bits. The 9th position bit can be used for parity. The EBCDIC code is used in IBM mainframe models and other similar machines.

In EBCDIC, the first 4 bits are known as zone bits and remaining 4 bits represent digit values. Electronic circuits are available to transform characters from ASCII to EBCDIC and vice-varsa.

8.7 GRAY CODE

The gray code is an unweighted code not suited for arithmetic operations, but useful for input output devices, analog to digital converters etc. The **Gray code**, named after *Frank Gray*, is a binary numeral system where two successive values differ in only one digit. It is sometimes referred to as reflected binary, because the first eight values compare with those of the last 8 values, but in reverse order.

The Gray code was originally designed to prevent spurious output from electromechanical switches. Today, Gray codes are widely used to facilitate error correction in digital communications such as digital terrestrial television and some cable TV systems.

Table 8.2 : Gray Code

Decimal	Gray Code	Binary
0	0000	0000
1	0001	0001
2	0011	0010
3	0010	0011
4	0110	0100
5	0111	0101
6	0101	0110
7	0100	0111
8	1100	1000
9	1101	1001
10	1111	1010
11	1110	1011
12	1010	1100
13	1011	1101
14	1001	1110
15	1000	1111

Conversion from Binary to Gray code : The following rules explain how to convert from a binary number to a Gray code **:**

a) The most significant bit (left most) in the gray code is the same as the corresponding most significant bit in the binary code.

b) Going from left to right, add each pair of adjacent pair of binary code to get the next Gray code bit. Discard carries.

For example, the conversion of the binary number 1100 to gray code is as follows:

Binary code 1 1 0 0

MSB (Most significant bit) LSB (Least significant bit)

MSB of Gray code will be same as MSB of Binary code. Here, it will be 1. Now , addition of each pair of adjacent bits of Binary code:

1 + 1 = 10 = 0 , discarding the carry 1

1 + 0 = 1, no carry

0 + 0 = 0, no carry

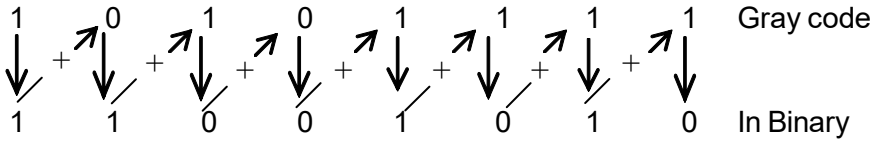
Therefore, 1100 in gray code is 1010

Conversion from Gray to Binary code : The following rules apply:

a) The most significant bit (left most) in the binary code is the same as the corresponding most significant bit in the gray code.

b) Add each binary code bit generated by Gray code bit in the next adjacent position. Discard carries.

For example, binary 10101111 in gray code will be like this :





CHECK YOUR PROGRESS

Q.1. Convert the following binary numbers to gray code

(i) 11011 (ii) 10110

Q.2. Convert the following Gray codes to Binary codes.

(i) 11011 (ii) 100111

Q.3. (i) The American Standard Code for Information Interchange is a standard _____ bits code.

(ii) EBCDIC uses _____ bits per character.

(iii) Code is a representation of _____.

(iv) In both binary and gray code, the _____ is same.

Q.4. Add the following BCD numbers :

(i) 00100011 + 00010001 (ii) 1001 + 0100



8.8 LET US SUM UP

- BCD code is a way to express decimal digits with a binary code.
- BCD, ASCII, EBCDIC,and Gray Code are the commonly used commputer codes.
- BCD is a numeric code and can be used in arithmetic operations.
- ASCII is a 7 bit binary code.
- EBCDIC uses 8-bits to represent character.
- Gray codes are used to facilitate error correction.
- ASCII stands for American Standard Code for Information Interchange.
- Gray codes area not not suited for arithmetic operations.



8.9 FURTHER READING

- *Ram, B. (2000), Computer Fundamentals Architecture and Organization*, New Age International
- *Mano, M.M. (2017), Digital Logic and Computer Design*, Pearson Education India
- *Sinha, P. K. and Sinha P. (2010), Computer Fundamentals*, BPB Publication.
- *Talukdar, P. (2010), Digital Techniques*, N.L. Publications.



8.10 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : (i) 10110 (ii) 11101

Ans. to Q. No. 2 : (i) 10010 (ii) 111010

Ans. to Q. No. 3 : (i) 7 (ii) 8 (iii) Discrete elements

(iv) MSB-bit

Ans. to Q. No. 4 : (i) Solution :

0010	0011	35	
+0001	0001	+17	
0011	0100	+52	Valid BCD no. (since both <9)

(ii) Solution :

1001			9
+0100			+4
1101		Invalid BCD number (>9)	+13
+0110		Add 6 (i.e., 0110)	
0001	0011	A valid BCD number which is 13 in decimal	



8.11 MODEL QUESTIONS

Q1. What is BCD code?

Q2. Convert the decimal numbers 15, 275 to BCD.

Q3. Add the two BCD numbers 10000111 and 01010011.

Q4. Write BCD for the following decimal number:

(i) 679 (ii) 45.96

Q.5. Give the importance of Gray codes.

Q.6. Write a short note on ASCII codes.

****

UNIT 9 : BOOLEAN ALGEBRA

UNIT STRUCTURE

9.1 Learning Objectives

9.2 Introduction

9.3 Boolean Operators

9.4 Basic Theorems and Postulates of Boolean Algebra

9.5 Let Us Sum Up

9.6 Further Reading

9.7 Answers to Check Your Progress

9.8 Model Questions

9.1 LEARNING OBJECTIVES

After going through this unit, you will be able to :

- define Boolean algebra
- explain basic concepts of Boolean Algebra
- define the basic theorems and postulates of Boolean Algebra
- define Boolean function

9.2 INTRODUCTION

In the previous units we have been introduced to concepts like binary arithmetic, data representation and code conversion. We have learnt about the methods for representing data. We have also learnt about the different types of computer codes like gray, BCD, ASCII and EBCDIC codes. The conversion process from one code to another is also discussed in previous units.

In this unit, you will be able to learn about the fundamentals of Boolean Algebra. You will also learn the basic theorems and postulates of Boolean algebra. Different concepts like the principles of Duality, De Morgan's theorem are also covered in this unit.

In the next unit, we will learn about the different types of logic gates. Conversion of logic gates is also covered in the next unit.

9.3 BOOLEAN OPERATORS

Boolean algebra may be defined with the help of three sets of components viz,

i) a set of elements,

ii) a set of operators and

iii) a number of unproved axioms or postulates.

A set of elements is any collection of objects having a common property. A set of operators are the binary operators, which are rules that are assigned to each pair of elements of the set, a unique element from the set. The axioms or postulates form the basic assumptions from which it is possible to deduce the rules, theorems and properties of Boolean algebra. A variable in Boolean algebra can take only two vales, 1 (TRUE) or 0 (FALSE). i.e. TRUE is represented by 1 and FALSE is represented by 0. Boolean algebra is used for designing and analysing digital circuits. There are three basic operations in Boolean algebra and these operations are done with the help of three operators, viz. AND, OR and NOT. (These are also called basic logic operation).

AND operation : Boolean AND operator for two variables A and B can be represented as:

A and B or A.B or AB

It results 1 or TRUE if both the operands A and B are 1 (TRUE), otherwise the result is 0 (FALSE).

OR Operation : The OR operator for the same variables can be represented as :

A OR B or A+B

The result of this operation in 0 (FALSE) if both the variables are 0 (FALSE); otherwise the result is 1 (TRUE).

NOT Operation : The NOT operation for a variable A can be represented as:

NOT $\bar{A}$ or A or $A^{/}$.

It returns the opposite value of the variable i.e. returns 0 (FALSE) if A is 1 (TRUE) and vice versa.

The results of these Boolean operations can be represented in a tabular form, which is referred to as the "***truth table***".

Table 9.1: Truth Table for AND, OR and NOT operation

A	B	(A AND B) A.B	(A OR B) A+B	NOT(A)
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Note : Boolean addition is same as the OR operation and Boolean multiplication is same as AND operation.

Truth Table : A truth table is a table which gives the output for all combination of input values. A truth table can be drawn for a particular function/operation. In the truth Table 9.1, three tables are merged into a single one.

In addition to these three basic Boolean operations, three more operators have been defined for Boolean algebra. They are: XOR (Exclusive OR), NOR (Not+OR) and NAND (Not+AND).

9.4 BASIC THEOREMS AND POSTULATES OF BOOLEAN ALGEBRA

The theorems and postulates are the most basic relationships in Boolean algebra. Six theorems and four postulates of Boolean algebra are listed in Table 9.2. The postulates are basic axioms of the algebric structure and need no proof, but the theorems must be proven with the help of the postulates. Both the postulates and theorems are listed in pairs: one is the dual of the other.

Table 9.2 Postulates and Theorems of Boolean Algebra

Postulate 2	(a) $x + 0 = z$	(b) $x.1 = x$
Postulate 5	(a) $x + x' = 1$	(b) $x. x' = 0$
Theorem 1	(a) $x + x = x$	(b) $x. x = x$
Theorem 2	(a) $x + 1 = 1$	(b) $x. 0 = 0$
Theorem 3, Involution $(x')' = x$		
Postulate 3, Commutative	(a) $x + y = y + x$	(b) $xy = yx$
Theorem 4, Associative	(a) $x + (y + z) = (x + y) + z$	

		(b) x (yz) = (xy)z
Postulates 4, Distributive	(a) x (y + z) = xy + xz	
		(b) x+yz = (x+y)(x+z)
Theorem 5, De Morgan's	(a) $(x + y)' = x' y'$	(b) $(xy)' = x' + y'$
Theorem 6, Absorption	(a) x + xy = x	(b) x(x + y) = x

Duality Principle : This is an important property of Boolean algebra. It states that every algebric expression deducible from the postulates of Boolean algebra remains valid if the operator and identity elements are interchanged. In a two-valued Boolean algebra (which is defined on a set of two elements, B = {0, 1}, with rules for the two binary operators + and•), the identity elements and the elements of the set B are same: 1 and 0. If we need the dual of an algebric expression we simply interchange OR and AND operators and replace 1's by 0's and 0's by 1's.

The proofs of the theorems with one variable are given below:

Theorem 1 (a) x + x = x

x + x = (x + x).1 by postulate	:	2 (b)
= (x + x)• $(x + x')$	:	5 (a)
= x + x x'	:	4 (b)
= x + 0	:	2 (a)
= x		

Theorem 1 (b) : x. x = x

x. x = x. x + 0 by postulate	:	2 (a)
= xx + x x'	:	5 (b)
= x $(x + x')$	:	4 (a)
= x.1	:	5 (a)
= x	:	2 (a)

If we observe carefully we see that theorem 1 (b) is the dual of theorem 1 (a) and each step of the proof in part (b) is the dual of part (a). Thus any dual theorem can be similarly derived from the proof of its corresponding pair.

Theorem 2 (a): x + z = 1

x + 1 = 1. (x + 1) by postulate	:	2 (b)

$= (x + x') (x + 1)$: 5 (a)

$= x + x'.1$: 4 (b)

$= x + x'$: 2 (b)

$= 1$: 5 (a)

Theorem 2 (b): x. 0 = 0 by duality.

Theorem 3: $(x')' = x$

We have $x. x' = 0$ from postulate 5 (b), which defines complement of x.

The complement of x' is x and is also $(x')'$. Since the complement is unique, therefore we have $(x')' = x$.

We can prove the theorems which involve two or three variables, algebraically, from the postulates and theorems which have already been proven. Let us consider the absorption theorem.

Theorem 6 (a) : $x + xy = x$

$x + xy = x.1 + xy$ by postulate 2 (b)

$= x (1 + y)$ by postulate 4 (a)

$= x (y + 1)$ by postulate 3 (a)

$= x.1$ by theorem 2 (a)

$= x$ by postulate 2 (b)

Theorem 6 (b): x (x + y) = x by duality.

The theorems of Boolean algebra can also be proved easily with the help of truth table. The truth table shown in Table 9.3 verifies the theorem 6 (b).

Table 9.3 Truth Table for verification of Theorem 6 (b).

1	**2**	**3**	**4**
x	**y**	**x + y**	**x (x + y)**
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

Here we see that, column 4 and column 1 are same, i.e. x (x + y) = x. Since the algebric proof of the De Morgan's theorem and the associative law are very long, we can show their validity with truth tables easily. Let us consider the De Morgan's theorem: $(x + y)' = x' y'$.

Now, the truth table for this is shown in Table 9.4.

Table 9.4 : Truth table for De Morgan's theorem

1	2	3	4	5	6	7
x	**y**	**x + y**	**$(x + y)'$**	**x'**	**y'**	**x' y'**
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Column 4 is equal to column 7, so, $(x + y)' = x' y'$

Complement of a Boolean Function : The complement of a function F is F'. It can be obtained from an interchange of $0'$s for $1'$s and $1'$ s for $0'$s in the value of F. Algebraically, the complement of a function may be derived through De Morgan's theorem. The De Morgan's theorem can be extended to any number of variables.

The theorem can be generalized as :

$(A + B + C + ... + G)' = A'B'C' ... G'$

$(ABC...G)' = A' + B' + C' + ... + G'$

This generalized form of De Morgan Theorem states that the complement of a function is obtained by interchanging AND and OR operators and complementing each literal. (A literal is a primed or unprimed variable). Let $F_1 = A'BC + A'B'C'$. Now F_1' i.e. complement of the function can be obtained by applying the De Morgan's theorem as follows:

$F_1' = (A'BC + A'B'C')'$

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

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

An easier procedure for deriving the complement of a function is to take the dual of the function and complement each literal. Thus, the dual of $F_1 = (A'+B+C) (A'+B'+C')$ and then complementing each literal, $F_1' = (A+B' +C') (A+B+C)$.



CHECK YOUR PROGRESS

Q.1. What is a truth table? Create a truth table for AND and OR operation?

Q.2. Define the principle of Duality.

Q.3. State and prove the De Morgan's theorem for two variables. (Using truth table)



9.5 LET US SUM UP

- Boolean addition is same as logical OR and Boolean multiplication is same as logical AND operation.
- The logical NOT operation changes logical 1 to 0 and vice versa.
- To get dual of any Boolean expression, you have to replace every 0 with 1 and every 1 with 0, and replacing every operator (+) with (.) and every (.) with (+).
- Any Boolean expression obtained by interchanging 0s and 1s and the operator (of an expression) is called the dual expression. This is the duality principle of Boolean algebra.
- De Morgan's theorem can be extended to any number of variables. It is useful in obtaining the complement of a Boolean function.



9.6 FURTHER READING

- Mano, M.M. (1982), Computer System Architecture, Englewood Cliffs, N. J. 53-54
- Mano, M.M. (2017), Digital logic and Computer Design, Pearson Education India.
- Ram, B. (2000), Computer Fundamentals Architecture and Organization, New Age International.

9.7 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : A truth table is a table, which has two sides, viz, input and output. In input side, we have all the possible combinations of input variables, i.e. for 2 variables, we have $2^2 = 4$ input combinations(4 rows), for 3, we have 8 combinations and so on. The output side gives us the result of the function or operation either 1 (TRUE) or 0 (FALSE). Output 1 indicates that the corresponding input combination (minterm) is present in the function.

Truth table for AND and OR operations

Input		Output	
A	B	A.B	A+B
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Ans. to Q. No. 2 : The principle of duality states that for any Boolean expression, another valid expression can be obtained by replacing each 0 with 1 and 1 with 0 and interchanging the binary operators (.) and (+). For any pair of expression so obtained, the original one is called the primal and the new one is called the dual expression. For example – the expression $a\,b' + cd = 1$, its dual is $(a + b')(c + d) = 0$.

Ans. to Q. No. 3 : The De Morgan's theorem for two variables A and B can be state as:

I. $(A + B)' = A'B'$

II. $(AB)' = A' + B'$

i.e. complement of an expression can be obtained by complementing each literal and interchanging the binary operators (.) and (+).

Truth Table

Proof :

A	B	A'	B'	$A'B'$	(A+B)	$(A+B)'$	AB	$(AB)'$	$A'+B'$
0	0	1	1	1	0	1	0	1	1
0	1	1	0	0	1	0	0	1	1
1	0	0	1	0	1	0	0	1	1
1	1	0	0	0	1	0	1	0	0

= =



9.8 MODEL QUESTIONS

Q.1. State and prove De Morgan's Theorem.

Q.2. What are the different ways of representing a Boolean function?

Q.3. What is Duality Principle?

Q.4. Define truth table.

Q.5. Prove that x (y + z) = xy + yz using truth tables.

Q.6. Prove that x + xy = x.

****

UNIT 10 : LOGIC GATES

UNIT STRUCTURE

10.1 Learning Objectives
10.2 Introduction
10.3 Logic Gates
  10.3.1 OR Gate
  10.3.2 AND Gate
  10.3.3 NOT Gate
  10.3.4 NAND Gate
  10.3.5 NOR Gate
  10.3.6 XOR Gate
  10.3.7 XNOR Gate
10.4 De Morgan's Theorem
10.5 Truth Table
10.6 Conversion of the Logic Gates
10.7 Let Us Sum Up
10.8 Further Reading
10.9 Answer to Check Your Progress
10.10 Model Questions

10.1 LEARNING OBJECTIVES

After going through this unit, you will be able to :

- describe different logic gates
- prepare truth table for AND, OR, NOT, NAND, NOR, XOR and XNOR gates
- obtain boolean function from a truth table
- perform NAND implementation of a Boolean function
- perform NOR implementation of a Boolean function

10.2 INTRODUCTION

In the previous units, we have learnt about number system, binary arithmetic, boolean algebra and conversion codes. Different number systems have been covered along with different types of computer codes in previous units. In addition to these, boolean operators and basic postulates

and theorems are also covered in the previous units.

In this unit, we will discuss about the different types of logic gates. The graphical symbols and truth table of the logic gates are also discussed in detail. The conversion of logic gates is also covered in this unit. In the next unit, we will discuss in detail about the floating point representation.

10.3 LOGIC GATES

A logic gate is an electronic circuit, which is used or collectively can be used to transform a boolean function or expression into a logic diagram. Logic gates have only one output and atleast two inputs except for the NOT gate, which has only one input. The output signals appears only for certain combinations of input signals. Binary information available in the input lines are manipulated by the gates. Three basic logic circuits, commonly called gates are used to make logic decision : they are OR, AND and NOT gate. Logic gates are available in the form of various IC families and are the basic building block of various circuits. Each gate has a distinct graphic symbol and its operation can be described by means of an algebric function. The input-output relationship of the binary variables for each gate can be represented in tabular form in a truth table.

Before going to discuss about the functions of the logic gates, we have to know few basic terms that are associated with the functions of gates. Logic 1 and 0, that are applied as input or may be obtained as ouput of a gate, are represented by voltage levels. Positive logic (or active high levels) means that the most positive logic voltage level (also referred to as the high level) is defined to be the logic state1. On the other hand, the most negative logic voltage level (also referred to as the low level) is defined to be the logic state 0. Negative logic (or active law levels) is just the opposite, the most positive (high) level is 0, and the most negative (low) level is a 1. For instance, if the voltage levels are (–0.1)v and (-5)v, then in a positive logic system, the (-5)v level represents a zero and the –0.1v represents a 1. Conversely, if the voltage levels are 0.1v and 5v, then in a negative logic system, the 5v levels represents a zero and the 0.1v represents a one.

The choice of positive or negative logic is made by the individual logic designer. We cannot say one is advantageous over the other. It is common to see that most logic designers and text books on logic design use positive logic.

10.3.1 OR Gate

An OR gate has two or more inputs and a single output. It is an electronic circuit and the output of an OR gate is HIGH (logic1) if atleast one of the inputs is HIGH, otherwise (i.e. if all inputs are LOW) the output is LOW (0). Figure 10.1 shows the graphic symbol used for an OR gate.

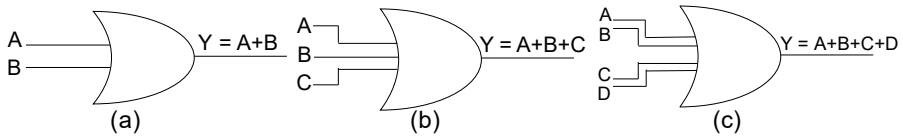


Figure 10.1 : Graphic symbols for (a) two inputs, (b) 3 inputs and (c) 4 inputs or gate

The algebric function for OR gate (2 inputs) is : F = x+y, and the truth table is :

Table 10.1 Truth table of OR gate (2inputs)

x	y	F
0	0	0
1	0	1
1	1	1

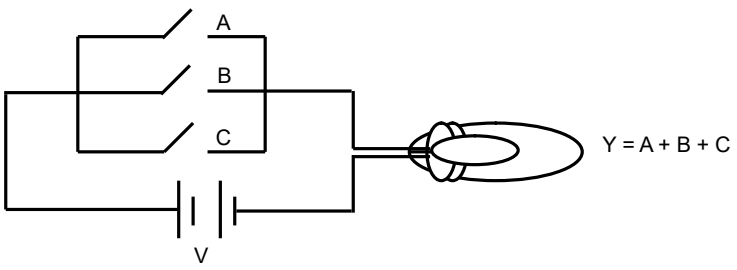


Figure 10.2 : Shows the switching circuit analogy of OR function

The circuit of an OR gate is arranged in such a way that the output is in state 1, when anyone of the inputs is in state 1 ; i.e. when input A or input B or input C is 1 (in case of 3 inputs OR gate). The circuit can be illustrated by the analogy shown in Figure 10.2. The circuit consists of a battery, a lamp and three parallel switches connected in series. Battery switches are the inputs to the lamp and the light from the lamp represents the circuit output.

Let us define an open switch as a 0 state, i.e., no light represents 0 state, and a closed switch represents state 1, i.e., a glowing lamp as a 1 state. We can list the various combinations (8 combinations) of switch states as inputs to the circuit and the resulting output states in a truth table. It is clear from the truth table that all switches must be opened (0 state) for the light to be off (output in 0 state). This type of circuit is called an OR gate.

Table 10.2 : Truth table of 3 inputs OR gate

Inputs			Outputs
A	B	C	Y = A + B + C
0	0	0	0
0	0	1	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

10.3.2 AND Gate

An AND gate also may have two or more inputs and a single output. In order to have a HIGH (1) output, all the inputs of the AND gate must be HIGH (1), otherwise the output is LOW. Figure 10.3 shows graphic symbols used for an AND gate.

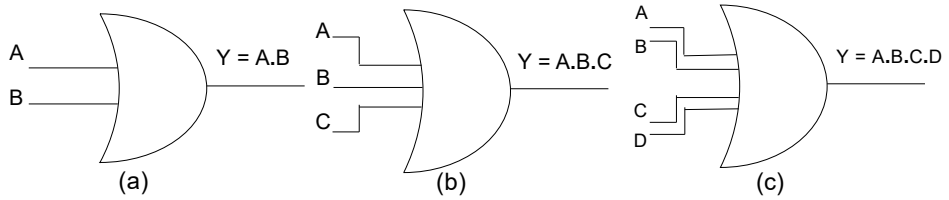


Figure 10.3 : Graphic symbol for (a) two inputs (b) three inputs and (c) four inputs AND gate

The algebric function for a two inputs AND gate is :

Y = A.B

Table 10.3 shows the truth table for this function.

Table 10.3 : Truth table of AND gate

X	Y	F
0	0	0
0	1	0
1	0	0
1	1	1

The AND function can be explained by a series switching circuit as shown in Figure 10.4. It has two switches X and Y in series with a bulb and a power supply. The bulb will glow if and only if both the switches X and Y are simultaneously on.

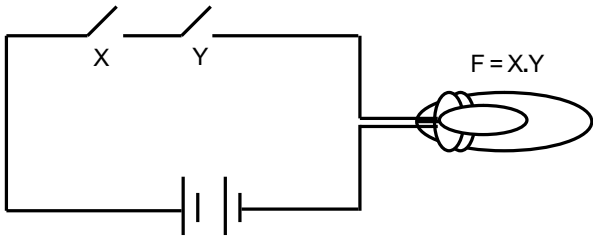


Figure 10.4 : Switching circuit analogy of AND function

In the truth table 10.3,

x = y = 0 represents that the switches are OFF

x = y = 1 represents that the switches are ON

F = 0 represents that the bulb will not glow

F = 1 represents that the bulb will glow.

10.3.3 NOT Gate

NOT gate circuit has a single input and single output. The NOT gate circuit is also called a complementary circuit or an inverter as it complements its input i.e. it accomplishes a logic negation. Figure 10.5 shows the different graphic symbols used for the NOT gate.



Figure 10.5 : Graphic symbols for NOT gate

The algebric function for a NOT gate is : F = x/. Table 6.4 is the

truth table for a NOT gate.

Table 10.4 : Truth table of NOT gate

X	F
0	1
1	0

The NOT gate is understood by the short circuit switch A which when it is closed, (ON) the bulb is bypassed and it does not glow, but when the switch is opened (OFF), the current will flow through the bulb and it would glow. i.e., when A is ON the bulb will be OFF and when A is OFF the lamp will be ON.

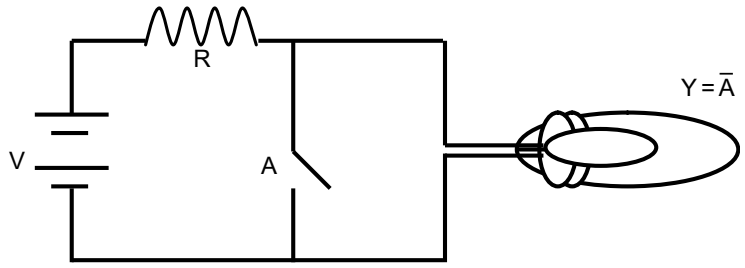


Figure 10.6 : Switching analogy of the NOT function

10.3.4 NAND Gate

A NAND gate is the cascade combination of all AND and a NOT gate. It is an AND gate followed by an inverter. The NAND operation is the complement of the AND operation. Fig. 10.7 shows the graphic symbols for a NAND gate. The algebric function is defined as : $F = (xy)'$. Table 10.5 shows the truth table for NAND operation.

Table 10.5

x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

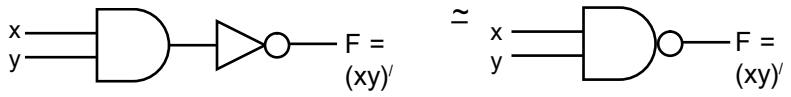


Figure 10.7 : Graphic symbols of NAND gate

10.3.5 NOR Gate

The negation of the OR function is called NOT-OR or NOR. A NOR gate is the cascade combination of NOT and OR gates. The NOR operation is the complement of OR operation. The graphic symbols used normally for a NOR gate are shown in figure 10.8. The NOR function is defined as : $F = (x+y)'$.

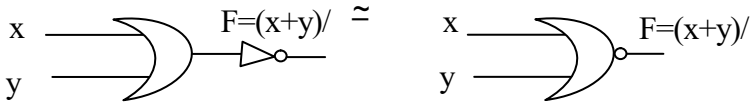


Figure 10.8 : Two Graphic or logic symbols of NOR gate

Table 10.8 : Truth table of NOR gate

x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

Note : Any boolean function can be implemented using NAND or NOR gates. So, NAND and NOR gates are called universal gates.

10.3.6 Exclusive - OR (XOR) gate

The exclusive - OR (XOR) gate has a graphic symbol similar to that of the OR gate, except for the additional curved line on the input side. The output of a two input XOR gate is a logic 1, if the input x or input y is a logic 1 exclusively, i.e., they are not 1 simultaneously. The graphic symbol is shown in Figure 6.9 and the XOR function can be written as $F = X \oplus Y = XY' + X'Y$.

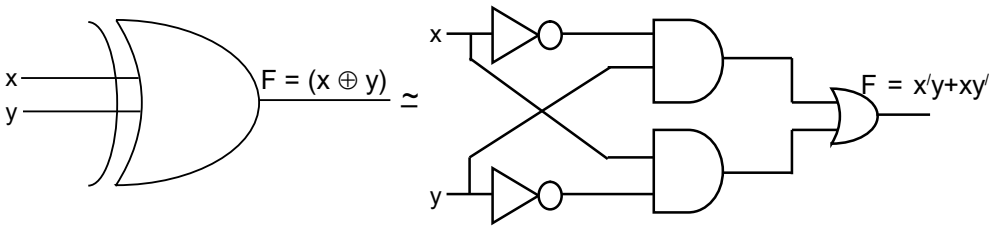


Figure 10.9 : (a) Graphic symbol fo XOR gate (b) XOR gate using basic gates

The truth table of XOR operation is shown in Table 10.7

Table 10.7 : Truth table for XOR operation

X	Y	F = X ⊕ Y
0	0	0
0	1	1
1	0	1
1	1	0

From the truth table, it is clear that the output is 1 (HIGH), when any one of the inputs is at 1 (HIGH). The output is 0 (LOW), when both the inputs are at 1 (HIGH) or at 0 (LOW), i.e. same. In case of more than two inputs, the output of a XOR gate is high when an odd number of inputs is HIGH, such as one or three or five etc. On the otherhand, when there is an even number of HIGH inputs, the output will be always LOW.

10.3.7 XNOR Gate

The XNOR i.e., exclusive NOR gate is the complement of the XOR gate. XNOR function is also called equivalence function. The graphic symbol of XNOR gate is similar to that of the XOR gate, except for the additional inverter gate (or a small bubble) on the output side. The output of a two inputs XNOR gate is a logic 1 (HIGH), if both the inputs are either 1 (HIGH) or 0 (LOW). If the inputs are different (not same), the output is 0 (LOW), In general, we can say the output is 0 (LOW), when the inputs to an XNOR gate have an odd numbers of 1s. The graphic symbol of XNOR gate is shown in Figure 10.10.

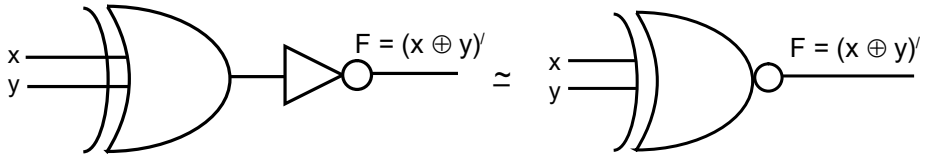


Figure 10.10 : Graphic symbols for XNOR or euivalence gate

The truth table is given in Table 10.8. Here, we see that the output F is the complement of output of the XOR gate. The boolean function for XNOR gate is :

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

$= (x'y)' (xy')'$ Applying DcMorgan's Theorem

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

$= xx' + xy + x'y' + yy'$

$= xy + x'y'$

Table 10.8 : Truth table for XNOR gate

Input		Output
x	y	$F = (x \oplus y)'$
0	0	1
0	1	0
1	0	0
1	1	1

10.4 DE MORGAN'S THEOREM

In the previous unit, we have mentioned the De Morgan's Theorem and its proof for two variables also has been explained using truth table. De Morgan, a great mathematician contributed two most important theorems of Boolean algebra. These two theorem are extremely useful in simplifying an expression in which the product of the sum of variables is complemented. The two theorems can be extended to any number of variables and generalized as:

Theorem 1 : $(A + B + C + ...)' = A' . B' . C' . .....$

Theorem 2 : $(A . B . C . .....)' = A' + B' + C' + ...$

In words, we can write Theorem 1 as: The complement of an OR sum equals the AND product of the complements, and the theorem 2 as :

The complement of an AND product is equal to the OR sums of the complements.

The complement of any boolean function may be found by means of these theorems. It consists of two simple steps to form a complement of a function.

Step 1 : Interchange the symbols / operands "+" and "*".

Step 2 : Each and every term in the expression is complemented.

Example 10.1 : Find the complements of the following functions using De Morgan's Theorem.

(i) $F = (A + B)^{/} (A^{/} + C^{/}) (B^{/} + C)$

(ii) $F = A^{/}B + ABC + AB^{/}C$

Solution : (i) $F = (A + B)^{/} (A^{/} + C^{/}) (B^{/} + C)$

$F^{/} = [(A + B)^{/} (A^{/} + C^{/}) (B^{/} + C)]^{/}$

$= (A + B) + (A^{/} + C^{/})^{/} + (B^{/} + C)^{/}$

$= (A + B) + AC + BC^{/}$

$= A + B + AC + BC^{/}$

(ii) $F = A^{/}B + ABC + AB^{/}C$

$F^{/} = (A^{/}B + ABC + AB^{/}C)^{/}$

$= (A^{/}B)^{/} (ABC)^{/} (AB^{/}C)^{/}$

$F^{/} = (A + B^{/}) (A^{/} + B^{/} + C^{/}) (A^{/} + B + C^{/})$

10.5 TRUTH TABLE

Basic idea of truth table has already been given in the earlier unit. Here, we will learn the process of deriving an expression from a truth table.

The general procedure for obtaining the expression from a truth table in sum of products (SOP) can be summarized as follows:

1 Write an AND term (minterm) for each combination of input variables in the table for which output is 1.

2 Each AND term contains each input variable either in normal form or in complemented form. If the corresponding variable is 0 then it is complemented in the AND term.

3 All the AND terms are then ORed together to produce the final output expression.

Example : 10.2. Obtain the logic function specified by the following truth table. Simplify it using algebric manipulation and implement it with logic diagram.

x	y	z	F	
0	0	0	1 →	$x'y'z'$
0	0	1	0	
0	1	0	0	
0	1	1	0	
1	0	0	1 →	$xy'z'$
1	0	1	0	
1	1	0	1 →	xyz'
1	1	1	1 →	xyz

Solution : The resultant boolean bunction is :

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

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

$= y'z' + xy$

This is the simplified function.

The logic diagram is :

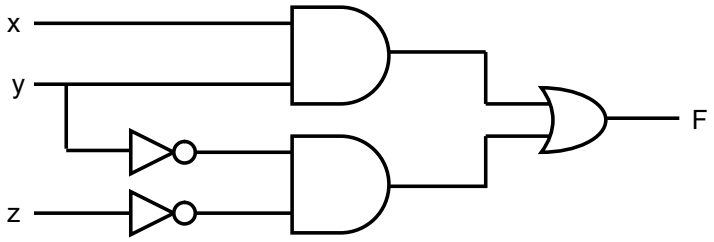


Figure 10.11 : Logic diagram for $F = xy + y'z'$

Note : Since the function has 2 AND terms, we need 2 AND gates and 1 OR gate to implement it.

10.6 CONVERSION OF LOGIC GATES

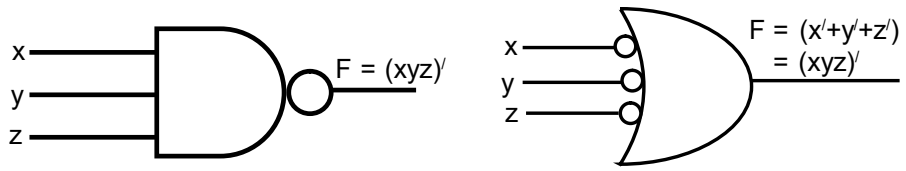
We have already discussed the functions of three basic gates, viz, OR, AND and NOT gates, which perform logical addition, logical multiplication and inversion operations respectively. Besides these, we also have come to know about NAND and NOR gates and their functions. In this section, we

will discuss how the digital circuits can be implemented with NAND or NOR gates.

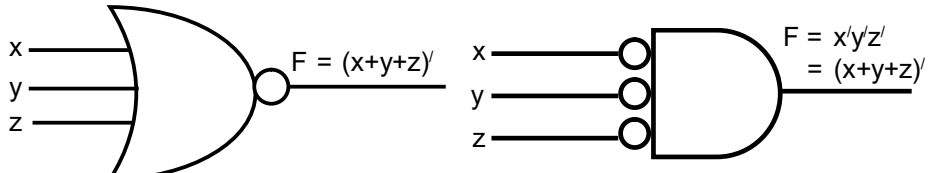
NAND and NOR gates are easier to fabricate with electronic components and are the basic gates used in all IC digital logic families. So, digital circuits are more frequently constructed with NAND or NOR gates than with AND and OR gates. Because of the prominence of NAND and NOR gates in the design of digital circuits, rules and procedures have been developed for the conversion from boolean functions given in terms of AND, OR and NOT into equivalent NAND or NOR logic diagram. Here, we will consider the procedure for only two level implementation.

First, we will define two other graphic symbols for NAND and NOR gates, which will make the conversion procedure easily understandable. Two equivalent symbols for the NAND gate are shown in Figure 10.12 (a).

The AND invert symbol has been defined previously. It is possible to represent a NAND gate by an OR graphic symbol preceded by small circles in all the inputs. This symbol i.e. invert - OR symbol for the NAND gate follows from the De Morgan's theorem. and from the convention that the small circle denotes complementation.



(a) Two graphic symbols for NAND gate



(b) Two graphic symbols for NOR gate



(c) Three graphic symbols for inverter

Figure 10.12 : Graphic symbols for NAND, NOR and NOT gates

The invert AND is an alternative that uses De Morgan's theorem and the convention that small circles in the inputs denote complementation.

A one input NOR gate or NAND gate is equivalent to an inverter. So, an inverter gate can be drawn in three ways as shown in Figure 10.12 (c).

NAND Implementation : To implement a boolean function with NAND gates, it is required that the function is to be simplified in the sum of products (SOP) form. We can see the relationship between a sum of products expression and its equivalent NAND implementation, by considering the logic diagrams of Figure 10.13. All three diagrams are equivalent and implement the function : Y = ABC + DE + F

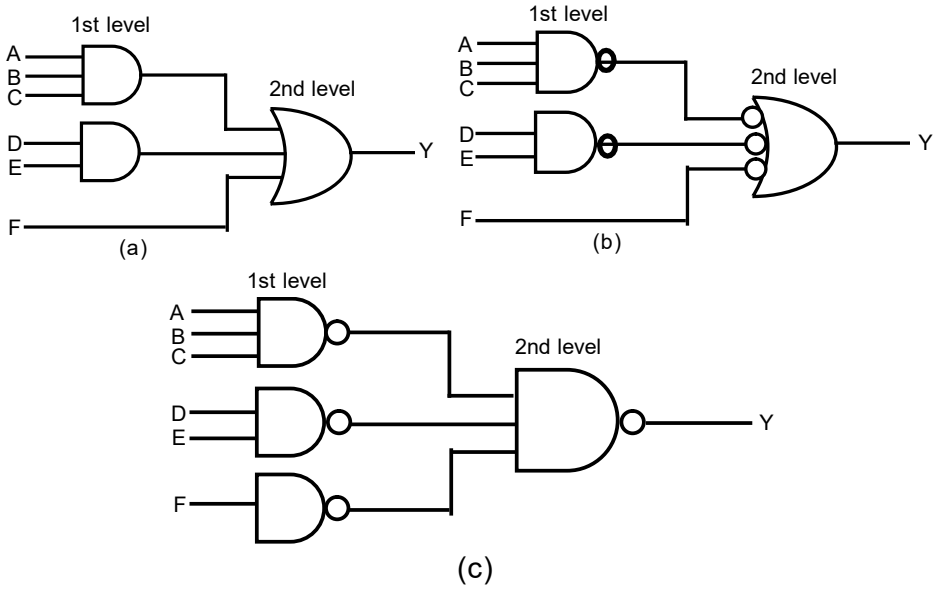


Figure 10.13 : Three ways to implement Y = ABC + DE + F

The function is implemented in SOP form with AND and OR gates in Figure 10.13 (a). The AND gates are replaced by NAND gates and the OR gate is replaced by a NAND gate with an invert OR symbol. The simple variable F is complemented and applied to the second level invert OR gate. A small circle represents complementation. Therefore two circles on the same line represent double complementation and both can be removed. The complement of F goes through a small circle which complements the variable again to produce the normal value of F. Thus, if we remove the small circles in the gates of Figure 10.13 (b), we get Figure 10.13 (a). Therefore, the two diagrams implement the same function and are equivalent.

In Figure 10.13(c),the output NAND gate i.e. the second level NAND gate is replaced with the conventional symbol. The one input NAND gate complements variable F. The diagram in (c) is equivalent to the one in (b), which is turn is equivalent to the diagram in (a). Thus, we implement the circuit, with NAND gates in Figure 6.13(b) or 6.13 (c), which is first implemented with AND and OR gates in Figure 6.13 (a).

The NAND implementation can also be verified algebrically as:

$Y = [(ABC)' . (DE)' . F']'$

$= [(A' + B' + C') (D' + E') . F']'$

$= ABC + DE + F$

From, the transformation shown in Figure 6.13 we see that a boolean function can be implemented with two levels of NAND gates. The rule for obtaining the NAND logic diagram from a boolean function is as follows:

1. Simplify the function in sum of products (SOP).
2. For each product term of the function that has atleast two literals, draw a NAND gate.
3. Draw a single NAND gate (using the AND invert or INVERT -OR graphic symbol) in the second level, with inputs coming from outputs of first -level gates.
4. A term with a single variable requires an inverter in the first level or may be complemented and applied as an input to the second-level NAND gate.

There is a second way to implement a boolean function with NAND gates. We can combine the 0's in a map to obtain the simplified expression of the complement of the function in sum of products. The complement of the function so obtained can then be implemented with two levels of NAND gates using the above stated rules. To obtain the normal output of the circuit, it is required to insert a one input NAND or inverter gate to generate the true value of the output variable. When the designer wants to generate the complement of the function, the second method in preferred.

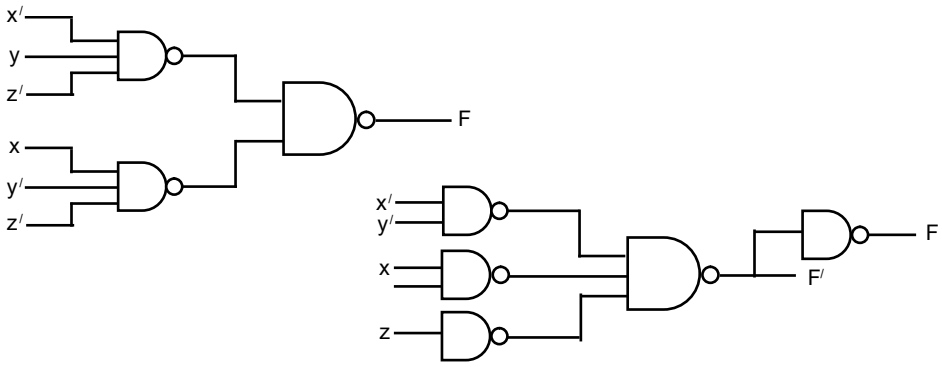
Example 10.3 : Implement the following function with NAND gates F (x, y, z) = $\sum(2,4)$.

The first step is to simplify the function in sum of products form. We draw a map and plot the terms (Figure 10.14).

There are only two 1's in the map, and they cannot be combined, since they are not adjacent to each other. The simplified form of the function in SOP is F = $x'yz' + xy'z'$

x\yz	00	01	11	10
0	0	0	0	1
1	1	0	0	0

(a) Map simplification in SOP F = $x'yz' + xy'z'$



(b) F = $x'yz' + xy'z'$ (c) $F' = x'y' + xy + z$

Figure 10.14 : Implementation of Function in eg. 10.3 with NAND gates

The two level NAND implementation is shown in Figure 10.14 (a) Now, we try to simplify the complement of the function in SOP. This is done by combining to 0's in the map: Thus, $F' = x'y' + xy + z$

The two-level NAND gate for generating F' is shown in Figure 6.14 (c). If output F is required, we have to add a one input NAND gate to invert the function. (We assume that the input variables are available in both the normal and complement forms).

NOR Implementation : The NOR function is the dual of the NAND function. So, all procedures and rules for NOR logic are the dual of the corresponding procedures and rules developed for NAND logic.

The implementation of a boolean function with NOR gates requires that the function be simplified in product of sums (POS) form. A product of sums expression specifies a group of OR gates for the sum terms, followed by an AND gate to produce the product. The transformation from the OR-AND to the NOR-NOR diagram is shown in Figure 10.15, which is similar to the NAND transformation discussed already, except that here we use the product of sums expression :

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

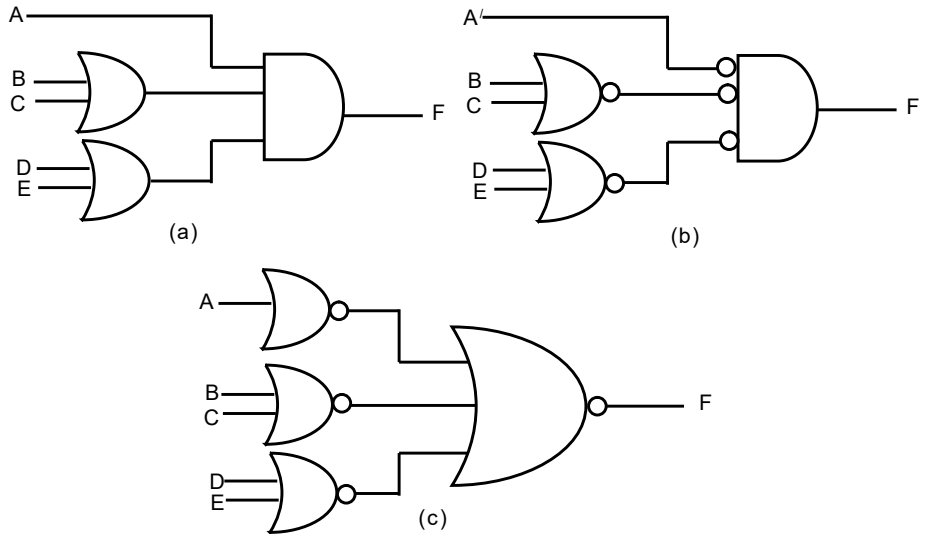


Figure 10.15 : Implementation of the function F = A(B + C) (D + E)

The procedure for obtaining the NOR logic diagram from a boolean function can be derived from this transformation. It is similar to the three step NAND rule, except that the simplified expression must be in the product of sums and the terms for the first level NOR gates are the sum terms. A term with a single variable requires a one input NOR or inverter gate or may be complemented and directly applied to the second-level NOR gate.

Another way to implement a function with NOR gate is to use the expression for the complement of the function in product of sums. It gives a two level implementation for F' and a three level implementation gives the normal output F.

Simplified product of sums can be obtained from a table by combining the 0's and then complementing the function. To obtain the simplified product of sums expression for the complement of the function, we have to combine

the 1's in the map and then complement the function. The NOR gate implementation procedure is demonstrated in the following example.

Example 10.4 : Implement the function of e.g. 10.3 with NOR gates.

The map for this function is drawn in Figure 10.14 (a) Now, combining the 0's we obtain $F' = x'y' + xy + z$

This is the complement of the function in SOP. To obtain it in POS form, as required for NOR implementation, complement F'.

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

The two level implementation with NOR gate is shown in Fig. 10.16.

Another implementation that is possible from the complement of the function in POS is left as an exercise.

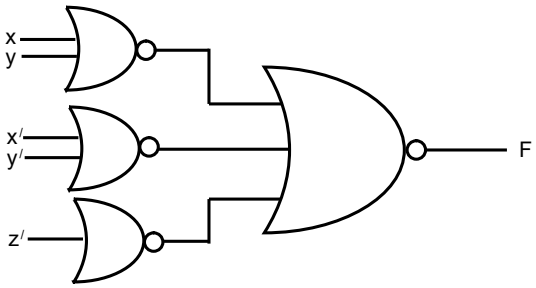
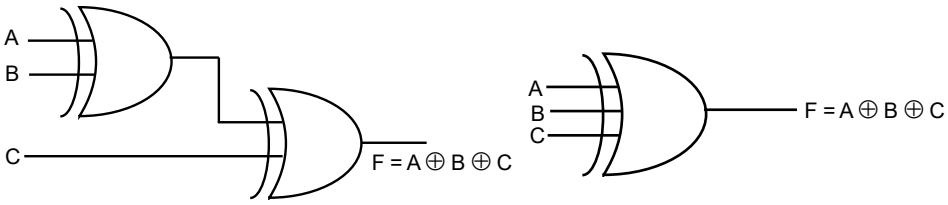


Fig. 10.16 : Implementation of function with NOR gates

Example 10.5 : Draw the logic diagram for the function $F = A \oplus B \oplus C$



(a) using 2-input gates (b) using 3-input gate



CHECK YOUR PROGRESS

Q.1. When the output of an OR gate is High?

Q.2. When the output of an AND gate is HIGH?

Q.3. What is the function of a NOT gate?

Q.4. When the output of a XOR gate is HIGH?

Q.5. Why the digital circuits are more frequently constructed with NAND or NOR gates?



10.7 LET US SUM UP

- A logic gate is an electronic circuit that is used to implement a booleam function by a logic diagram.
- An OR gate produces a HIGH output when atleast one input is HIGH; whereas an AND gate produces a HIGH output when all inputs are HIGH.
- A NAND gate is an AND gate followed by an inverter. It produces a LOW output if all its inputs are HIGH.
- A NOR gate is an OR gate followed by an inverter. It produces a HIGH output when all its inputs are LOW.
- The realization of basic qater viz., AND, OR, and NOT can be made by using either NAND or NOR gates. For this reason, NAND and NOR gates are called universal gates.



10.8 FURTHER READING

- Mano, Morris (2007), *Digital Logic and Computer Design*. Pearson Education.
- Kumar, A. Anand. (2003), *Fundamental of Digital Circuit*, New Delhi, PHI.



10.9 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : If atleast one of the inputs is HIGH.

Ans. to Q. No. 2 : If all the inputs to it are HIGH.

Ans. to Q. No. 3 : The function of the NOT gate is to invert / complement its single line input variable or function.

Ans. to Q. No. 4 : If both the input variables to a XOR gate is not equal then the output is HIGH.

Ans. to Q. No. 5 : Because, NAND and NOR gates are easier to fabricate.



10.10 MODEL QUESTIONS

Q.1. What is a logic gate?

Q.2. List the three basic logic operations.

Q.3. What are the positive and negative logic?

Q.4. What is the only set of input conditions that will produce a LOW output for an OR gate?

Q.5. Write a truth table for a 3 input OR gate?

Q.6. Write a Boolean expression for a 4 input AND gate

Q.7. What is the only input combination that will produce a HIGH at the output of a 4 input AND gate?

Q.8. What input/ logic level should be applied to the second input of a 2-input AND gate to inhibit the logic signal at the first input from reaching output?

Q.9. Develop a truth table for a 3- input AND gate.

Q.10. Is there any difference between 1 OR 1 and 1+1 (binary addition) ?

Q.11. Name the logic gate which has only one input, and show the logic / graphic sym bol.

Q.12. What are the NAND and NOR gates?

Q.13. Write the logic symbols of NAND /NOR gates and develop its truth table.

Q.14. What are the universal gates?

Q.15. Draw the logic diagram of OR gate using NOR /NAND gate.

Q.16. Draw the logic diagram of AND gate using NOR / NAND gate.

Q.17. Draw the logic diagram of NOT gate using NOR / NAND gate

Q.18. What is an XOR gate ? Write its truth table for 2 variables.

Q.19. Show the logic diagram of an XOR gate using basic gates.

Q.20. Draw the logic diagram of an XOR gate using NAND gates.

Q.21. What is an XNOR gate ? Write its truth table.

Q.22. Draw the logic diagram of an XNOR gate using basic gates.

Q.23. Draw the symbol of an XNOR gate and its Boolean expression

Q.24. Implement the following functions with (i) NAND gates (ii) NOR gates.

Q.25. Write the procedure for obtaing NAND logic implementation of a Boolean function.

Q.26. Write the procedure for obtaing NOR logic implementation of a Boolean function.

Q.27. Explain the operation of 3-input AND / OR gate and realize it using NAND /NOR gates

Q.28. Explain the operation of 2-input XOR gate and realize it using NAND /NOR gates

Q.29. Explain the operation of 2-input XNOR gate and realize it using NAND /NOR gates.

****

UNIT 11 : REDUCTION TECHNIQUES

UNIT STRUCTURE

11.1 Learning Objectives

11.2 Introduction

11.3 Minterm and Maxterms

11.4 Simplification of Boolean Function

- 11.4.1 Algebric Manipulation
- 11.4.2 Map Method
- 11.4.3 Don't Care Condition

11.5 Let Us Sum Up

11.6 Further Reading

11.7 Answers to Check Your Progress

11.8 Model Questions

11.1 LEARNING OBJECTIVES

After going through this unit, you will be able to :

- define and describe minterms and maxterms
- describe the algebric manipulation method
- describe the map method
- describe dont care conditions.

11.2 INTRODUCTION

In the previous units, we have learnt important concepts like number systems, boolean algebra and logic gates. In this unit, you will be introduced to reduction techniques. Minterns and maxterms are introduced in this unit. Simplification of Boolean function using methods like algebric manipulation and map method are covered in this unit.

11.3 MINTERMS AND MAXTERMS

A Boolean function can be expressed in three ways, viz, i) in canonical form, ii) in standard form and in iii) non standard form. To understand these forms we must have the idea on Minterms and Maxterms.

Minterms and Maxterms : Let us consider two binary variables x and y, which may appear in their normal form or its complement form. Now combining these two variables with an AND operation, we get four possible combinations: $x'y'$, $x'y$, xy', xy. Each of these AND terms represents one of the distinct areas in the Venn diagram of the fig 11.1, and is called a ***minterm*** or a ***standard product***.

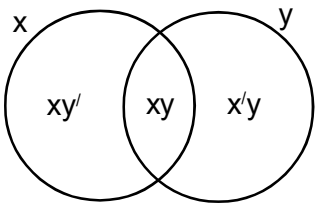


Figure 11.1 : Venn Diagram for two variables

In this way, n variables can be combined to form 2^n minterms. Each minterm is obtained from an AND term of the n variables, with each variable being primed if the corresponding bit of the binary number is a 0 and unprimed if a 1. Each minterm is designated as m_j, where j denotes the decimal equivalent of the binary number of the minterm designated.

Table 11.1 : Minterms and Maxterms for three binary variables

			Minterms		Maxterms	
x	y	z	Term	Designation	Term	Designation
0	0	0	$x' y' z'$	m_0	$x + y + z$	M_0
0	0	1	$x' y' z$	m_1	$x + y + z'$	M_1
0	1	0	$x' y z'$	m_2	$x + y' + z$	M_2
0	1	1	$x' y z$	m_3	$x + y' + z'$	M_3
1	0	0	$x y' z'$	m_4	$x' + y + z$	M_4
1	0	1	$x y' z$	m_5	$x' + y + z'$	M_5
1	1	0	$x y z'$	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

Similarly, with these n variables an OR term can be formed with each variables being primed or unprimed. Thus, we may have 2^n possible combinations and these are called ***Maxterms*** or ***standard sums***. In Table

11.1, the eight maxterms for three variables x, y and z, together with their symbolic designation, are listed. Any 2^n maxterms for n variables may be determined similarly. Each maxterm is obtained from an OR term of the n variables, with each variable being unprimed if the corresponding bit is a 0 and primed if a 1. Each maxterm is the complement of its corresponding minterm and vice versa.

If the truth table for a Boolean function is given we can express the Boolean function algebraically by forming a minterm for each combination of the variables which produces a 1 in the function, and then taking the OR of all those terms.

In the Table 11.2 two functions f_1 and f_2 are listed.

Table 11.2 : Functions of three variables

a	b	c	function f_1	function f_2
0	0	0	1	0
0	0	1	0	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	1
1	1	0	1	0
1	1	1	0	1

From the table, we see that, function f_1 is 1 for four combinations of a, b, c i.e. $a'b'c'$, $a'b\,c$, $a\,b'c$ and $a\,b\,c'$. Thus, for each one of these minterms the function f_1=1, and we can express the function f_1 as:

$f_1 = a'b'c' + a'\,b\,c + a\,b'c + a\,b\,c'$

$= mo + m3 + m5 + m6$

$= \sum (0, 3, 5, 6)$

(The symbol '$\sum$', is used to mean sum of minterms. Similarly, we can easily verify that –

$f_2 = a'\,b\,c' + a\,b'\,c' + a\,b'\,c + a\,b\,c$

$= m_2 + m_4 + m_5 + m_7$

$= \sum (2,4, 5, 7)$

From here we observe an important property of Boolean algebra. Any Boolean function can be expressed as a sum of minterms (sum means Or ing of terms)

We can obtain the complement of f_1 by forming a minterm for each combination of a, b, c, that produce a 0 in the function and then ORing those terms.

Thus $f1' = a' b' c + a' b c' + a b' c' + a b c$

Now taking the complement of $f1'$, we obtain the function f_1:

$(f_1') = f_1 = (a' b' c + a' b c' + a b' c' + a b c)'$

$= (a + b + c') (a + b' + c) (a' + b + c) (a' + b' + c')$

$= M_1 M_2 M_4 M_7$

$= \pi (1, 2, 4, 7)$ (π is used to mean product of maxterms)

This demonstrates a second important property of Boolean algebra: Any Boolean function can be expressed as product of maxterms (product means ANDing of terms).

Now, we are in a position to define the canonical form of a Boolean function. Boolean functions expressed as a sum of minterms or product of maxterms (i.e. each term of the function must have all the literals) are said to be in canonical form.

Sum of Minterms : It is sometimes convenient to express the Boolean function in its sum-of-minterms form. It can be done by first expanding the expression into a sum of AND terms. Then each term is inspected to see if it contains all the variables or literals. If one or more variable is missing in a term, it is ANDed with an expression $(a + a')$, where a is one of the missing variables. Let us take an example to clarify this procedure.

Note : $a + a' = 1$

Example 11.1: Express the Boolean function $xy + x z' + y'$ in a sum of minterms. Here we see that the function has 3 variables and there are 3 AND terms. Two terms missing one variable and one term missing two variables.

Solution : In term xy, z is missing, therefore: $xy (z + z')$

$= x y z + x y z'$

Similarly in $x z'$, y is missing therefore $x z' (y + y') = x y z' + x y' z'$

In y' two variables x & z are missing, therefore:

$y' (z + z') = y' z + y' z'$, still missing x

$\therefore y' z (x + x') + y' z' (x + x')$

$= x y' z + x' y' z + x y' z' + x' y' z'$

Now, combining all the terms

$F = xy + xz' + y'$

$= x y z + x y z' + x y z' + x y' z + x' y' z' + x' y' z + x y' z' + x' y' z'$

Eliminating duplicate terms: (x + x = x)

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

Rearranging the minterms in ascending order

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

$= \sum (0, 2, 4, 5, 6, 7)$

Products of Maxterms : A Boolean function can also be expressed as a product of maxterms. For this, the function is first brought into a form of OR terms. It can be done by using the distributive law x + yz = (x + y) (x + z). Then for any missing variable x in each OR term, the term is ORed with xx'. The process will be clear from the following example.

Example 11.2: Express the Boolean function $f = x y' + x z'$ in a product of maxterm form.

First we convert the function into OR terms using the distributive law.

$f = xy' + x z'$ (Let $x y' = x$, y = x & , $z = z'$)

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

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

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

Now ORing each OR term with $x x'$ (where x is the missing variable)

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

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

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

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

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

Combining all the terms and removing those that appear more than once we finally obtain:

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

$= M_0 M_1 M_2 M_3 M_7$

$= \pi (0, 1, 2, 3, 7)$ (The symbol π denotes the ANDing of maxterms)

Conversion between canonical forms : Boolean functions represented in one canonical form can easily be convered to other canonical forms. It will be clear from the following example.

Example 11.3: Convert the function $F(x, y, z) = \sum (2, 4, 5, 7)$ into product of maxterm form.

Solution : We have $F(x, y, z) = \sum (2, 4, 5, 7) = m_2+m_4+m_5+m_7$

Taking complement $F'(x, y, z) = \sum (0, 1, 3, 6) = m_0+m_1+m_3+m_6$

Now, if we take the complement of F' by De Morgan's theorem,

We obtain $F = (m_0+m_1 + m_3+m_6)'$

$= m_0' m_1' m_3' m_6'$

$= M_0 M_1 M_3 M_6$

$= \pi (0, 1, 3, 6)$

Note : From Table 11.1, it is clear that relation $m_j'=M_j$ is true i.e. a maxterm is a complement of the corresponding minterm and vice versa.

The simplest way to convert from one canonical form to other is: Interchange the symbols $\sum$ and π and list those numbers missing from the original form.

Standard form : In this form, the terms that form the Boolean function may contain one, two or any number of literals. Two types of standard forms are: ***the sum of product*** and ***product of sums***.

The sums of products is a Boolean expression containing, AND terms, called product terms, of one or more literals each.

eg: $F_1 = a + b c' + a b' c$

A product of sums is a Boolean expression containing OR terms, called sum terms. Here also, each term may have any number of literals.

eg: $F_2 = y (x + y') (x' + z) (x + y' + z')$

A Boolean function may also be expressed in a non standard form.

eg: $F_3 = (x' y + x' z) (x' y + x'y')$

This function is neither in sum of product nor in product of sums. Using distributive law it can be converted into a standard form.

11.4 SIMPLIFICATION OF BOOLEAN FUNCTION

We have seen in previous sections that representation of a Boolean function is not unique. The same function can be represented by different numbers of literals and terms. These Boolean functions have to be finally implemented by digital logic gates. The complexity of the digital logic gates is directly related to the complexity of the algebric expression from which the function is implemented. So, it is an important task to simplify the Boolean functions so that they can be implemented easily in a cost effective manner. In this section, we will discuss two methods of simplifying a Boolean function, viz, ***algebric manipulation method*** and the ***map method***.

11.4.1 Algebric Manipulation

On implementing a Boolean function with logic gates, each literal in the function designates an input to a gate, and each term is implemented with a gate. The minimization of the number of literals and the number of terms results in a circuit with less equipment, but it is not always possible to minimize both simultaneously. We will now consider literal minimization. Literal minimization can be done by algebric manipulation. There are no specific rules to follow which will ensure the final answer. The only method available is a cut-and-try procedure using the postulates, the basic theorems and any other manipulation method. To illustrate this procedure let us consider the following examples:

Simplify the following Boolean functions to a minimum number of literals.

1) $x y + x y'$
   $= x (y + y')$
   $= x. 1$
   $= x$

2) $x y z + x' y + x y z'$
   $= x y (z + z') + x' y$
   $= x y + x' y$
   $= y (x + x')$
   $= y.$

3) $xy + x' z + yz$
   $= xy + x' z + y z (x + x')$
   $= x y + x' z + x y z + x' y z$
   $= x y (1 + z) + x' z (1 + y)$
   $= x y + x' z$

In these examples, literals are minimized in 1 and 2, and in 3 a term is minimized.

The algebric manipulation procedure of minimizing a Boolean function is awkward because it lacks specific rules to predict each succeeding step in the process.

11.4.2 Map Method

The map method provides a simple straight forward method for minimizing Boolean functions. It was first proposed by Veitch and slightly modified by Karnaugh and known as the **Veitch diagram** or the "**Karnaugh Map**". This method may be regarded as the pictorial form of a truth table.

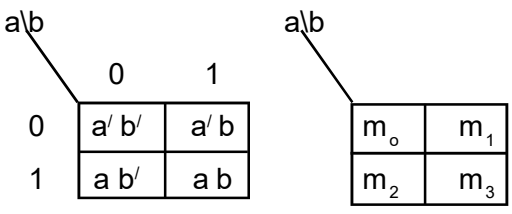
The map is a diagram made up of squares. Each square represents one minterm. As we have already seen that any Boolean function can be represented as a sum of minterms so a Boolean function is recognized graphically in the map from the area enclosed by those squares whose minterms are included in the function.

The number of squares in a K-map depends upon the number of variables of the Boolean expression. For n variables, we have 2^n minterms (possible combinations of n variables) and hence 2^n squares in the map. However, the map method is convenient only for small number of variables, i.e. upto six variables. In this section, we will consider up to four variables map.

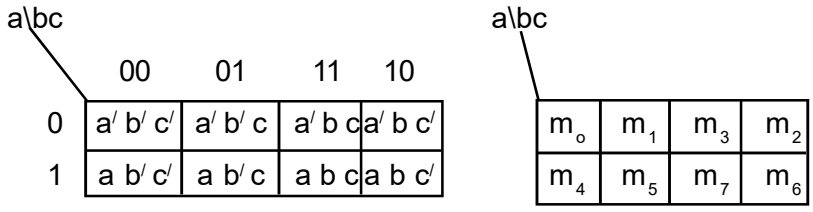
Two, Three and Four variables maps :

K-maps for two, three and four variables map are shown in Fig.11.2 a, b & c.

(a) Two variable K map



(b) Three variables K map



(c) Four variables K map

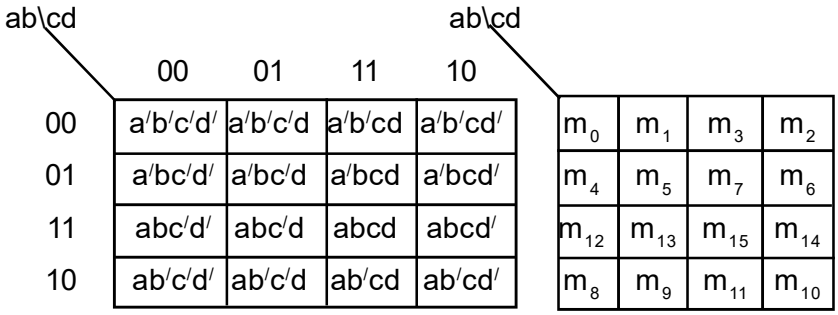
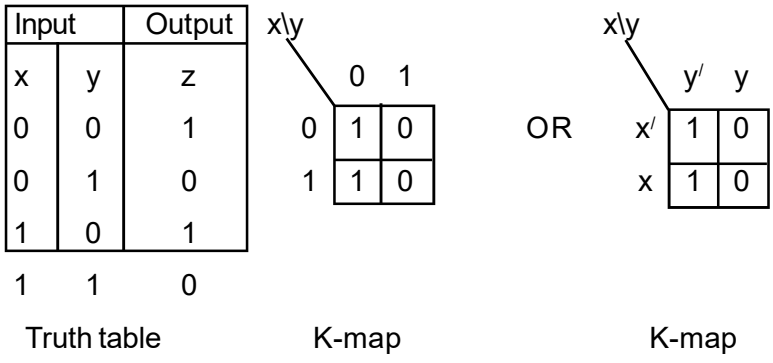


Figure 11.2 : K-map

There are four minterms for two variables; hence the map consists of four squares, one for each minterm. Similarly, for three and four variables there are eight and sixteen minterms respectively and hence the maps consist of eight and sixteen squares.

Representation of Truth Table on K-maps :

We have already known that truth table defines the relationship between output and input.



Truth table K-map K-map

Here, we see that we put 1's in the squares whose minterms are 0 and 2. This is because, from the truth table we see that the output is 1, only for the input combinations x=0, y=0 and x=1 and y=0. For others the

output is 0, and we put 0's in the map accordingly. Similarly, we can represent a truth table having n variables with a K-map of 2^n squares.

Plotting a Boolean expression on the Karnaugh Map : As we have already seen that each square of the K map represents one minterm, so the expression to be plotted must be in sum of product form. The following example clarifies the point.

Example 11.4: Plot the Boolean function $f=yz + xy'z' + xyz'$ on a K-map.

Solution: The term y z is missing one variable, so

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

Now, $F = x' y z + x y z + x y' z' + x y z'$

$= m_3 + m_7 + m_4 + m_6$

$= m_3 + m_4 + m_6 + m_7$

Since the expression has three variables, we need a K-map of $2^3 = 8$ squares.

x\yz	00	01	11	10
0	0	0	1	0
1	1	0	1	1

We plot a 1 in the squares of minterms 3, 4, 6 and 7, and a zero in the other squares.

Simplification of Boolean Function by K-map : We have already seen how a Boolean function can be plotted on the K-map. Now, we will see the process of simplification of Boolean function using K-map. If you observe carefully the K-maps of two, three and four variables you will see that the neighbouring squares are adjacent to each other. Two squares are said to be adjacent if they differ in only one variable. Consider the following three variables k-map.

a\bc	00	01	11	10
0	$a'b'c'$	$a'b'c$	$a'bc$	$a'bc'$
1	$ab'c'$	$ab'c$	abc	abc'

OR

a\bc	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

In this map, m_0 ($a'b'c'$) is adjacent to m_1 ($a'b'c$) because only one variable i.e. "c" appears in m_o in its complement form and m_1, appears in normal form, other variables are same. Similarly, m_o is adjacent with m_4 and m_2; m_1 is adjacent with m_3, m_0 and m_5; m_4 is adjacent with m_o, m_5, and m_6; and so on. When we are going to simplify a Boolean function using K-map, we will consider this adjacency of squares and try to combine as much as possible adjacent squares. We may combine the number of adjacent squares as two, four, eight and sixteen in a four variables map.

By combining two adjacent squares of a four variable map we get a term of three variables or literals.

– combining four adjacent squares we get a term of two literals.

– combining eight adjacent squares we get a term of one literal

– combining sixteen adjacent squares we get a 1 i.e. the function in equal to 1. (i.e. the sum of all minterms is equal to 1).

Example 11.5: Simplify the Boolean function:

Solution: $F = x' y' z' + x' y' z + x y z' + x y z$

	00	01	11	10
0	(1)	(1)		
1			(1)	(1)

Note : Here, adjacent squares are marked with ovals.

After plotting the function, adjacent squares are combined with each other.

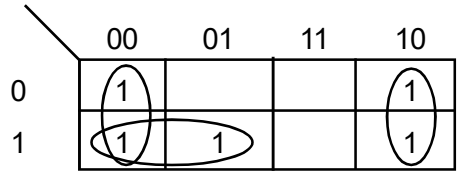
Combining the two squares on top we get the term $x' y'$ (i.e. here variable z is changing, so omit it).

Similarly, combining the two squares on bottom, we get xy.

So, the given function is simplified to $F = x' y' + xy$.

Example 11.6: f (a, b, c) = ∑ (0, 2, 4, 5, 6)

Solution: a\bc



Observing the K-map we see that minterm m_0 is adjacent with m_4 and collectively they are adjacent to m_2 and m_6. This type of adjacency of squares may be determined easily as follows.

Consider the table as an open book, fold it in middle, the squares overlapped with each others are the adjacent squares.

Now, combining m_0, m_2, m_4 and m_6 we get c' (i.e. other variables/ literals are changing, only c' is common to all these squares).

Again, combining m_4 and m_5 we get a b'.

We may combine one square more than once, if necessary.

So, the simplified function is f (a, b, c) = c' + a b'.

Example 11.7: Simplify the Boolean function:

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

Solution : Here, we need a four variable map. The first, second and fourth terms are missing one variable each. So ANDing with (x + x') (where x is the missing variable) we get the function:

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

$+ A B' C' D' + A B' C' D$

Now, plotting the function to the K-map, we get

AB\CD

	00	01	11	10
00	m_0 1	m_1 1	m_3	m_2 1
01	m_4	m_5	m_7	m_6 1
11	m_{12}	m_{13}	m_{15}	m_{11}
10	m_8 1	m_9 1	m_{11}	m_{10} 1

Combining mean terms i.e. squares in the four corners, i.e. m_o, m_2, m_8 and m_{10}, we get $B' D'$.

Combining m_0, m_1, with m_8, m_9 we get $B' C'$. Now, only one square, m_6 remains without combining. We can combine it with m_2 ($\therefore$ they are adjacent) and it gives the term $A' B C'$. Note that if we do not combine m_6, we have to write the term as $A'BCD'$, i.e. with four literals.

So, the simplified form of the function is $F = B' D' + B' C' + A' C D'$

Product of Sum Simplification : In all the previous examples, the minimized Boolean functions derived from the K-maps were expressed in the sum of products (SOP) form. The product of Sums (POS) form can be obtained with a minor modification.

The procedure for obtaining a minimized function in product of sums follows from the basic properties of Boolean functions. The minterms included in the function are represented by placing 1's in the corresponding squares on the map. The minterms not included in the function denote the complement of the function. If we mark these empty squares by 0's and combine them into valid adjacent squares, we obtain a simplified expression of the complement of the function, i.e. of F'. If we take the complement of F, we get back the function F. The function so obtained is automatically in the product of sums form, because of the De Morgan's theorem. The following example clarifies the procedure.

Example 11.8: Simplify the Boolean function

$F (A,B,C,D) = \sum (1, 5, 8, 9, 10)$ in (a) sum of product and (b) product of sums.

Solution : Sum of products simplification of left as an exercise, and we will show only POS simplification.

AB\CD	00	01	11	10
00	0	1	0	0
01	0	1	0	0
11	0	0	0	0
10	1	1	0	1

The squares marked with 0's represents the minterms not included in F i.e., they denote the complement of F. Combining the 0's as shown in the map, we obtain the simplified complemented function:

$F' = AB + CD + A'D'$

Applying De Morgan's theorem, we obtain:

$F' = AB + CD + A'D'$

$(F')' = F = (AB + CD + A'D')'$

$= (AB)' (CD)' (A'D')'$

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

which is the simplified function in product of sums.

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

11.4.3 Don't Care Condition

The function is said to be completely specified when for every possible combination of input variables, we know an output value. Accordingly, these output values (i.e. 1's and 0's) are plotted in the map. The combinations are usually obtained from a truth table that lists the conditions under which the function is a 1. The function is assumed equal to 0 under all other conditions. But this assumption is not always true. There are some applications where certain combinations of input variables never occur. For example, we use 4 bit code to represent the decimal numbers, (0 to 9). Thus, we use the codes from 0000 to 1001, but other 6 codes from 1010 to 1111 are never used. These combinations are called don't care conditions. These don't care conditions can be used on a map to provide further simplification of the function.

It is to be kept in mind that a don't care condition cannot be marked with a 1 on the map because it would mean that the function always produces a 1 for such input combinations. Similarly, for the same reason, we cannot mark these by a 0. To distinguish from 1's and 0's don't care conditions are marked by an X. The X's may be assumed to be either 0 or 1, in choosing adjacent squares to simplify the function in the map depending on which give the simplest expression. An X, which does not contribute to cover a larger area i.e. more minterms/squares, need not be used.

Example 11.9: Simplify the Boolean function

F (A,B,C,D) = ∑ (1,3,7,11,15) with the don't care conditions:

d (A,B,C,D) = ∑ (0, 2, 5) in (a) SOP and (b) POS.

Solution :

Step 1 : Construct a four variables K-map

Step 2 : Squares that represent the minterms of the function are marked with 1s.

Step 3 : The squares for don't care conditions are marked with Xs.

Step 4 : The 1s and Xs are combined in any convenient manner so as to enclose the maximum number of adjacent squares.

AB\CD	00	01	11	10
00	X	1	1	X
01		X	1	
11			1	
10			1	

(a) SOP

AB\CD	00	01	11	10
00	X			X
01	0	X		0
11	0	0		0
0	0	0		0

(b) POS

Step 5 : Combining the squares with 1s and Xs on top in fig. (a), we get the term: $A^/ B^/$

Combining the four 1s we get the term: CD

F (A, B, C, D) = $A^/ B^/$ + CD is the solution in SOP.

Step 6 : The square marked with 0's represent the complement of the function. Combining 0's and Xs in fig (b) we get:

$F^/ = D^/ + AC^/$

$(F^/)^/ = F = (D^/ + AC^/)^/$

= D ($A^/$ + C) - (applying De Morgan's theorem)

This is the simplified POS form of the function.



CHECK YOUR PROGRESS

Q.1. What is a minterm?

Q.2. What is a maxterm?

Q.3. What is a Karnaugh map?

Q.4. What do you understand by sum of products and product of sums?

Q.5. How are the Don't care conditions useful?



11.5 LET US SUM UP

- The three basic logic functions and operations are AND (logical multiplication), OR (logical addition), and the NOT (logical complementation) operation.
- Boolean addition is same as logical OR and Boolean multiplication is same as logical AND operation.
- We can simplify a Boolean function using algebric manipulation method and map method. There is no straight forward procedure in the algebric manipulation method.
- The K-map method is simple systematic and straight forward.
- Any Boolean function can be expressed as a sum of minterms (SOP) and as a product of maxterms (POS). Minterms are AND terms and maxterms are OR terms.
- The variables in a product term (minterm) and in a sum term (maxterm) may appear either in a complemented form or a normal form.
- A function is said to be completely specified, when for every possible combination of input variables the output is defined, otherwise the function is incompletely specified.
- In an incompletely specified function those combination of input variables for which output values are not known i.e. which minterms

or maxterms are not used as parts of the output function are called Don't care terms.

- Don't care terms may be considered as 1s or as 0s, in simplifying a Boolean function.



11.6 FURTHER READING

- Mano, M.M. (1982). Computer System Architecture, Englewood Cliffs, N J, 53-54.
- Mano, M.M. (2017). Digital logic and Computer Design, Pearson Education India
- Ram, B. (2000). Computer Fundamentals Architecture and Organization, New Age International.



11.7 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : A minterm is an AND term also called product term Minterms are obtained by combining the variables, which may appear in their normal form or complement form, with an AND operation. For example, for the two variables "a" and "b", we have four minterms, viz, $a'b'$ (m_o), a' b (m_1), a b' (m_2) and a b (m_3).

Ans. to Q. No. 2 : A maxterm, on the other hand an OR term, also called sum term. Maxterms are obtained by combining the variables with an OR operation. As for example, for two variables "a" and "b", we have four maxterms and they are: a+b (M_0), a+b' (M_1), a'+b (M_2), and a'+b' (M_3).

Note : In case of maxterm, observe that – when a=0, b=0 the maxterm is: a + b i.e. a and b appears in normal form.

Ans. to Q. No. 3 : The Karnaugh map is a pictorial or graphical method used to simplify a Boolean function in an orderly and

systematic manner. It is convenient to use 2 to maximum 6 variable map, but complexity in handling the map increases as the number of variables increase.

Ans. to Q. No. 4 : We have already seen that minterms are product terms. One of the important properties of Boolean algebra is that: any Boolean function can be expressed as a sum of minterms/product terms (SOP). Those minterms, which are present in a function are combined with OR operation and this is called sum of product representation of the function. Example: F (x, y z) = x' y z + x y' z + x y z.

On the other hand, product of sum (POS) form of a Boolean function is obtained by combining the maxterms present in the function with AND operations, Example: F (x, y, z) = (x + y' + z) (x' + y + z) (x' + y' + z).

Note : Since each maxterm is the complement of its corresponding minterms and vice versa, the SOP form of Boolean function is also the complement of its POS form and vice versa.

Ans. to Q. No. 5 : Don't care conditions are plotted in the K-map using the symbol "X". If we can get a larger area by considering these Xs, (i.e. more squares can be combined) then we get a more simplified form of the function. Thus, don't care condition helps in further simplification of a function.



11.8 MODEL QUESTIONS

Q.1. How are the terms "maxterm" and "minterms" related to each other?

Q.2. What is the canonical form of a Boolean function?

Q.3. What is the standard form of a Boolean function?

Q.4. How can you convert a function from one canonical form to other?

Q.5. The minterms of a 4 variable map are marked as: m_0, m_1, m_3, m_2 (in the first raw), not as m_o, m_1, m_2, m_3. Why?

Q.6. Simplify the following Boolean functions in SOP form.

a) F (a, b, c) = (2, 3, 6, 7)

b) F (w, x, y, z) = (7, 13, 14, 15)

c) F (a, b, c, d) = (2, 3, 12, 13, 14, 15)

d) x y $z^/$ + x y z + $x^/$ $z^/$

e) a $b^/$ c + a b $c^/$ + $a^/$ b c + a b c

Q.7. Consider the following truth table.

a	b	c	F_1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

a) Express the function F_1 in product of sum form.

b) Obtain the simplified function in sum of products.

c) Obtain the simplified function in product of sums.

Q.8. Obtain the simplified expression in POS.

a) F (a, b, c) = π (0, 1, 4, 5)

b) F (A, B, C, D) = π (0, 1, 2, 3, 4, 10, 11)

Q.9. Simplify the Boolean function F using the Don't care condition d in (1) SOP and (ii) POS.

(a) F = $w^/$ $x^/$ $z^/$ + $w^/$ y z + $w^/$ x y

d = $w^/$ x $y^/$ z + w y z + w $x^/$ $z^/$

(b) F = $A^/$ $B^/$ C + $A^/$ $B^/$ $C^/$ + $A^/$ B C D + $B^/$ C $D^/$+ A $B^/$ $D^/$

d = $A^/$ B $C^/$ D + $A^/$ B C $D^/$ + A C D

****

UNIT 12 : COMBINATIONAL CIRCUITS

UNIT STRUCTURE

12.1 Learning Objectives

12.2 Introduction

12.3 Combinational Circuit

12.4 Multiplexer

- 12.4.1 2-to-1 Multiplexer
- 12.4.2 4-to-1 Multiplexer
- 12.4.3 8-to-1 Multiplexer
- 12.4.4 16-to-1 Multiplexer
- 12.4.5 Multiplexer Application

12.5 De Multiplexer

- 12.5.1 1 to 16 De Multiplexer

12.6 Encoder

- 12.6.1 Octal to Binary Encoder
- 12.6.2 Decimal to BCD Encoder

12.7 Decoder

12.8. 13-to-8 Decoder

12.9 Magnitude Comparator

12.10 Let Us Sum Up

12.11 Further Reading

12.12 Answers to Check Your Progress

12.13 Model Questions

12.1 LEARNING OBJECTIVES

After going through this unit, you will be able to :

- define and describe multiplexer and de-multiplexer
- describe different types of multiplexer and de-multiploxer
- identify encoder and decoder
- describe magnitude comparator

12.2 INTRODUCTION

In the previous unit, we have learnt about the reduction techniques. Maxterms and minterns, along with methods to simplify Boolean functions are discussed in the previous unit. Map method and algebraic manipulation method here discussed in detail. In this unit, you will be introduced to the concept of combinational circuit. Multiplexer, De-multiplexer, encoder and decoder are also introduced in this unit. The digital circuits in which the status of the output at any instant of time solely depend on the status of the input at that instant are called combinational circuits. Different combinational circuits are discussed in detail in this unit. In the next unit, binary arithmetic circuits will be explored in detail.

12.3 COMBINATIONAL CIRCUIT

A combinational circuit is a collection of some interconnected set of logic gates whose outputs at any time are determined directly from the present combination of inputs without regard to previous inputs.

A combinational circuit consists of input variables, logic gates, and output variables. The logic gates accept binary signals from the inputs and generate some binary signals to the outputs. Here, binary signals mean only two possible values, one representing logic 1 and the other logic 0. A combinational circult can be represented as a network of gates and, therefore, can be expressed by a truth table or boolean expression. A block diagram of combinational circuit is shown in Fig. 12.1.

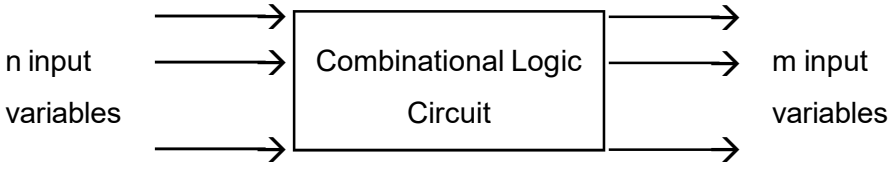


Figure 12.1 Block Diagram of Combinational Circuit

Here, n input binary variables come from an external source and the m output variables go to an external destination. For n input variables, there are 2n possible combinations of binary input values. For each possible input combination, there is only one posible output combination. A

combinational circuit can be described by m boolean functions, one for each output variable. Each output function is expressed in terms of the n input variables.

The basic design issue related to combinational circuits is: the minimization of number of gates. The normal circuit constraints for combinational circuit design level.

- the depth of the circuit should not exceed a specific level.
- number of input to a gate and to how many gates its output can be fed.

12.4 MULTIPLEXER

In the simplest term multiplexer is a combinational circuit which perform sharing. A digital data multiplexer is a logic circuit that has many inputs and only one output. It can select any one of its many inputs by applying a control signal and steer the selected input to the output. A multiplexer is also called a data selector. Generalized block diagram of a multiplexer is shown in Figure 12.2

It has 2^n inputs where, n is the numbers of control lines and only one output. A multiplexer is also called a many – to – one data selector.

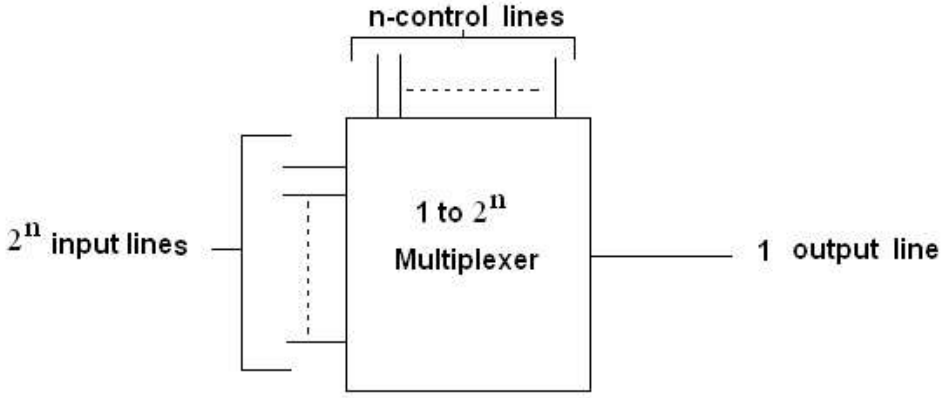


Figure 12.2 : Block Diagram of Multiplexer

12.4.1 2-to-1 Multiplexer

A 2-to-1 multiplexer has 2 inputs and a single control line as shown in Figure 12.3. Here, data inputs are denoted by I_0 and I_1. The control signal A specifies which input is routed to the output. When A = 0, then I_0 is routed to the output and when A = 1 then I_1 is routed to the output.

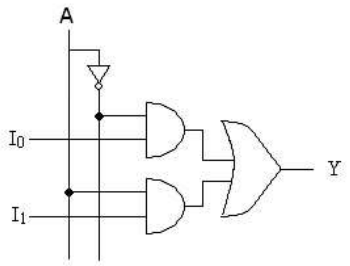


Figure 12.3 : 2-to-1 Multiplexer

12.4.2 4-to-1 Multiplexer

To construct a 4-to-1 multiplexer we use two control signals which can control four inputs. The two bit binary number at the control A and B specifies which of the four data inputs is to go to the output. The four data inputs are termed as I_0, I_1, I_2 and I_3. If AB = 00, then I_0 is allowed to go to the output. If AB = 01, the I_1 is allowed to go to the output. In this manner, AB = 10 will take I_2 and AB = 11 will take I_3 to the output. The circuit for the 4-to-1 multiplexer is shown in Figure 12.4.

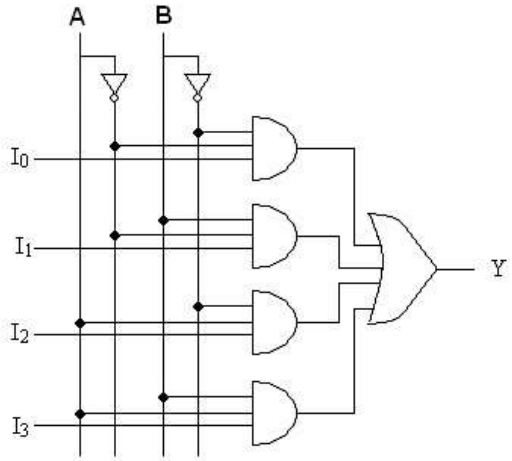


Figure 12.4 : 4-to-1 Multiplexer

12.4.3 8-to-1 Multiplexer

Figure 12.5 shows a 8-to-1 multiplexer, where there are 8 inputs, 3 control lines and 1 output. The eight inputs are labeled as $I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7$ and the control lines are as A, B, C. Here, which input is steered to the output depends on the value of ABC. For example, if ABC = 000 then the upper AND gate is enabled and all other AND gates are

disabled. As a result, the input I_0 alone is steered to the output. Similarly if ABC = 110, then the AND gate connected to the data line I_6 is enabled while all the other AND gates are disabled. Therefore, the input I_6 appears at the output. Hence when ABC = 110 , the output is Y = I_6

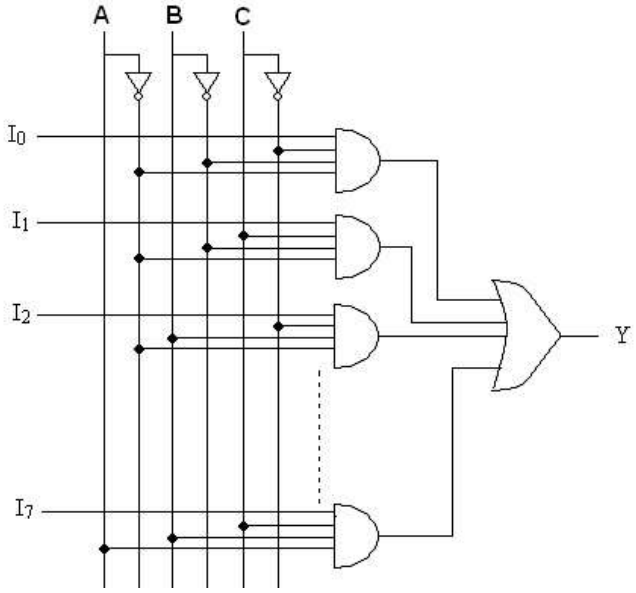


Figure 12.5 : 8-to-1 Multiplexer

12.4.4 16-to-1 Multiplexer

Fig 12.6 shows a 16 – to – 1 multiplexer. In this circuit, there are 16 data input lines, 4 control lines and 1 output. The input lines are denoted by $I_0, I_1, I_2, I_3 ..... I_{15}$ and the control lines are by ABCD. The output is denoted by Y.

Out of 16 input lines, only one is transmitted to the output depending upon the value of ABCD. If ABCD = 0000 then I_0 is steered to the output since the upper AND gate is enabled alone and all others are disabled. Similarly, if ABCD = 0010 then I_2 appears at the output. If ABCD = 1111 then the last AND gate is only enabled and therefore I_{15} appears at the output.

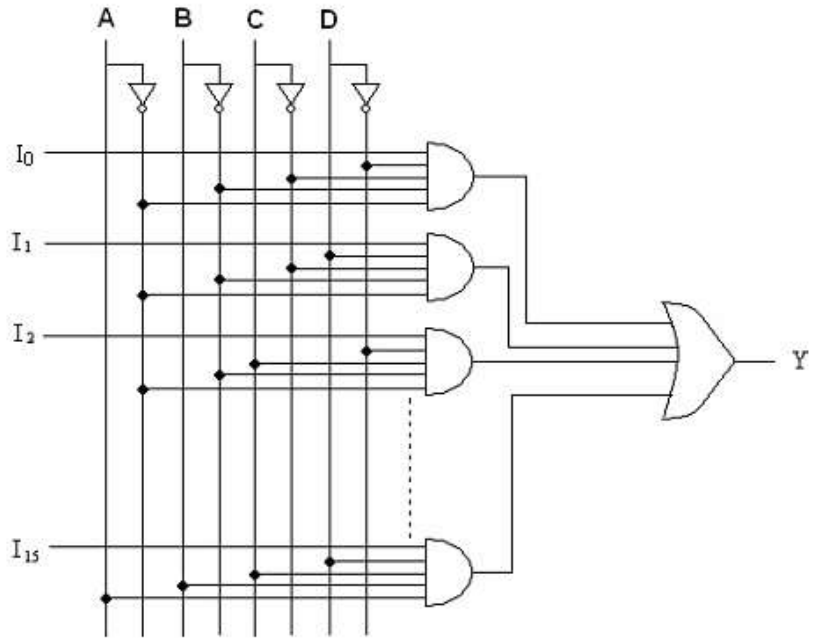


Figure 12.6 : 16-to-1 Multiplexer

12.4.5 Multiplexer Application

- **Function Generator**

A multiplexer can perform any logic function which otherwise needs logic gates to implement. It can be connected in a manner so it duplicates the logic of any truth table. In such application, the multiplexer is viewed as a function generator. One advantage of this use of a multiplexer is that a single IC can perform a function that might need many ICs. Another advantage is that the implemented function can be easily changed if required due to changing necessity.

In the design of a function generator using a multiplexer first the truth table is to construct that is corresponding to the logic expression to be implemented. The next step is to connect logic 1 to each input of the multiplexer corresponding to each combination of input variables for which there is 1 in the output of the truth table. Then, to connect logic 0 to all remaining inputs of the multiplexer. The variables of the truth table are used as control signals. The application can be best understood by one example. Here, we implement the logic function

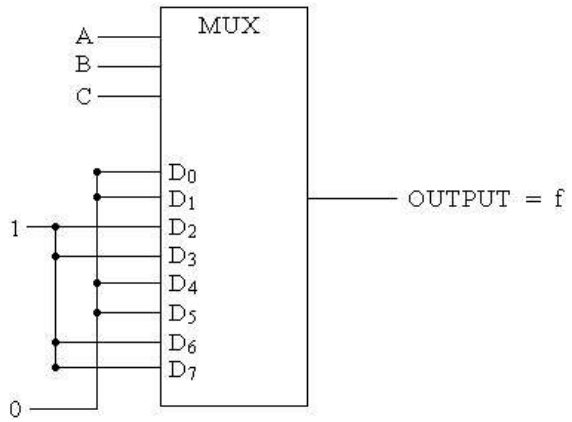
$$f = AB + \overline{A}BC + B\overline{C}$$

The truth table for the function is :

Truth Table

A	B	C	AB	$\overline{A}BC$	$B\overline{C}$	$f = AB + \overline{A}BC + B\overline{C}$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	1	1
0	1	1	0	1	0	1
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	1	1
1	1	1	1	0	0	1

Since the expression has three variables we use these three variables as three control or selector signals. As such the multiplexer must have $2^3 = 8$ data inputs line. The Boolean expression has output 1 for ABC = 010, 011, 110 and 111. We therefore connect logical 1 to data inputs 2, 3, 6 and 7 of the multiplexer and logical 0 to all the remaining inputs. Figure 10.7 shows the circuit of the function generator for the chosen Boolean expression.



Figurer 12.7 : Multiplexer as a function generator

- **Multiplexing Seven-Segment Display**

Seven-segment displays are often used as optical output display in many applications. These displays consume considerable power. To minimize power consumption in small portable device like pocket calculator,

where several such displays are to be illuminated simultaneously, multiplexers are used. In this technique, all the displays are not illuminated simultaneously, instead they are illuminated one after another in a round robin fashion. If all the displays are illuminated about 30 times per second, the human eye will not be able to detect any flicker, rather the displays appear as illuminated steadily. The power consumption is greatly reduced in this technique.

12.5 DE-MULTIPLEXER

A demultiplexer steers a single input to one of many outputs. It is opposite in function to the multiplexer. De-multiplexer means One-to-Many. It can be visualized as a distributor, since it distributes the same data to different output terminals. As such it has 1 input and many outputs. With the application of appropriate control signal, the common input data can be steered to one of the output lines. A good example of demultiplexer is the distributor of an automobile ignition system where electric pulse is distributed among different fuel ignition plugs. Fig 12.8 shows a generalized block diagram of a de-multiplexer. To have 2^n output lines, there must be n control lines in a de-multiplexer.

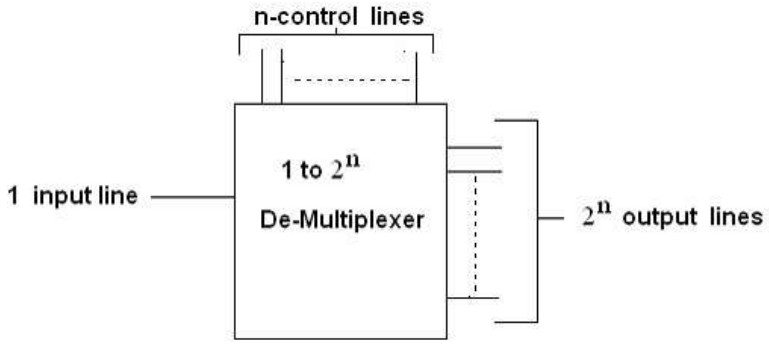


Figure 12.8 : Block diagram of a De-Multiplexer

12.5.1 1-to-16 De-Multiplexer

In Figure 12.9 we have shown a 1-to- 16 de-multiplexer.

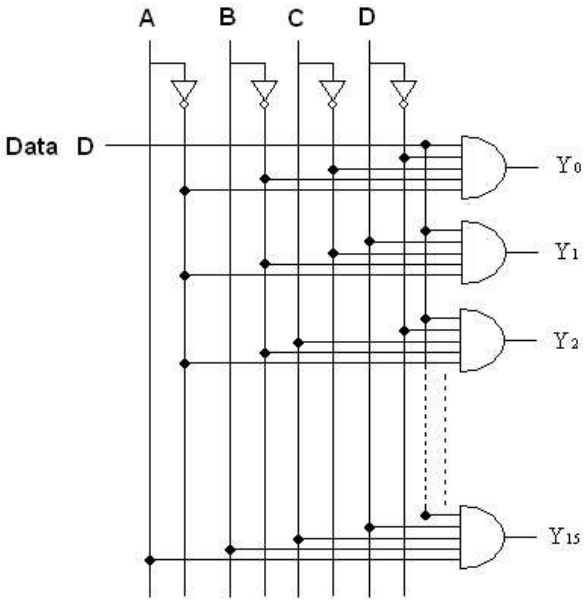
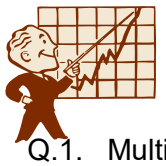


Figure 12.9 : 1-to-16 De-Multiplexer

Here the data input line is denoted by D. This input line is connected to all the AND gates through which output appears. Depending upon the control signal, only one AND gate becomes enabled and the data input D appears through that AND gate. So, when ABCD = 0000, the upper AND gate is enabled and data input D appears at Y_0 as output.

When ABCD = 1111, the bottom AND gate becomes enabled and D appears at Y_{15} as output. For other combination, D appears at other output terminal.



CHECK YOUR PROGRESS

Q.1. Multiplexer means :

(a) multiple to many (b) one to many

(c) many to one (d) one to one

Q.2. A 16-to-1 multiplexer has

(a) 1 control lines (b) 2 control lines

(c) 3 control lines (d) 4 control lines

Q.3 De-multiplexer means

(a) deduct multiple bits (b) one-to-many

(c) multiple-to-multiple (d) one-to-one

Q.4. To implement the logic function $F = AC + BC + \overline{A}\overline{B}\overline{C}$ we need a

(a) 2-to-1 multiplexer (b) 4-to-1 multiplexer

(c) 8-to-1 multiplexer (d) 16-to-1 multiplexer

12.6 ENCODER

An encoder is a combinational circuit which has 2n or less input lines and n output lines. An example of an encoder is the octal to binary encoder which has eight inputs, one for each of the eight digit and three outputs that generates the corresponding binary number. A generalized view of an encoder is shown in Figure 12.10.

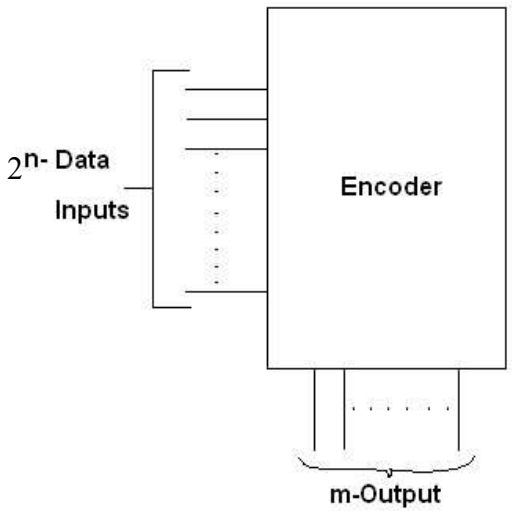


Figure 12.10 : Block Diagram of an Encoder

In the figure we can see that there are 2^n input lines and n – numbers of output lines. Out of inputs, only one input line is active at a time. Encoder generates a coded output which is unique for each of the active input.

12.6.1 Octal-To-Binary Encoder

An octal-to-binary encoder is a device which takes octal number as input and generates the equivalent binary number. Figure 12.11 shows a circuit of an octal-to-binary encoder. It is assumed here that any one input keys is pressed at a time. The keys are connected to a +5 Volt. When the input key is pressed, 5 volt signal is passed to the horizontal lines to which three OR gates are connected selectively.

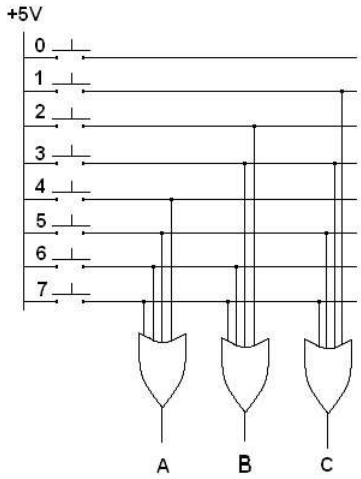


Figure 12.11 : Octal-to-Binary encoder

12.6.2 Decimal-To-BCD Encoder

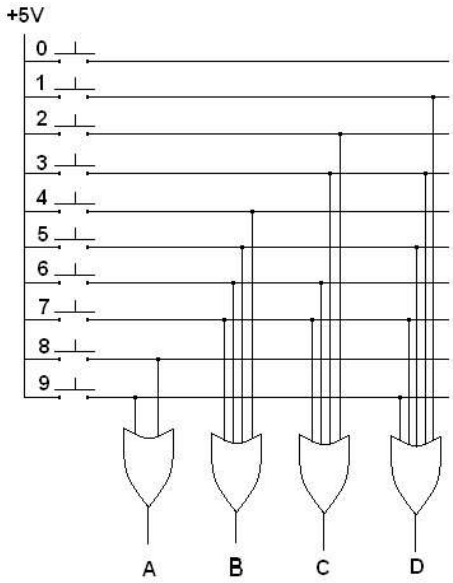


Figure 12.12 : Decimal-to-BCD Encoder

A decimal to BCD encoder is shown in Fig 10.12. This circuit generates BCD output when any one of the push button switches is pressed. As for example, if button 6 is pressed, the B and C OR gates have high ABCD = 0110. If button 8 is pressed, the OR gate A receives a high input and therefore the output becomes ABCD = 1000.

12.7 DECODER

A decoder is a digital circuit which has n-input lines and 2^n or less output lines. A decoder and a de-multiplexer share similarity. In a de-multiplexer, there is a single input line connected to every output 'AND' gate whereas in a decoder the input line is absent. Various kinds of decoder are constructed to decode signals given in different forms. The input to a decimal decoder is a binary code and the output is the decimal digit the code represents. Some decoders take 3 bits binary inputs and give octal outputs.

In practical applications, decoders are used to select one of many devices in a computer system. For example, a computer system has many peripheral devices like hard disk drive, DVD drive, printer etc for each of which there exist an unique address. Generally all the devices are connected to the computer system through a decoder. The computer generates the address of a particular device when it wants to access it in the form of binary code. The decoder receives that address as input and activates the lines connecting the particular device to make it enable.

12.7.1 3-to-8 Decoder

A 3 – to – 8 decoder is shown in Fig 10.13. It's truth table is shown below :

Truth table for a 3-to-8 decoder

Inputs			Outputs							
A	B	C	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

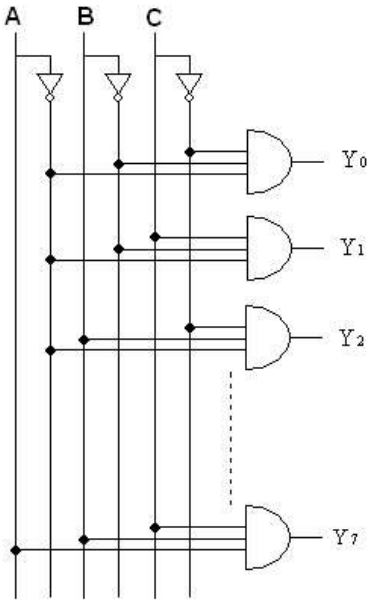


Figure 12.13 : 3- to -8 Decoder

12.8 MAGNITUDE COMPARATOR

A magnitude comparator circuit can compare two binary numbers to determine whether one is greater than the other or if they are eqqual. Such a magnitude comparator has three output lines for A > B, A = B, A < B where A and B are two n-bits binary numbers. Every bit of one number is compared with the corresponding bit of the other number by ExOR gate.

A 4-bit magnitude comparator, SN 7485 is available in chip form the block diagram of which is shown in Fig 12.14. It compares two 4-bit binary numbers $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$. Three output terminals are available for A < B, A = B and A > B.

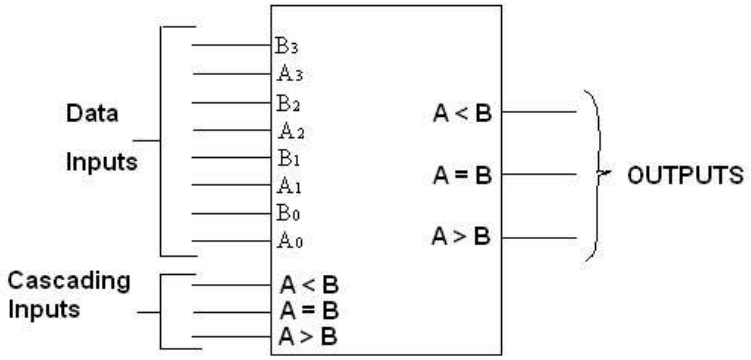


Figure 12.14 : Two 4-bit words magnitude comparator SN 7485

FUNCTION TABLE SN 7485

Comparing Inputs				Cascading Inputs			Outputs		
A_3B_3	A_2B_2	A_1B_1	A_0B_0	A > B	A < B	A = B	A > B	A < B	A = B
$A_3 > B_3$	X	X	X	X	X	X	H	L	L
$A_3 < B_3$	X	X	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 > B_2$	X	X	X	X	X	H	L	L
$A_3 = B_3$	$A_2 < B_2$	X	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	X	X	X	X	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	X	X	X	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	H	L	L	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	H	L	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	L	H	L	L	H

To compare any number having more than 4-bits, two or more such chips can be cascaded. The A > B, A < B and A = B outputs of a stage that handles less significant bits are connected to the corresponding cascading inputs of the next stage that handles the more significant bits.



CHECK YOUR PROGRESS

Q.5. An encoder :

(a) converts a digital input to another form of digital output.

(b) converts analog input to digital output

(c) selects one out of many inputs (d) none of these.

Q.6. Decoder has n inputs line and

(a) n output lines (b) 2^n output lines.

(c) n^2 output lines (d) no output lines.

Q.7. Magnitude comparator

(a) compares two multi bit binary number

(b) magnify any digital signal

(c) compress binary numbers.

(d) check error in a binary number.



12.9 LET US SUM UP

- A combinational circuit is some combinations of logic gates as per specific relationship between inputs and outputs.
- A multiplexer is a combinational circuit which selects one of many inputs.
- Demultiplexer is opposite to a multiplexer.
- An encoder generates a binary code for 2^n input variables.
- A decoder decodes an information receives from n input lines and transmits the decoded information to maximum outputs.
- A magnitude comparator circuit can compare two binary numbers to determine which one is greater than the other or their equality.



12.10 FURTHER READING

- Mano, M. M. (2017), *Digital Logic and Computer Design*, Peason Education India.
- Mano, M. M. (1982), *Computer System Architecture*, Englewood Oliffs, N.J., 53-54.
- Malvino, A.P. & Leach, P. (1986), *Digital Principles and Applications*, Mcgraw-Hill International
- Lee, C. (1976), *Digital Circuits and Logic Design*
- Talukdar P. (2010), *Digital Techniques*, N. L. Publications
- Bogart, F. (1992), *Introduction to Digital Circuits*.



12.11 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : (c)

Ans. to Q. No. 2 : (d)

Ans. to Q. No. 3 : (b)

Ans. to Q. No. 4 : (c)

Ans. to Q. No. 5 : (a)

Ans. to Q. No. 6 : (b)

Ans. to Q. No. 7 : (a)



12.12 MODEL QUESTIONS

Q.1. Define combinational circuit with example.

Q.2. Using truth table and logic diagram explain the working of a full-adder circuit.

Q.3. Using truth table and logic diagram explain the working of a full-subtractor circuit.

Q.4. Draw and explain the operation of a 4-bit parallel adder.

Q.5. Explain how a serial adder works using block diagram.

Q.6. Define multiplexer? Using diagram explain the working of a 8-to-1 multiplexer.

Q.7. Use an 8-to-1 multiplexer to generate the logic function

Q.8. Explain the principle of an encoder. Draw a decimal-to-BCD encoder.

Q.9. What is a decoder? Draw and explain the working of a 3-to-8 decoder.

Q.10. What is magnitude comparator? Draw block diagram and the function table of the magnitude comparator SN 7485.

****

UNIT 13 : BINARY ARITHMETIC CIRCUITS

UNIT STRUCTURE

13.1 Learning Objectives

13.2 Introduction

13.3 Half-Adder

13.4 Full-Adder

13.5 Parallel Adder

13.6 Serial Adder

13.7 Subtractor Circuit

- 13.7.1 Half-Subtractor
- 13.7.2 Full-Subtractor

13.8 Let Us Sum Up

13.9 Further Reading

13.10 Answer to Check Your Progress

13.11 Model Questions

11.1 LEARNING OBJECTIVES

After going through this unit you will be able to :

- describe working principle of half-adder
- describe working principle of full-adder
- describe working of parallel adder
- describe working of serial adder
- describe working principle of half-subtractor
- describe working principle of full-subtractor

11.2 INTRODUCTION

In the previous unit, you have already been accustomed with different logic gates and their associated boolean expressions. Understanding of these would help you to follow the subject matter of this unit.

The advancement in technology has made it possible to fabricate

cheap **SSI (Small Scale Integration)** circuits, **MSI (Medium Scale Integration)** functional devices and **LSI (Large Scale Integration)** systems. Example of these are single chip pocket calculator, single chip computer memories and single chip CPU etc. Many of these products now become building blocks to design other higher level digital devices.

Adder circuits are used to add binary bits. If we require to find sum of two 1-bit we use a circuit called *half-adder*. Sometimes it becomes necessary to add three 1-bit binary numbers. The circuit for this is called *full-adder*.

13.3 HALF-ADDER

In the truth table, the *sum(S)* and *carry(C)* bits are shown as result which are obtained when two 1-bit number *X* and *Y* are added. The truth table contains all the possible combination of values of these two numbers. From the table two individual circuits can be constructed to obtain *sum* and *carry* in response to every combination of the two input numbers. Combining both a single circuit is constructed to treat it as a single combinational circuit to give us two outputs viz *sum* and *carry.* The circuit is generally termed as *half-adder.*

Hence we can say that a half-adder is a circuit that can add two binary bits. Its outputs are SUM and CARRY.

Truth Table for a Half-Adder

X	Y	CARRY (C)	SUM (S)
0	0	0	0
0	1	0	1 ($\bar{X}Y$)
1	0	0	1 ($X\bar{Y}$)
1	1	1 (XY)	0

The minterms for Sum and CARRY are shown in the bracket. The Sum-Of-Product equation for SUM is :

$S = \bar{X}Y + X\bar{Y}$ ..... (1)

Similarly the SOP equation for the CARRY is :

$C = XY$ ..... (2)

Combining the logic circuits for equation (1) & (2) we get the circuit for Half-Adder as :



Figure 13.1 : Half-Adder Circuit and Symbol

13.4 FULL-ADDER

To add two 2-bit binary numbers, we first add the two least significant bits of each number and take the carry (if any) that generate into the addition of the two most significant bits. Hence the second addition involves addition of three single bits. That necessitates another type of addition circuit which we will discuss here. This new type of adder circuit is known as *full-adder*.

Full-Adder is a logic circuit to add three binary bits. Its outputs are SUM and CARRY. In the following truth table X, Y, Z are inputs and C and S are CARRY & SUM.

Truth Table for Full-Adder

X	Y	Z	CARRY (C)	SUM (S)
0	0	0	0	0
0	0	1	0	1 ($\bar{X}\bar{Y}Z$)
0	1	0	0	1 ($\bar{X}$**Y**$\bar{Z}$)
0	1	1	1 (($\bar{X}$**Y**Z	0
1	0	0	0	1 ($X\bar{Y}\bar{Z}$)
1	0	1	1 ($X\bar{Y}Z$)	0
1	1	0	1 ($XY\bar{Z}$)	0
1	1	1	1 (XYZ)	1 (XYZ)

The minterms are written in the brackets for each '1' output in the truth table. From these the SOP equation for full summation can be written as :

$$S = \bar{X}\bar{Y}Z + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XYZ$$

$$= \bar{X}(\bar{Y}Z+Y\bar{Z}) + X(\bar{Y}\bar{Z}+YZ)$$

$$= \bar{X}S + XS \text{..... (3)}$$

Here S is SUM of Half-Adder.

Again SOP equation for Full – Adder CARRY is :

$C = \bar{X}YZ + X\bar{Y}Z + XY\bar{Z} + XYZ$

$= \bar{X}YZ + XYZ + X\bar{Y}Z + XY\bar{Z}$

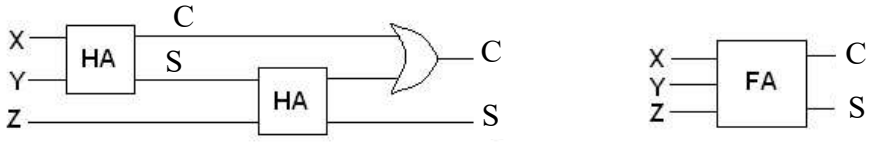
$= (\bar{X} + X)\, YZ + X\,(\bar{Y}Z + Y\bar{Z})$

$= YZ + XS$

$= C + XS$ ..... (4)

Here also, C means CARRY of half-adder and S means SUM of half-adder.

Now using two half-adder circuits and one OR gate we can implement equation (3) and (4) to obtain a full-adder circuit as follows.



Figurer 13.2 : Full-Adder Circuit and Symbol

13.5 BINARY PARALLEL ADDER

When two multi-bit binary numbers are to be added under the situation that their bits are available in parallel form, then a parallel adder is used. It can add each corresponding bit of the numbers and simultaneously produce the sum bits as parallel output.

● 4-BIT BINARY PARALLEL ADDER

Block diagram of a 4-bit binary parallel adder capable of adding two 4-bit numbers is shown in Fig 11.3. The numbers to be added are designated as $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ and the corresponding sum bits are $S_3S_2S_1S_0$. To build the parallel adder, one full-adder is required for each pair of corresponding bits in the numbers to be added, except for the pair of least significant bits, for which a half-adder is sufficient. But to facilitate cascading the pair of least significant bits is also added by a full-adder. The carry out from each stage is taken as the carry into the next more significant stage. In figure 13.3 we use four full-adder to construct the 4-bit binary parallel

adder where the carry in to the full-adder for the least significant bits is made logical 0.

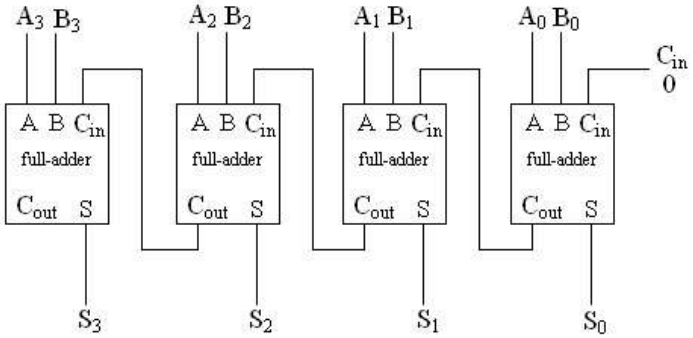


Figure 13.3 : A 4-bit binary parallel adder

13.6 SERIAL ADDER

A serial adder is used to add serial binary numbers. To implement a serial adder, one full-adder is required. In figure 11.4, a serial adder is shown. The bits to be added come serially one after another from both the numbers in synchronism, first A_0 and B_0, then after one clock pulse A_1 and B_1 and so forth. The carry bit generated (C_{out}) in the process is saved for one clock pulse by a flip-flop and it is added to the next higher order pair of input bits as C_{in}. It can be seen from the figure that the output sum bits are shifted into the output register as the input bits are shifted out of the input register. In practice, to minimize the circuit, the output bits are shifted in to one of the input registers behind the data being shifted out. The register which contains one of the binary numbers to be added before the commencement of addition and contains the sum after the addition process is completed, is called an *accumulator*.

It can be easily understood that the serial adder is slower than a parallel adder, as they require one clock pulse to add one pair of bits. A serial adder requires much lesser hardware than a parallel adder and hence it finds application in such devices where space is of much concern than speed. One such application is pocket calculator.

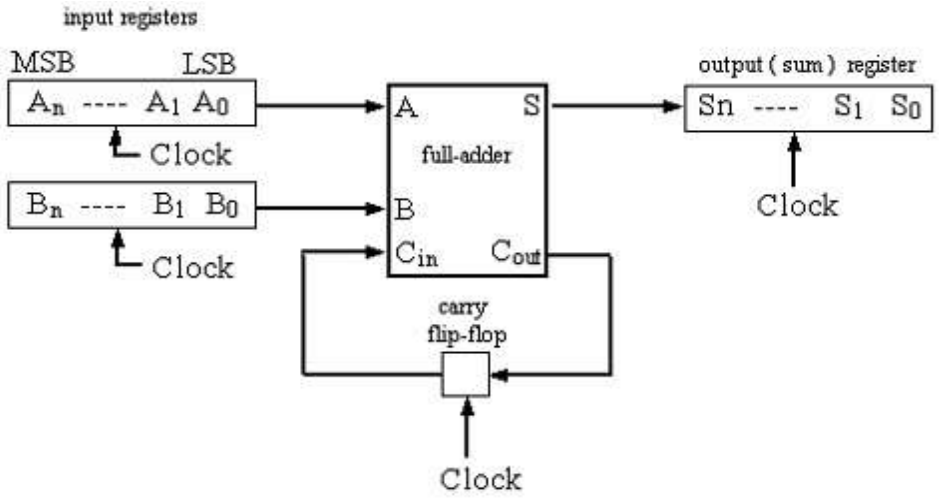


Figure 13.4 : A serial adder

13.7 SUBTRACTOR CIRCUIT

In arithmetic expression subtraction is also a frequently occurring operation like addition. Hence it is necessary to design subtractor circuits involving two and three binary bits. Just like adder circuits, we have two subtractor circuits viz *half-subtractor* and *full-subtractor*.

13.7.1 Half-Subtractor

A half-subtractor subtracts one bit from another bit. It has two outputs viz DIFFERENCE (D) and BORROW (B).

Truth Table for Half-Subtractor

X	Y	BORROW (B)	DIFFERENCE (D)
0	0	0	0
0	1	1 ($\bar{X}Y$)	1($\bar{X}Y$)
1	0	0	1 ($X\bar{Y}$)
1	1	0	0

The mean terms are written within parenthesis for output 1 in each column. The SOP equations are :

$D = \bar{X}Y + X\bar{Y}$

$= S$ ..... (5)

$B = \bar{X}Y$ ..... (6)



Fig. 11.5 : The half-subtractor circuit and symbol

13.7.2 Full-Subtractor

A full-subtractor circuit finds the difference and borrow for the subtraction operation involving three binary bits.

Truth Table of Full-Subtractor

X	Y	Z	BORROW (B')	DIFFERENCE(D')
0	0	0	0	0
0	0	1	1 ($\overline{X}\overline{Y}Z$)	1 ($\overline{Y}\overline{X}Z$)
0	1	0	1 ($\overline{X}Y\overline{Z}$)	1 ($\overline{X}Y\overline{Z}$)
0	1	1	1 ($\overline{X}YZ$)	0
1	0	0	0	1 ($X\overline{Y}\overline{Z}$)
1	0	1	0	0
1	1	0	0	0
1	1	1	1 (XYZ)	1 (XYZ)

The SOP equation for the DIFFERENCE is :

$D' = \overline{X}\overline{Y}Z + \overline{X}Y\overline{Z} + X\overline{Y}\overline{Z} + XYZ$

$= \overline{X}Y\overline{Z} + X\overline{Y}\overline{Z} + XYZ + \overline{X}\overline{Y}Z$

$= (\overline{X}Y + X\overline{Y})\,\overline{Z} + (XY + \overline{X}\overline{Y})\,Z$

$= D\overline{Z} + DZ$ ..... (7)

And SOP equation for BORROW is :

$B' = \overline{X}\overline{Y}Z + \overline{X}Y\overline{Z} + \overline{X}YZ + XYZ$

$= \overline{X}\overline{Y}Z + XYZ + \overline{X}Y\overline{Z} + \overline{X}YZ$

$= (\overline{X}\overline{Y} + XY)\,Z + \overline{X}Y\,(\overline{Z} + Z)$

$= \overline{D}Z + \overline{X}Y$ ..... (8)

In equation (7) and (8), D stands for DIFFERENCE of half-subtractor. Now from the equations (7) and (8) we can construct a full-subtractor using two half-subtractor and an OR gate.

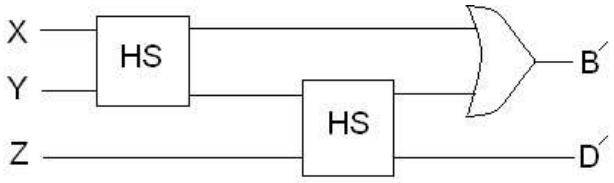
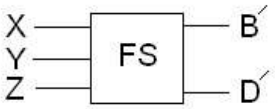


Fig. 13.6 : Full-Subtractor Circuit

The symbol of full-subtractor is :





CHECK YOUR PROGRESS

Q.1. A half-adder can add :

(a) Two binary number of 4 bit each

(b) Two binary bit

(c) Add half of a binary number

(d) None of these

Q.2. A full-adder is a logic circuit that has two output namely:

(a) product & sum

(b) sum & borrow

(c) sum & carry

(d) carry & borrow

Q.3. A parallel adder can be implemented using

(a) Two or more half-adder

(b) Two or more full-adder

(c) One full-adder and remaining half-adder

(d) none of these.

Q.4. A serial adder use

(a) One full-adder

(b) One half-adder

(c) One full-adder with a flip-flop

(d) One full-adder, one flip-flop, three shift registers

Q.5. A half-subtractor can perform

(a) subtraction of two binary bits

(b) product of two binary bits

(c) complement of half binary bits

(d) none of these

Q.6. A full-subtractor has the ability to do

(a) subtraction of two binary numbers

(b) subtraction of three binary bits

(c) product of three binary bits

(d) division three binary bits



13.8 LET US SUM UP

- Adder circuits are used to add binary bits.
- Subtractor circuits are used to subtract bits.
- Adder and subtractor circuits can perform binary addition and subtraction.
- There are different types of adder and subtractor like 2 bit adder/ subtractor, 3 bit adder/subtractor etc.
- Serial adder is used to add serial binary numbers.



13.9 FURTHER READING

- Mano, M. M. (2017), *Digital Logic and Computer Design*, Peason Education India.
- Mano, M. M. (1982), *Computer System Architecture*, Englewood Oliffs, N.J., 53-54.
- Malvino, A.P. & Leach, P. (1986), *Digital Principles and Applications*, Mcgraw-Hill International
- Lee, C. (1976), *Digital Circuits and Logic Design*
- Talukdar, P. (2010), *Digital Techniques*, N. L. Publications
- Bogart, F. (1992), *Introduction to Digital Circuits*.

13.10 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : (b)

Ans. to Q. No. 2 : (c)

Ans. to Q. No. 3 : (b)

Ans. to Q. No. 4 : (d)

Ans. to Q. No. 5 : (a)

Ans. to Q. No. 6 : (b)

13.11 MODEL QUESTIONS

Q.1. Use truth table and logic diagram to explain the working of a half adder and full-adder circuit.

Q.2. Use truth table and logic diagram to explain the working of a half and full-subtractor circuit.

Q.3. Draw and explain the operation of a 4-bit parallel adder.

Q.4. Explain how a serial adder works using block diagram.

Q.5. Give the logic diagram of half adder and half subtractor.

Q.6. Give the truth table for full subtractor.

****

UNIT 14 : SEQUENTIAL CIRCUITS

UNIT STRUCTURE

14.1 Learning Objectives

14.2 Introduction

14.3 Sequential Circuits

14.4 Flip-Flop

- 14.4.1 RS Flip-Flop
- 14.4.2 D Flip-Flop
- 14.4.3 JK Flip-Flop
- 14.4.4 MS Flip-Flop

14.5 Let Us Sum Up

14.6 Further Reading

14.7 Answers to Check Your Progress

14.8 Model Questions

14.1 LEARNING OBJECTIVES

After going through this unit, you would be able to :

- define flip-flops
- describe the working principle of flip-flops
- describe different flip-flops like RS, D, JK and Master Slave flip-flop.

14.2 INTRODUCTION

In the previous units, we have learnt about different combinational circuits. We have learnt about multiplexer, de-multiplexer, encoder and decoders. We have also been introduced to binary arithmetic circuits like half adder, full adder, half subtractor and full subtractor.

In this unit, you will be introduced to sequential circuits and flip flops. Different flip flops like RS flip flop, D flip flop, JK flip flop, D flip flop and MS flip flop are covered in this unit. In this unit we will discuss several sequential circurits in detail.

14.3 SEQUENTIAL CIRCUITS

If the output of a circuit depends on its present inputs and immediate past output, then the circuit is called sequential circuit. To build a sequential circuit, we need memory circuits along with some other logic gates. Flip-Flops are used as memory circuit, the application of which we would see in counter, register etc.

We already know from unit 10, that a combinational circuit has no memory element. A combinational circuit has some basic differences with a sequential circuit. The most important difference is that sequential circuits has memory elements and the present value of outputs is determined by the present inputs and its present state. Some differences are listed below:

Combinational Circuit	Sequential Circuit
• It contains no memory element	• It contains memory elements
• The present value of its outputs is determined solely by the present values of its inputs	• The present value of its outputs is determined by the present values of its inputs and its present state
• Its behavior is described by the set of output functions	• Its behavior is described by the set of next-state functions and the set of output functions.

14.4 FLIP-FLOPS

A digital circuit that can produce two states of output, either high or low is called a multi-vibrator. Multivibrators are further divided into three types, viz. monostable, bi-stable and astable.

A Flip-Flop is a bi-stable multivibrator and therefore it has two stable states of output-either high or low. It is a device that can be triggered to high state or triggered to low state. Depending on its inputs and previous output, its new output is either high (or 1) or low (or 0). The applied inputs must be kept in the input terminals of the flip-flop so that the inputs can affect the output. Once the output is fixed, the inputs can be removed and then also

the already fixed output will be retained by the flip-flop. Hence a flip-flop can be used as a basic circuit of memory for storing one bit of data. To store multiple bits we can use multiple flip-flops. Such set of flip-flops are called register.

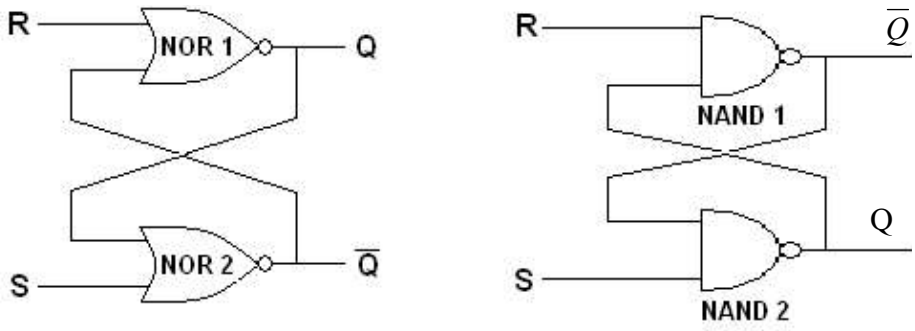
Flip-flop also serves as a fundamental component of another important digital circuit called counter. This device can keep track of events that occur in a digital circuit such as the number of pulses that a circuit can generate in a given time period. It can also divide frequency of a clock pulse which can lead to a digital clock circuit.

Flip-flops are constructed in a number of ways, each type having specific characteristics necessary for a particular application. Here we discuss the following types of flip-flops :

- RS Flip-flop
- D Flip-Flop
- JK Flip-Flop
- MS Flip-Flop

14.4.1 RS Flip-Flop

RS Flip-Flop is also called Set-Reset Flip-Flop. An RS flip-flop can be built using many different circuits. Here, we have shown two RS flip-flop circuits using NOR and NAND gate also called the basic flip flop. To study its working principle, we can take any one of them though their working principle differs slightly. Let us consider the RS flip-flop using NOR gate. Note that the NOR gates are cross-coupled, that is one of the inputs of each gate is the output of the other gate.



(a) **(b)**

Figure 14.1 : Basic circuit of RS Flip-Flop using NOR and NAND gate

RS flip-flop has two inputs—Set(S) and Reset(R). It has two outputs, Q and $\overline{Q}$. It should be noted $\overline{Q}$ is always the complement of Q. Though both outputs are not used in all the applications, the availability of both makes the flip-flop more versatile. Various combinations of inputs and their corresponding outputs are listed in the truth table below :

R	S	Q	Action
0	0	last value	No change
0	1	1	set
1	0	0	Reset
1	1	?	Forbidden

Truth Table of RS Flip-Flop

To understand the operation of the flip-flop let us assume that the flip-flop is initially *SET*, that means Q = 1 and $\overline{Q}$ = 0. Now the first input condition in the table i.e. R = 0, S = 0, will not change the state of the flip-flop. We can verify that by observing the fact that the input to the NOR gate 2 are 0 and 1, which keeps $\overline{Q}$ at 0. Since $\overline{Q}$ is 0, the input to the NOR gate 1 are both 0, making Q = 1. So we can say R = S = 0 has no effect on its output, the flip-flop retains its previous state. Hence Q remains unchanged.

The second input condition R = 0, S = 1 makes no change. The output of gate 2 remains low when both inputs are made high. Thus a 1 at the S input and 0 at R input will SET the flip-flop and Q will be equal to 1.

The third input condition R = 1, S = 0 will reset the flip-flop. The high

input to gate 1 causes its output Q to change from 1 to 0. That, 0 is coupled to gate 2, which now has two low inputs. Therefore its output changes from 0 to 1, that 1 is coupled to gate 1 and therefore Q is held at 0. Hence, when R = 1, S = 0, then Q = 0, R = 1. Thus the flip-flop is RESET.

The last input condition in the table R = 1, S = 1 is forbidden since it forces both the NOR gates to the low state, means both Q = 0, and R = 0 at the same time, which violets the basic definition of flip-flop that requires Q to be the complement of $\overline{Q}$. Hence this input condition is forbidden and its output is unpredictable.

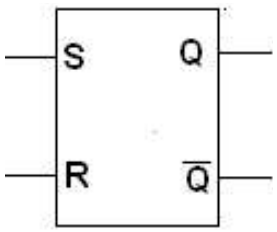
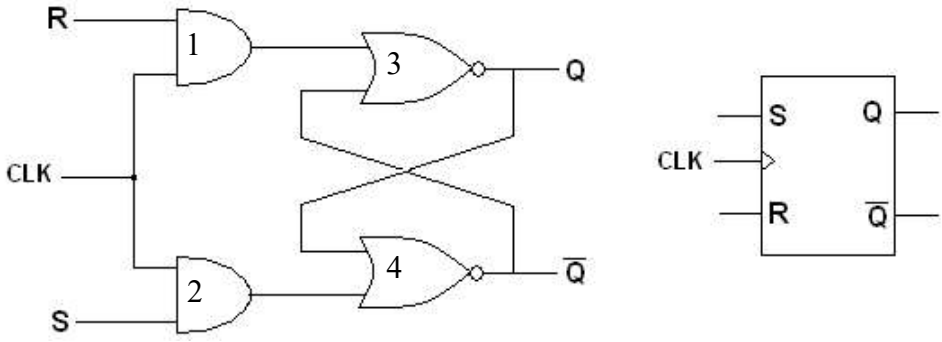


Figure 14.2 : Symbol of RS Flip-Flop

CLOCK INPUT : For synchronization of operation of multiple flip-flop, an additional signal is added to all types of flip-flop which is called clock signal, generally abbreviated as CLK. Addition of CLK signal ensures that, what ever may be the input to the flip-flop, it effects the output only when CLK signal is given. Figure 14.3 shows a clocked RS flip-flop.



(a) Clocked RS Flip-Flop **(b)Symbol of Clocked RS FF**

Figure 14.3 : RS flip-flop

SET-UP TIME : When the clock is low, the changes in R and S inputs does not effect the state of a flip-flop since the output of gate 1 and gate 2 (i.e. R and S input to the basic flip-flop) are zero or low. To effect the output, the R and S input must remain constant for a minimum time before a clock

pulse is applied. This minimum time is called *set-up time*. Its value is given in manufacturers' specifications.

14.4.2 D FLIP-FLOP

D flip-flop is a modification of RS flip-flop. In RS flip-flop when both the inputs are high i.e. R = 1, S = 1, the output becomes unpredictable and this input combination is termed as forbidden. To avoid this situation, the RS flip-flop is modified so that both the inputs can not be same at a time. The modified flip-flop is called D flip-flop. Figure 14.4 shows a clocked D flip-flop.

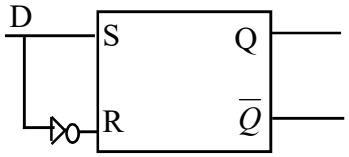


Figure 12.4 : (a) D flip-flop

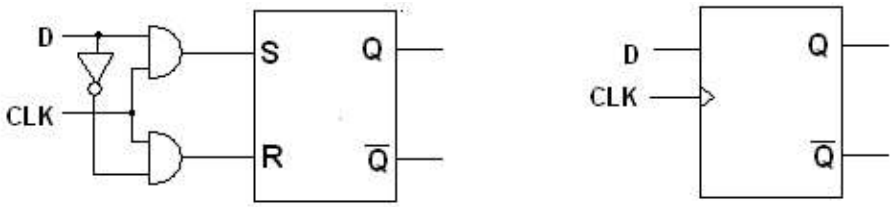


Figure 14.4 : (b) Clocked D Flip-Flop (c) Symbol of D Flip-Flop

In D flip-flop, both inputs of RS flip-flip are combined together to make it one by a NOT gate so that inputs can not be same at a time. Hence in D flip-flop there is only one input. The truth table is shown below:

CLK	D	Q
0	X	Last state
1	0	0
1	1	1

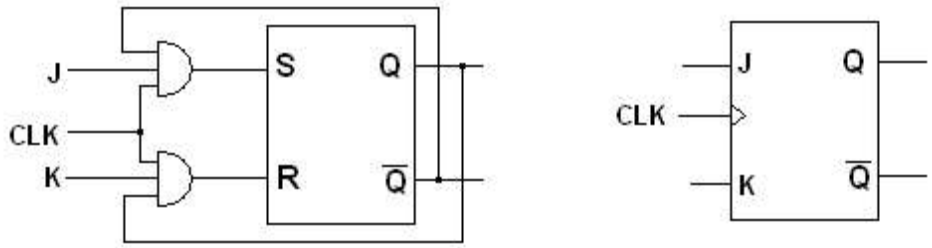
Truth Table of D F lip-Flop

In a clocked D flip-flop the value of D cannot reach the output Q when the clock pulse is low. During a low clock, both AND gates are disabled, therefore, D can change value without affecting the value of Q. On the

other hand, when the clock is high, both AND gates are enabled. In this situation, Q is forced to be equal to the value of D. In another way we can say that in the D flip-flop above, Q follows the value of D while the clock is high. This kind of D flip-flop is often called a D latch.

14.4.3 JK FLIP-FLOP

In RS flip-flop, the input R = S = 1 is called forbidden as it causes an unpredictable output. In JK flip-flop this condition is used by changing the RS flip-flop in some way. In JK flip-flop both input can be high simultaneously. Figure 14.5 shows one way to build a JK flip-flop.



(a) (b)

Figure 14.5 : (a) JK Flip-Flop, (b) Symbol of JK Flip-Flop

CLK	J	K	Q
X	0	0	Last state
↑	0	1	0
↑	1	0	1
↑	1	1	Toggle

Truth table for JK Flip-Flop



Propagation : Propagation delay is the time required to convert the inputs of a circuit to outputs

The inputs J and K are called control inputs because their combinations decide what will be the output of JK flip-flop when a +ve clock pulse arrives. When J and K are both low, both the AND gates are disabled. Therefore the CLK pulse has no effect. The first input combination of the truth table shows this and under this case the output Q retains its last state.

When J is low, K is high, the upper AND gate is disabled while the lower AND gate is enabled. Hence, the flip-flop cannot be set ; instead it is reset, i.e. Q = 0. This is shown by the second entry in the truth table.

When J is high, K is low, the upper AND gate is enabled while the lower one is disabled. So the flip-flop is set, there by making Q = 1.

When J and K are both high, then the flip-flop is set or reset depending on the previous value of Q. This is implemented by the feedback connections from $\overline{Q}$ to the upper AND gate and Q to the lower AND gate, as shown in Figure 14.5(a). If Q is high previously, the lower AND gate sends a RESET trigger to the flip-flop on the next clock pulse. Then Q becomes equal to 0. On the other hand if Q is low previously, the upper AND gate sends a SET trigger on the flip-flop making Q = 1.

So, when J = K = 1, Q changes its value from 0 to 1 or 1 to 0 on the positive clock pulse. This changing of Q to $\overline{Q}$ is called toggle. Toggle means to change to the opposite state.

Any flip-flop may be driven by +ve as well as –ve clock. As such JK flip-flop can also be driven by positive clock as well as negative clock. Figure 14.6 shows symbol of positive clocked and negative clocked JK flip-flop.

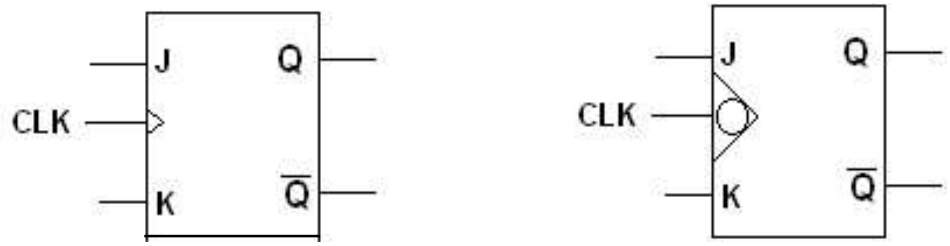
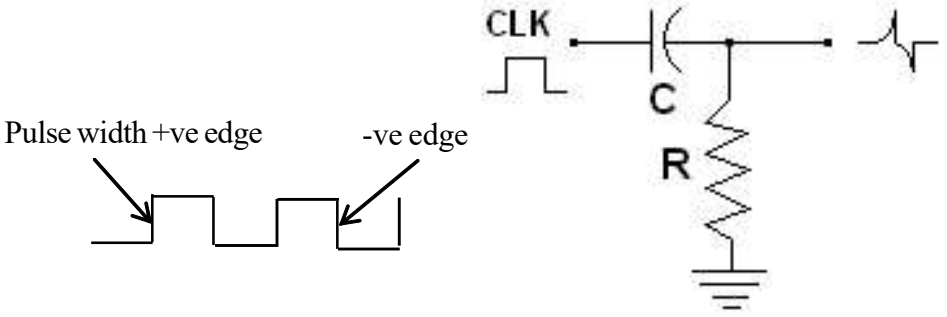


Figure 14.6 : Positive clocked JK Flip-Flop Negative clocked JK Flip-Flop

RC Differentiator Circuit : The clock pulse applied to a flip-flop is a square wave signal. Clock pulse is used to achieve synchronization of different flip-flops operation. The time span of the square wave is a longer time in comparison to the response time of flip-flops. Therefore, the time span of the clock pulse is reduced by converting its shape to a narrow spike, so that the flip-flops can change their states at an instant rather than transiting their states one after another in a lethargic manner. It makes the synchronization more precise. The square wave pulse is modified to a narrow spike by using a RC differentiator circuit as shown in Figure 14.7



(a) Clock pulse (b) RC Differentiator Circuit

Figure 14.7 : Clock pulse and Circuit

Racing: Racing means multiple transition of states of a flip flop for the same input. It results because of the feedback connections in the JK flip flops. It can be avoided by making the flip flop edge triggered sensitive or using MS flip flop.

The upper tip of the differentiated pulse is called positive edge and the lower tip is called is negative edge. When a flip-flop is triggered by this type of narrow spike, it is called edge triggered flip-flop. If the flip-flop is driven by +ve edge, it is called +ve edge triggered flip-flop. If it is driven by negative edge, it is called negative edge triggered flip-flop.

Racing : In a flip-flop if the output toggles more than ones without changing input during a clock edge than it is called racing. All flip-flop have a propagation delay, means the output changes its state after a certain time period from applying the input and the clock pulse. So, when a flip-flop is edge triggered, then due to propagation delay the output cannot affect the input again, because by that time the edge of clock pulse has already passed away. If the propagation delay of a flip-flop is 20 ns and the width of the spike is less than 20 ns, then the returning Q and $\overline{Q}$ arrive too late to cause false triggering.

14.4.4 Master Slave Flip-Flop

MS flip-flop is another way to avoid racing. Figure 14.8 shows one way to build MS flip-flop using two JK flip-flop to one of which clock pulse is applied directly and to the other applied through an inverter. The first JK flip-flop is called master and the later is called slave. The master responds to its J and K inputs at the positive pulse and the slave responds to the negative clock pulse. When J=1, K=0, the master sets on the positive clock pulse. The high Q output of the master drive the J input of the slave. So, at the negative clock pulse, the slave also sets, copying the action of the master.

When J=0, K=1, master resets at +ve clock pulse and the slave resets at the –ve clock pulse.

When J=K=1 master toggles at +ve clock pulse, and the slave toggles at the –ve clock pulse.

Hence whatever master does, the slave copies it.

MS flip-flop is a very popular flip-flop in industry due to its inherent resistance to racing. Hence it is extensively used to build counter.

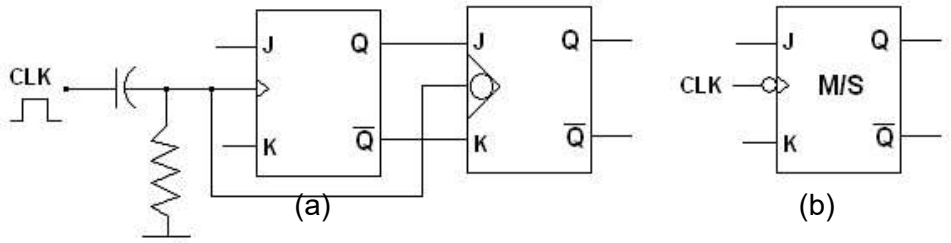
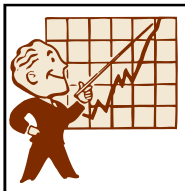


Figure 14.8 (a) Edge triggered JK MS Flip-Flop,
(b) Symbol of JK MS Flip-Flop



CHECK YOUR PROGRESS

Q.1. A flip-flop is basically a
- (a) mono-stable multi-vibrator
- (b) astable multi-vibrator
- (c) bi-stable multi-vibrator
- (d) none of these

Q.2. JK flip-flop has the specialty in
- (a) fast response time
- (b) toggle property
- (c) spike shaped clock input
- (d) preset input

Q.3. In MS flip-flop the master changes state
- (a) after the slave
- (b) with the slave at the same time
- (c) before the slave
- (d) never

Q.4. In edge triggering the clock pulse used is
- (a) Square wave
- (b) Triangular wave
- (c) Narrow spike
- (d) None



14.5 LET US SUM UP

- Digital circuits are of two categories - combinational and sequential
- A sequential circuit's output depends on past output and present inputs.
- A flip-flop is basically a single cell of memory which can store either 1 or 0.
- Sequential circuits use flip-flop as their building block.
- There are many types of flip-flop viz RS, D, JK, MS flip-flop.



14.6 FURTHER READING

- Mano, M. M. (2017), *Digital Logic and Computer Design*, Peason Education India.
- Mano, M. M. (1982), *Computer System Architecture*, Englewood Oliffs, N.J., 53-54.
- Malvino, A.P. & Leach, P. (1986), *Digital Principles and Applications*, Mcgraw-Hill International
- Lee, C. (1976), *Digital Circuits and Logic Design*
- Talukdar P. (2010), *Digital Techniques*, N. L. Publications
- Bogart, F. (1992), *Introduction to Digital Circuits*.



14.7 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : (c)

Ans. to Q. No. 2 : (b)

Ans. to Q. No. 3 : (c)

Ans. to Q. No. 4 : (c)



14.8 MODEL QUESTIONS

Q.1. Distinguish between combinational circuit and sequential circuit.

Q.2. Define flip-flop. Draw an RS flip-flop using NOR gate and explain its operating principle with truth table.

Q.3. What modification is done in RS flip-flop to construct D flip-flop? What is achieved after the modification ?

Q.4. Explain the operation of MS flip-flop with its symbol and truth table.

Q.5. What is the advantage of JK flip-flop over an RS flip-flop ? Write the working principle of a JK flip-flop.

Q.6. Why a square wave clock pulse is converted to a narrow spike ? How it is done ?

Q.7. What is the drawback of JK flip-flop? How it is removed?

****

UNIT 15 : REGISTERS

UNIT STRUCTURE

15.1 Learning Objectives

15.2 Introduction

15.3 Registers

15.4 Serial In—Serial Out(SISO) Register

15.5 Serial In –Parallel Out (SIPO) Register

15.6 Parallel In –Serial Out(PISO) Register

15.7 Parallel In –Parallel Out(PIPO) Register

15.8 Let Us Sum Up

15.9 Further Reading

15.10 Answers to Check Your Progress

15.11 Model Questions

15.1 LEARNING OBJECTIVES

After going through this unit, you would be able to :

- define registers
- describe the working principle of registers
- describe different registers like Serial In - Serial Out, Serial In - Parallel Out, Parallel In - Serial Out, Parallel In - Parallel Out.

15.2 INTRODUCTION

In the previous units, we have learnt about sequential and combinational circuits. In this unit, we will learn about registers. Different types of registers like SISO, SIPO, PISO and PIPO are covered in detail in this unit.

15.3 REGISTERS

A register is a set of flip-flops used to store binary data. The data to be stored is entered or shifted into the register and also taken out or shifted out as per necessity. Hence registers with the facility to shift their contents

right or left or in serial or parallel are known as shift registers.

Registers are used to store data temporarily. Registers can be used to perform some important arithmetic operations like complementation, multiplication, division etc. It can be connected to form counters, to convert serial data to parallel and parallel to serial data.

Types of registers: According to shifting of binary number different types of registers are:

- Serial In—Serial Out(SISO)
- Serial In –Parallel Out (SIPO)
- Parallel In –Serial Out(PISO)
- Parallel In –Parallel Out(PIPO)

15.3 SERIAL IN – SERIAL OUT (SISO) REGISTER

Figure 15.1 shows a typical 4 bit Serial In – Serial Out register using flip-flops. Here the content of the register is named as QRST. Let us consider that all flip-flops are initially reset. Hence at the beginning QRST = 0000. Let us consider a binary number 1011 we want to store in the SISO register.
At time A : A 1 is applied at the J input and 0 at the K input of the first flip-flop. At the negative edge of the CLK pulse, this 1 is shifted into Q. The O of Q is shifted into R, O of R is shifted into S and O of S is shifted into T. The output of the register just after time A is QRST = 1000.
(During the input/output operation of the register, whatever we give to the J input, we must give complement of it to the K input.)

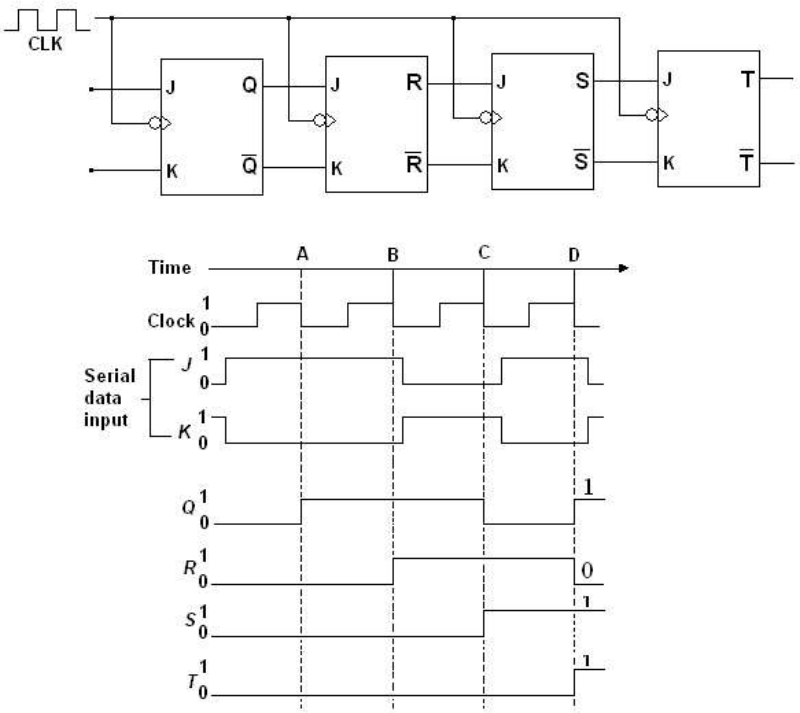


Figure 15.1 : 4 Bit Serial In – Serial Out Shift Register

At time B : Another 1 and 0 is applied in the J & K input of the first flip-flop. So at the negative CLK edge, the 1 is shifted to Q. The 1 of Q is shifted in R,0 of R is shifted in S, 0 of S is shifted into T. So, at the end of time B the output of all the flip-flops are QRST=1100.

At time C : A 0(zero) is applied in the J input and 1 at the K input of the 1st first flip-flop. At the negative CLK edge, the O shifts to Q. The 1 of Q shifts into R,1 of R shifts into S, O of S shifts into T. Hence the output becomes QRST=0110.

At time D : 1 is applied to J input and 0 at the K input of the first flip-flop. So after the negative clock pulse the 1 shifts into Q. The previous O of Q shifts into R, the 1 of R shifts into S, the 1 of S shifts into T. Hence at the end of time D, the registers contains QRST=1011.

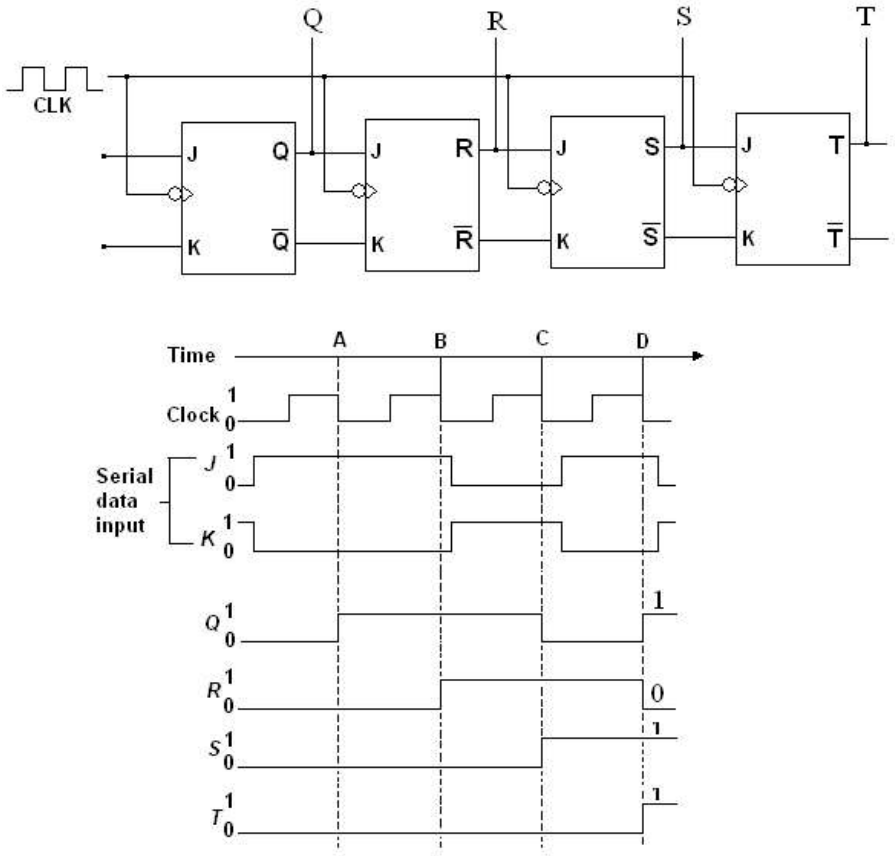
In the above steps, using 4 CLK pulses, we have shifted a 4 bit binary number 1011 in the register in a serial fashion.

To take out this binary number serially, we need another 4 CLK pulses and four O inputs into J pin and four 1 into K pin of the first flip-flop. The binary number leave the register serially through the T pin of the last flip-flop.

15.5 SERIAL IN—PARALLEL OUT (SIPO) REGISTER

In this type of shifts register, data is entered serially into the register and once data entry is completed it can be taken out parallely. To take the data parallely, it is simply required, to connect the output of each flip-flop to an output pin. All other constructional features are same as in Serial In—Serial Out (SISO) register.

The shifting of data into SIPO is same as SISO registers. In the SIPO of Figure 15.2 , a binary number say, 1011 would be shifted just like the manner as described in the previous section. It would take 4 CLK pulses to complete the shifting. As soon as shifting is completed, the stored binary number becomes available in the output pins QRST. SIPO register is useful to convert serial data into parallel data.


Q
R
S
T
CLK
J
Q
K
Q̄
J
R
K
R̄
J
S
K
S̄
J
T
K
T̄
Time
A
B
C
D
Clock
Serial data input
J
K
Q
R
S
T
1
0
1
0
1
0

15.6 PARALLEL IN-SERIAL OUT (PISO) REGISTER

PISO registers take data parallely and shift data serially. Commercially available TTL IC for PISO is 54/74166. To understand the functional block diagram of 54/74166, we should first understand the Figure 15.3.

Figure 15.3 (a) is a clocked RS flip-flop, which is converted to D flip-flop by a NOT gate. The output of the flip-flop is 1 if Data IN ($\overline{X}$) is 0. Next we add a NOR gate as in Figure 13.3(b). Here, if X_2 is at ground level, X_1 will be inverted by the NOR gate. As for example, if =1, then output of the NOR gate will be =0, thereby a 1 will be clocked into the flip-flop.

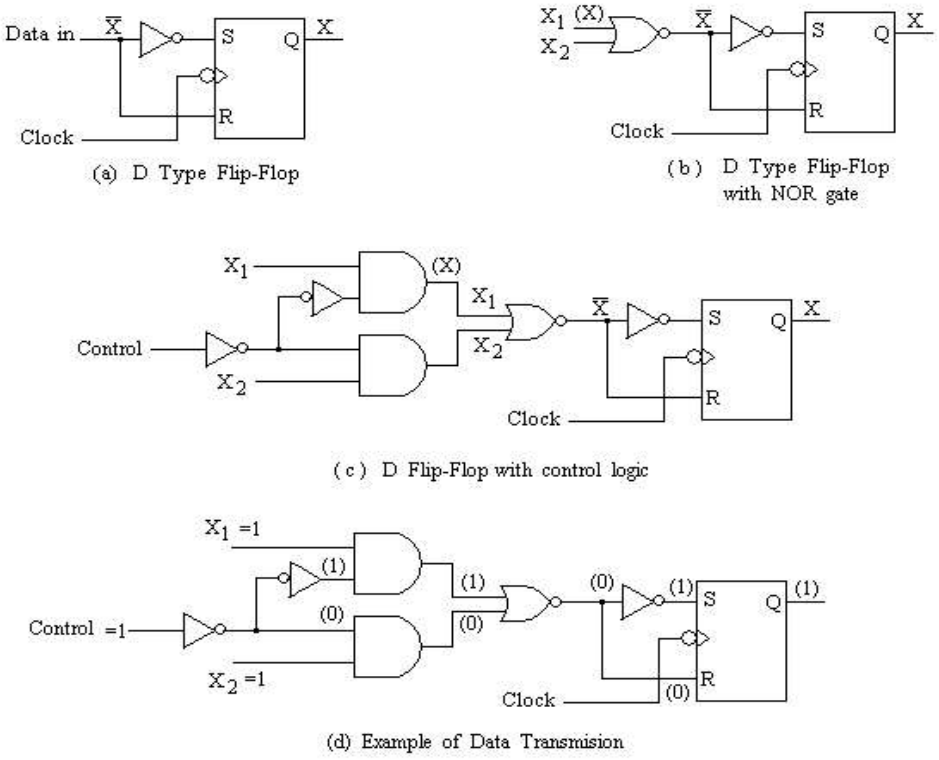
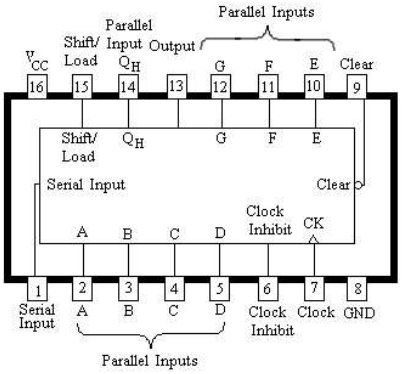


Figure 15.3 : Building Block of Parallel In- Serial Out Register

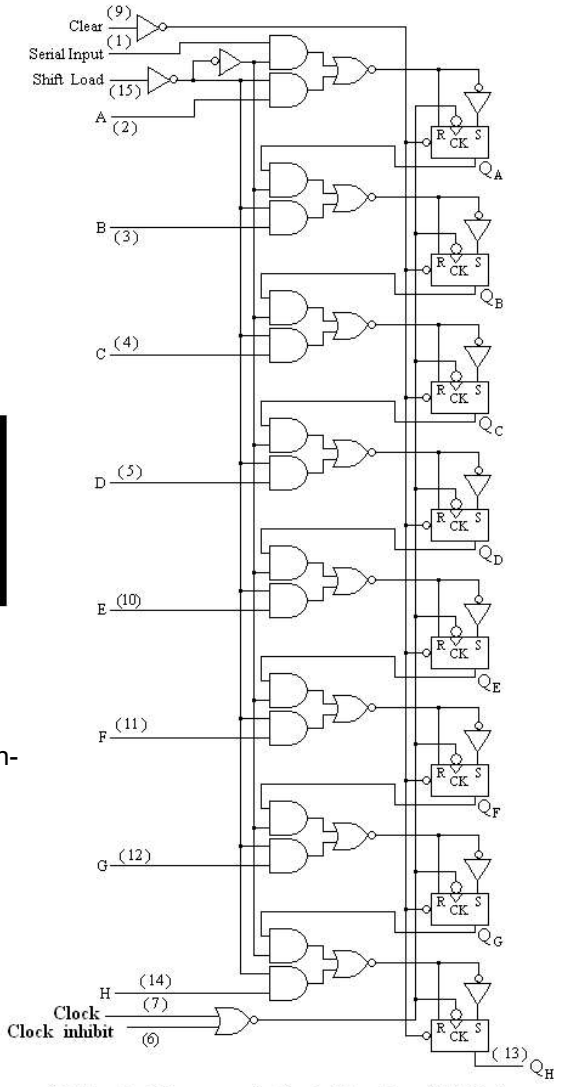
Now in Figure 15.3(c), two AND gates and two NOT gates are added. These will allow the selection of data . If the control line is high, the upper AND gate is enabled and the lower AND gate is disabled. Thus the data will enter at the upper leg of the NOR gate and at the same time the lower leg of the NOR gate is kept at ground. Opposite to this, if the control is low, the

upper AND gate is disabled and the lower AND gate is enabled.

If we study the Figure 13.4 of PISO, we see that circuit of Figure 15.3 (c) is repeated 8 times to form the 54/74166 shift register. These 8 circuits are connected in such a style that it allows two operations: (1) The parallel data entry and (2) shifting of data serially through the flip-flop from Q_A toward Q_B.



(a) Pin Out Diagram of Serial In-Parallel Out Register



(b) Logic Diagram of Serial In-Parallel Out Register

Figure 15.4 : Circuit of 54/74166

In Figure 15.4, the X_2 input of Figure 15.3(c) is taken out from each flip-flop to form 8 inputs named as ABCDEFGH to enter 8 bit data parallely to the register. The control is named here as SHIFT/LOAD which is kept low to load 8 bit data into the flip-flops with a single clock pulse parallely. If the SHIFT/LOAD is kept high it will enable the upper AND gate for each flip-flop. If any input is given to this upper AND gate then a clock pulse will shift a data bit from one flip-flop to the next flip-flop. This means data will be shifted serially.

15.7 PARALLEL IN –PARALLEL OUT (PIPO) REGISTER

The register of Figure 15.4 can be converted to PIPO register simply by adding an output line from each flip-flop.

The 54/74198 is an 8 bits such PIPO and 54/7495A is a 4 bit PIPO register. Here, the basic circuit is same as Figure 15.3(c). The parallel data outputs are simply taken out from the Q sides of each flip-flop. In Figure 15.5 the internal structure of 54/7495A is shown.

When the MODE CONTROL line is high, the data bits ABCD will be loaded into the register parallely at the negative clock pulse. At the same time the output is available at $Q_A Q_B Q_C Q_D$. When the MODE CONTROL is low, then the left AND gate of the NOR gate is enabled. Under this situation, data can be entered to the register serially through SERIAL INPUT. In each negative transition, a data bit shifted serially from Q_A to Q_B, from Q_B to Q_C and so on. This operation is called right-shift operation.

With a little modification of the connection, the same circuit can be used for shift-left operation. To operate in shift-left mode, the input data is to be entered through D input pin. It is also necessary to connect Q_D to C, Q_C to B, Q_B to A. MODE CONTROL line is high to enter data through the D input pin and each stored data bits of flip-flops will be shifted to left flip-flop on each negative clock transition. This is serial data and left shift operation.

Two clock inputs—clock1 and clock2 is used here to perform shift right and shift left operation.

Hence 54/7495A can be used as Parallel In –Parallel Out shift register as well as shift right and shift left register.

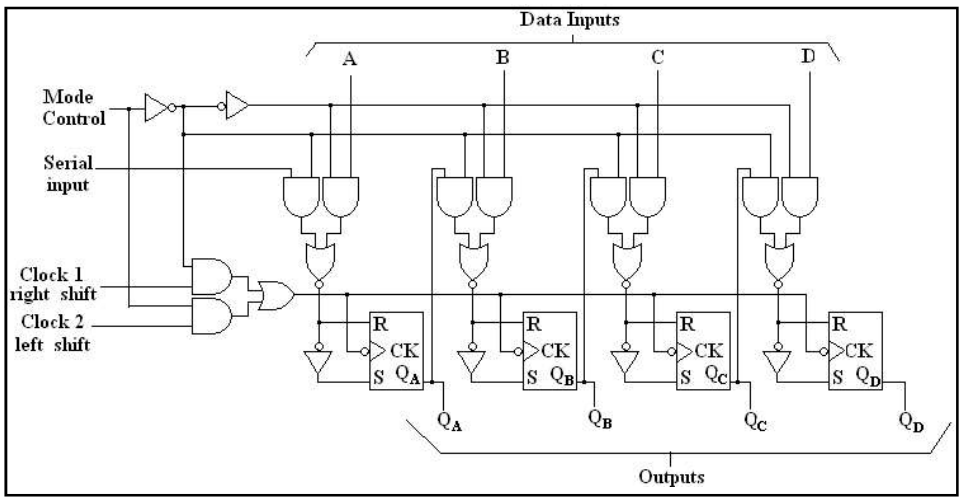


Figure 15.5 : Parallel In- Parallel Out Shift Register



CHECK YOUR PROGRESS

Q.1. Shift registers are.....

(a) basically a sequential circuit (b) a combinational circuit

(c) permanent memory (d) none of these

Q.2. In SIPO

(a) data enters parallely and leaves serially

(b) data enters serially and leaves serially

(c) data enters serially and leaves parallely

(d) data enters parallely and leaves parallely



15.8 LET US SUM UP

- A register is a set of flip flop used to store binary data.
- Shift registers are sequential circuit which are used to store binary bits.

- They are of four different types - Serial In- Serial Out, Serial In- Parallel Out, Parallel In- Parallel Out and Parallel In-Serial Out register.
- PISO registers take data parallely and shift data serially.
- In SIPO registers, data is entered serially and taken out parallely.



15.9 FURTHER READING

- Mano, M. M. (2017), *Digital Logic and Computer Design*, Peason Education India.
- Mano, M. M. (1982), *Computer System Architecture*, Englewood Oliffs, N.J., 53-54.
- Malvino, A.P. & Leach, P. (1986), *Digital Principles and Applications*, Mcgraw-Hill International
- Lee, C. (1976), *Digital Circuits and Logic Design*
- Talukdar, P. (2010), *Digital Techniques*, N. L. Publications
- Bogart, F. (1992), *Introduction to Digital Circuits*.



15.10 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : (a)

Ans. to Q. No. 2 : (c)



15.11 MODEL QUESTIONS

Q 1. What is register? Explain different types of register.

Q.2. Draw logic diagram with output wave form of a 4-bit Serial In-Parallel Out shift register for an input of 1101. Explain its operation.

Q.3. What are Shift registers?

Q.4. Describe the working of SISO register.

Q.5. Compare SISO register with PISO registers.

Q.6. Compare PIPO register with PISO registers.

Q.7. Compare SISO register with SIPO registers.

****

UNIT 16 : COUNTERS

UNIT STRUCTURE

16.1 Learning Objectives
16.2 Introduction
16.3 Counters
16.4 Asynchronous Counter
16.5 Synchronous Counter
16.6 Let Us Sum Up
16.7 Further Reading
16.8 Answers to Check Your Progress
16.9 Model Questions

16.1 LEARNING OBJECTIVES

After going through this unit, you would be able to :

- define and describe the working of counters
- describe the different types of counters
- describe asynchronous counter
- describe synchronous counter.

16.2 INTRODUCTION

In the previous units we have learnt about different sequential and combinational circuits. We have also learnt about the different types of registers like SIPO, PIPO, SISO and PISO.

In this unit, you will be introduced to the concept of counters. Different types of counters like asynchronous and synchronous counters are discussed in this unit.

16.3 COUNTERS

A counter is a sequential circuit that goes through a prescribed sequence of states when input pulses, also called count pulses are applied. These pulses may be clocked pulses or they may originate from an external

source. They may also occur at prescribed intervals of time or at random. A counter may follow the sequence of binary count or any other sequence of states. Counters are used for counting the number of occurences of an event and are useful for generating timing sequences to control operations in a digital system.

A counter can also be used as a frequency divider circuit. In this application, a counter can reduce the frequency of a clock pulse.

There are basically two types of counter—Synchronous counter and asynchronous counter.

Counter are constructed by using flip-flops and other logic gates. If the flip-flops are connected serially then output of one flip-flop is applied as input to the next flip-flop. Therefore this type of counter have cumulative settings time due to propagation delay. Counter of this type is called serial or asynchronous counter. These counters have speed limitation.

Speed can be increased by using parallel or synchronous counter. Here, flip-flops are triggered by a clock at a time and thus setting time is equal to the propagation delay of a single flip-flop. But this type of synchronous counters require more hardware and hence these are costly.

Combination of serial and parallel counter is also done to get an optimize solution of speed and hardware cost. If each clock pulse advances the count of the counter by one, it is called **up counter**. If the count of the counter goes down at each clock pulse, it is called **down counter**.

Before operation, sometimes it is required to reset all the flip-flops to zero. It is called “Clear”. Sometimes, it is required to set the flip-flops before hand. It is called preset. To do this, two extra inputs are there in every flip-flop called CLR and PR.

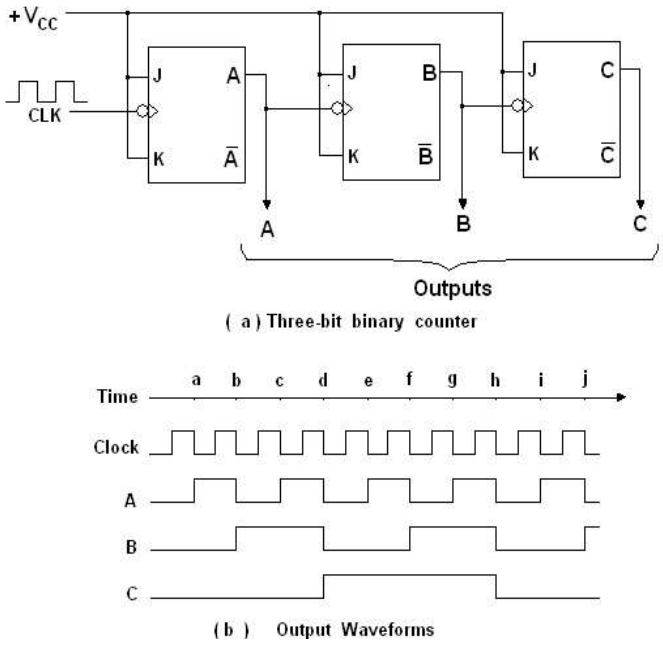
16.4 ASYNCHRONOUS COUNTER

When the output of a flip-flop is used as the clock input for the next flip-flop it is called asynchronous counter.

Asynchronous counters are also called ripple counter because flip-flop transitions ripple through from one flip-flop to the next in sequence until all flip-flop reach a new state.

A binary ripple counter can be constructed by using clocked JK flip-flop. Figure 16.1(a) shows three MS JK flip-flops connected in series. The clock drives flip-flop A. The output of A drives B and the output of B drives C. J and K inputs of all the flip-flops are connected to positive to make them equal to 1. Under this condition each flip-flop will change state (toggle) with a negative transition at its clock point.

In the counter shown in Fig 16.1 (a), the flip-flop A changes its state at the negative edges of the clock pulses. Its output is applied to the B flip-flop as its clock input.



Clock transitions	C	B	A
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
0	0	0	0

(c) Truth Table

Figure 16.1 : Three Bit Binary Counter

The output of B flip-flop toggles at the negative edges of the output of A flip-flop. Similarly, output of B flip-flop is used as clock input to the C flip-flop and therefore C toggles at the negative edges of the output of B flip-flop. We can see that triggering pulses move through the flip-flops like a ripple in water.

The wave form of the ripple counter is shown in Figure 16.1 (b). It shows the action of the counter as the clock runs. To understand the wave form, let us assume that the counter is cleared before the operation. The A output is assumed the list significant bit(LSB) and C is the most-significant-

bit (MSB). Hence at very beginning the contents of the counter is CBA=000.

Flip-flop A changes its state to 1 after the negative clock pulse transition. Thus at point 'a' on the time line, A goes high. At point 'b' it goes low, at 'c' it goes back to high and so on.

Now output of A acts as clock input of B. So, each time the output of A goes low, flip-flop B will toggles. Thus at point 'b' on the time line, B goes high, at point 'd' it goes low, and toggles back high again at point 'f' and so on.

Since B acts as the clock input for C, each time the output of B goes low, the C flip-flop toggles. Thus C goes high at point 'd' on the time line, it goes back to low again at point 'h'.

We can see that the output wave form of A has half the frequency then the clock input wave. B has half the frequency than that of A and C has half the frequency than that of B.

We can further see that since the counter has 3 flip-flops cascaded together, it progresses through 000—— 001——010—011—100—101—110—111 as its CBA output. After CBA= 111, it starts the cycle again from CBA=000. One cycle from 000—111 takes 8 clock pulses, as it is evident from the wave form as well as from the truth table.

MOD of a counter : The number of input clock pulses that causes a counter to reset to its initial state is called the *modulus* of a counter. Therefore, the modulus of a counter means the total number of distinct states including zero that a counter can store.

For example, the three stage counter shown in figure 5.9 has modulus 8 which can be seen from the wave diagram and the truth table. The modulus is called MOD in short.

When the modulus of a counter is equal to 2^n, where n is the number of stages or flip-flops used, then the counter is called a natural counter. Hence, a three stage counter has MOD = 2^3 = 8, four stage counter has MOD = 2^4 =16 etc and therefore these are natural counters.

A counter can be forcefully reset to its initial state before it completes its natural count by applying appropriate feedback. These type of counters have modulus lesser than its natural count. As such we can have MOD 6, MOD 7, MOD 10 counters etc.

16.5 SYNCHRONOUS COUNTER

An asynchronous counter or ripple counter has limitation in its operating frequency. Each flip-flop has a delay time which is additive in asynchronous counter.

In synchronous counter the delay of asynchronous counter is overcome by the use of simultaneous applications of clock pulse to all the flip-flops. Hence in synchronous counter, the common clock pulse triggers all the flip-flops simultaneously and therefore the individual delay of flip-flop does not add together. This feature increase the speed of synchronous counter. The clock pulse applied can be counted by the output of the counter.

To build a synchronous counter, flip-flops and some additional logic gates are required. Figure 16.2 shows a three stage synchronous or parallel binary counter along with its output wave forms and truth table. Here, the J and K inputs of the first flip-flop is kept high and therefore the flip-flops toggle at the negative clock transition at its clock input. From figure we can see that the output of A is ANDed with count enable input to drive the 2^{nd} flip-flop and the outputs of A, B are ANDed with CLK to drive the third flip-flop. This logic configuration is often referred to as “steering logic” since the inputs J and K are steered to each individual flip-flop.

In the figure, the clock pulse is directly applied to all the flip-flops. J and K are both high for the 1st flip-flop, so the first flip-flop toggles state at the negative transition of the input clock pulses. This can be seen at points a,b,c,d,e,f,g,h,i on the time line.

The AND gate X is enabled when A is high, and it allows the inputs J and K=1 high to reach the 2^{nd} flip-flop. So the 2^{nd} flip-flop toggles with every other negative clock transition at points b, d, f and h on the time line.

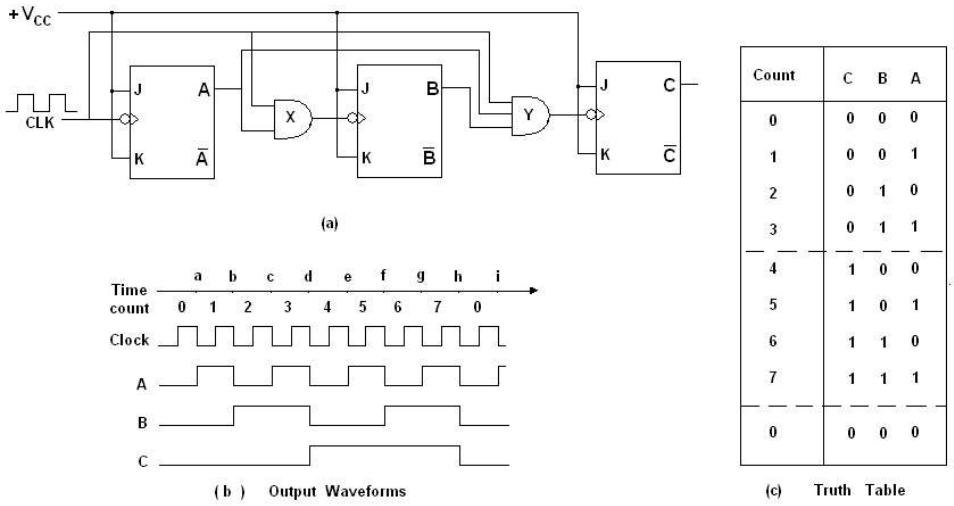


Figure 16.2 : 3 bit Parallel Binary Counter

The AND gate Y is enabled only when both A and B are high and it transmits the clock pulses to the clock input of the 3rd flip-flop. The 3rd flip-flop toggles state with every fourth negative clock transition at d and h on the time line.

The wave form and the truth table shows that the synchronous counter progresses upward in a natural binary sequence from 000 to 111. The total count from 000 to 111 is 8 and hence this counter can also be called MOD-8 counter, in count up mode.



CHECK YOUR PROGRESS

Q.1. State *true or false*

(a) Counters are non sequential digital circuits.

(b) Asynchronous counters are fast in operation than synchronous counters.

(c) The total number natural progression of a counter is called MODULUS.

(d) Counters can be used to build digital clock.

Q.2. Define Counters.



16.6 LET US SUM UP

- Modules of a counter is the number of clock pulses that cause a counter to reset its initial state.
- A counter is a sequential circuit that can count square waves given as clock input.
- There are two types of counters- asynchronous and synchronous counter.
- Asynchronous counters are also called ripple counters.
- Synchronous counters overcome the delay of asynchronous counters.



16.7 FURTHER READING

- Mano, M. M. (2017), *Digital Logic and Computer Design*, Peason Education India.
- Mano, M. M. (1982), *Computer System Architecture*, Englewood Oliffs, N.J., 53-54.
- Malvino, A.P. & Leach, P. (1986), *Digital Principles and Applications*, Mcgraw-Hill International
- Lee, C. (1976), *Digital Circuits and Logic Design*
- Talukdar P. (2010), *Digital Techniques*, N. L. Publications
- Bogart, F. (1992), *Introduction to Digital Circuits*.



16.8 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1 : (a) False, (b) False, (c) True, (d) True

Ans. to Q. No. 2 : A counter is a sequential circuit that goes through a perscribed sequence of states when input pulses are applied.



16.9 MODEL QUESTIONS

Q.1. What are the differences between asynchronous and synchronous counter? Draw a MODE-8 counter and explain its working principle.

Q.2. What do you mean by modulus of a counter? What is a forced counter?

Q.3. Describe asynchronous counters in detail.

Q.4. Describe synchronous counters in detail.

Q.5. What are ripple counters?

****