

BCBM202T

DIGITAL ELECTRONICS



DR. C.V. RAMAN UNIVERSITY

Kargi Road, Kota, Bilaspur, (C. G.),

Ph. 07753-253801, 07753-253872

E-mail: info@cvru.ac.in, Website: www.cvru.ac.in

BCBM202T
DIGITAL ELECTRONICS

Credit- 2

Subject Expert Team

Course Edited By:

Unit Written By:

Warning: All rights reserved, No part of this publication may be reproduced or transmitted or utilized or stored in any form or by any means now known or hereinafter invented, electronic, digital or mechanical, including photocopying, scanning, recording or by any information storage or retrieval system, without prior written permission from the publisher.

Edition: 2026

Publish by: DR. C.V. RAMAN UNIVERSITY

Kargi Road, Kota, Bilaspur, (C. G.),

Ph. 07753-253801, 07753-253872

STRUCTURE OF PROGRAM

Title of the Program	: Bachelor of Computer Application
Program Type	: Undergraduate
Program Credits	: 120 Credits / 160 Credits
Program Duration	: 3 Years / 4 Years
Eligibility Criteria	: 12th Pass

PROGRAMME OBJECTIVES

- To produce employable IT workforce, that will have sound knowledge of IT and business fundamentals that can be applied to develop and customize solutions for Small and Medium Enterprises (SME).
- To develop skilled manpower in the various areas of information technology like. Data base management, Software Development, Computer-Languages, Software engineering, Web based applications etc.
- To Work professionally with positive attitude as an individual or in multidisciplinary teams and communicate effectively.

PROGRAMME OUTCOMES

PO 1. Develop competence in basic technical subjects in computer applications like

Programming Languages, Data Structures, Databases, Operating Systems, Software Engineering.

PO 2. Identify, analyze, formulate and develop computer applications.

PO 3. Map real life scenarios to various theoretical optimal solutions.

PO 4. Provide simplest automated solutions to various legacy systems.

PO 5. Use modern computing tools and techniques with confidence.

PROGRAMME SPECIFIC OUTCOME

Upon completion of program in Bachelor of Computer application, students will be able:

PSO 1. Able to implement computer skill in the areas related to multimedia and website design, Data base, Hardware and networking.

PSO 2. Able to demonstrate basic knowledge in the areas such as, Software Engineering, Data communication and Networking, Data base management, Web Technology and Operating Systems for building IT applications.

PSO 3. Students will be able to understand, analyze and develop computer programs in the areas related to algorithm, system software, web design and networking for efficient design of computer-based system.

PSO 4. Apply standard software engineering practices and strategies in software project development using open-source programming environment to deliver a quality of product for business success.

PSO 5. Student will be able to know various issues, latest trends in technology development and thereby innovate new ideas and solutions to existing problems.

STRUCTURE OF COURSE

Title of the Program	:	Digital Electronics
Program Type	:	Core Course
Course Code	:	BCBM202T
Course Credits	:	06
Semester	:	02

COURSE LEARNING OBJECTIVES:

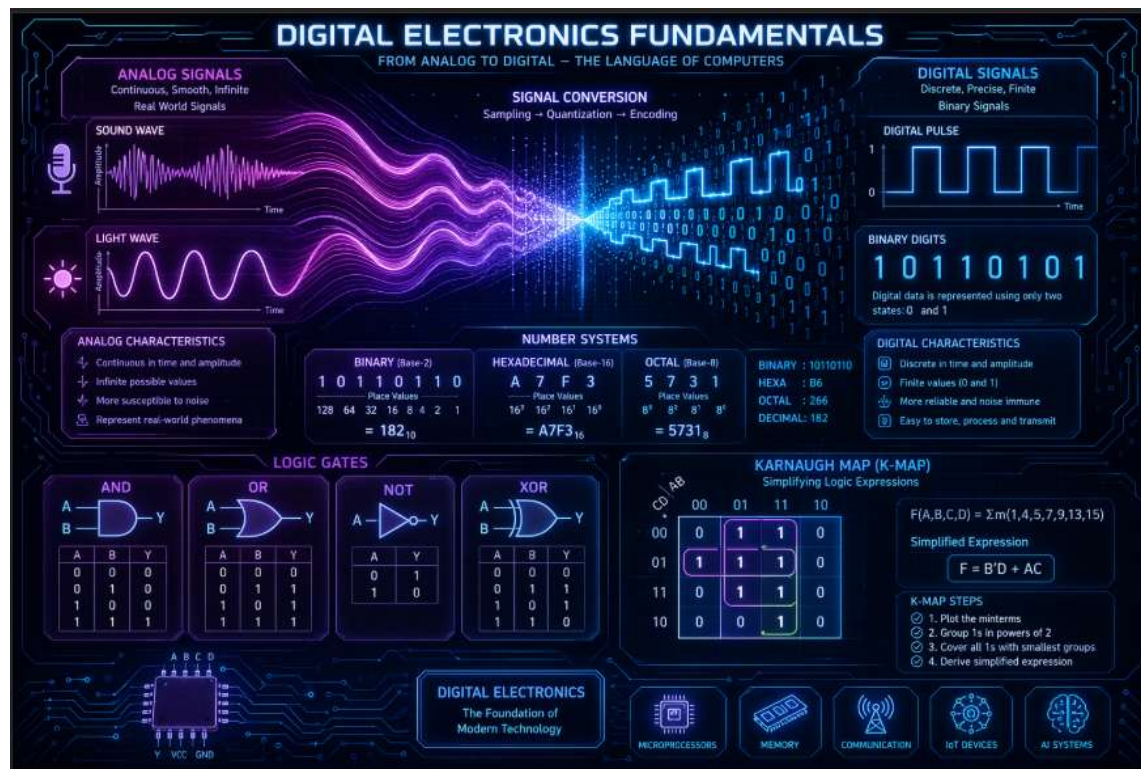
- Elaborate number representation and conversion between different number system.
- Build logic processes and implement logical operations using combinational logic circuits.
- Illustrating characteristics of memory and their classification.
- Correlating concepts of sequential circuits and to analyze sequential systems in terms of state machines.
- Organized memory working, uses and applications.

COURSE LEARNING OUTCOMES :

1. Define the basic concepts of excel.
2. Demonstrate the columns and rows formatting in excel.
3. Experiment with the formulas and data analysis in excel.
4. Determine the pivot table and chart concepts of excel.
5. Make the use of spreadsheets tools and macros in excel.

COURSE CONTENT		
Block - I	FUNDAMENTALS OF DIGITAL SYSTEMS	01 - 40
Unit 1	Analog and Digital Systems	03
Unit 2	Data Types and Number Systems	15
Unit 3	Logic Gates	25
Unit 4	Boolean Algebra and Simplification	33
Block II	COMBINATIONAL LOGIC DESIGN	43 - 84
Unit 5	Combinational Circuits : Adder and Subtractors	45
Unit 6	Combinational Circuits: Multiplexer and Demultiplexer	55
Unit 7	Encoders and Decoders	65
Unit 8	Digital Control Circuits and Sequencing	73
Block III	SEQUENTIAL LOGIC AND STATE MACHINES	87 - 150
Unit 9	Sequential Logic and Basic Flip Flops	89
Unit 10	Advanced Flip-Flop Configurations and Triggering	105
Unit 11	Shift Registers	117
Unit 12	Counters and State Diagrams	131
Unit 13	State Machines	141
Block IV	MEMORY ARCHITECTURE AND CONCEPTS	153 - 182
Unit 14	Auxiliary Memory	155
Unit 15	Semiconductor Memory	165
Unit 16	Advanced Memory Concepts	175
Block V	ADVANCED SYSTEM DESIGN AND MICROPROCESSORS	187 - 254
Unit 17	Arithmetic and Logic Circuits Architecture	189
Unit 18	Digital Design with PLDs	197
Unit 19	Timing and Control Circuits	209
Unit 20	Microprocessor Basics	219
Unit 21	Digital System Design	229
Unit 22	Future Trends in Digital Electronics	241

Block I	FUNDAMENTALS OF DIGITAL SYSTEMS	01 - 40
Unit 1	Analog and Digital Systems	03
Unit 2	Data Types and Number Systems	15
Unit 3	Logic Gates	25
Unit 4	Boolean Algebra and Simplification	33



This foundational section establishes the crucial framework required to navigate the landscape of digital electronics. Our journey begins by contrasting the analog world—where signals like sound and light are continuous and variable—with the digital paradigm, where information is abstracted into precise, discrete binary pulses (1s and 0s). You will quickly learn how digital representation offers immunity to noise and enables robust computational power. We then move to the essential language of computers: number systems. This block will master your fluency in Binary, Hexadecimal, and Octal formats, demonstrating how they mathematically translate human decimal logic into machine binary states. The section concludes with the atomized building blocks of all logic design—Basic Logic Gates (AND, OR, NOT, XOR). You will analyze their physical truth tables and characteristic operations before applying Boolean Algebra to simplify complex equations. This is arguably the most vital step in engineering; by using mathematical minimization techniques like Karnaugh Maps (K-Maps), you will ensure your future designs are maximally efficient, using the absolute minimum number of logical transistors while maintaining full sequential functionality. Mastering this block is prerequisites for progression into complex circuit design.

UNIT – 1

ANALOG AND DIGITAL SYSTEMS

NOTES

UNIT STRUCTURE

- 1.1. Introduction**
- 1.2. Objective**
- 1.3. Analog and Digital Systems**
 - 1.3.1. Definition of Analog System**
 - 1.3.2. Definition of Digital System**
 - 1.3.3. Continuous vs Discrete Signals**
 - 1.3.4. Advantages of Digital Systems**
- 1.4. Components of a Digital System**
 - 1.4.1. Input Devices**
 - 1.4.2. Processing Unit**
 - 1.4.3. Output Devices**
 - 1.4.4. Storage Elements**
- 1.5. Summary**
- 1.6. Glossary**
- 1.7. Answers to check your progress**
- 1.8. Suggested readings - OER**
- 1.9. Reference**
- 1.10. Exercise**

1.1. INTRODUCTION

Digital electronics is built upon the idea that information can be represented using discrete values rather than continuous signals. Before learning how circuits perform calculations or how memory stores data, it is essential to understand how information itself is expressed in a digital environment. This unit introduces the fundamental concepts of digital systems and the methods used to represent data numerically.

In everyday life, we use the decimal number system without much thought. However, digital systems do not operate in base 10. They rely on binary representation, where information is encoded using only two states—0 and 1. These two states correspond to physical conditions such as ON/OFF, HIGH/LOW voltage, or TRUE/FALSE logic levels. Understanding this transition from human-friendly decimal

NOTES

representation to machine-friendly binary representation is the first step toward mastering digital electronics.

This unit begins by distinguishing between analog and digital systems. Analog systems process continuous signals, whereas digital systems operate using discrete levels. Recognizing this difference helps in understanding why digital systems are more reliable, noise-resistant, and suitable for computation.

The unit then explores data types in digital systems and introduces positional number systems. The concept of radix (base) is explained to clarify how numerical values depend on position and base. By studying positional notation, learners develop the ability to interpret how different number systems function structurally.

A clear understanding of these foundational ideas ensures that subsequent topics—such as binary arithmetic, coding techniques, and logic design—are approached with conceptual clarity. This unit therefore establishes the intellectual groundwork for the entire course in digital electronics.

1.2. OBJECTIVE

After studying this unit, you will be able to:

- Differentiate between analog and digital systems based on signal characteristics and applications.
- Classify different types of data used in digital systems.
- Explain the concept of positional number systems and radix.
- Illustrate how numerical value depends on position and base in a number system.
- Interpret the structural differences between decimal and binary representation.
- Analyze the importance of discrete representation in digital electronics.

1.3. ANALOG AND DIGITAL SYSTEMS

In order to understand digital electronics, it is essential to first distinguish between analog and digital systems. Every electronic system processes information in one of these two fundamental ways. The difference between them lies in how signals are represented and manipulated.

An analog system represents information using continuous signals, whereas a digital system represents information using discrete signals. This distinction is not merely theoretical; it determines how devices operate, how data is stored, and how reliable the system performance is.

Understanding this difference forms the conceptual bridge between traditional electronics and modern digital systems.

1.3.1. Definition of Analog System

An analog system is a system that represents and processes information in the form of continuously varying physical quantities. These quantities may include voltage, current, frequency, pressure, temperature, or sound waves.

In an analog system:

- Signals vary smoothly over time.
- Values can take any magnitude within a given range.
- Information is represented in a continuous waveform.

For example:

- A traditional mercury thermometer shows continuously varying temperature.
- A vinyl record player produces continuous sound waves.
- An analog clock represents time using continuously moving hands.

In graphical form, an analog signal appears as a smooth sine wave or curved waveform.

Key Characteristic: Infinite possible values within a range.

However, analog systems are more sensitive to noise and signal distortion because small external disturbances directly affect the signal.

1.3.2. Definition of Digital System

A digital system represents and processes information using discrete values. Instead of continuous variation, digital systems operate using fixed levels—typically two levels known as binary digits (0 and 1).

In digital systems :

- Signals change in discrete steps.
- Only predefined values are allowed.
- Information is encoded using binary representation.

Examples include:

- Computers and laptops
- Smartphones
- Digital watches
- Calculators

In a graphical representation, a digital signal appears as a square wave with distinct high and low levels.

Key Characteristic: Finite number of possible values (usually two).

Because digital systems operate using fixed levels, they are less affected by small noise variations. Minor disturbances do not change the logical state unless they cross the defined threshold.

NOTES

1.3.3. Continuous vs Discrete Signals

The fundamental difference between analog and digital systems lies in the nature of their signals.

Continuous Signal (Analog)

- Smooth waveform
- Infinite intermediate values
- Directly proportional to physical quantity

Discrete Signal (Digital)

- Step-like waveform
- Limited values (0 and 1)
- Threshold-based switching

Comparison table :

Feature	Analog Signal	Digital Signal
Nature	Continuous	Discrete
Values	Infinite	Finite (0 & 1)
Noise Resistance	Low	High
Storage	Difficult	Easy

Table 1.1 : Structural differences.

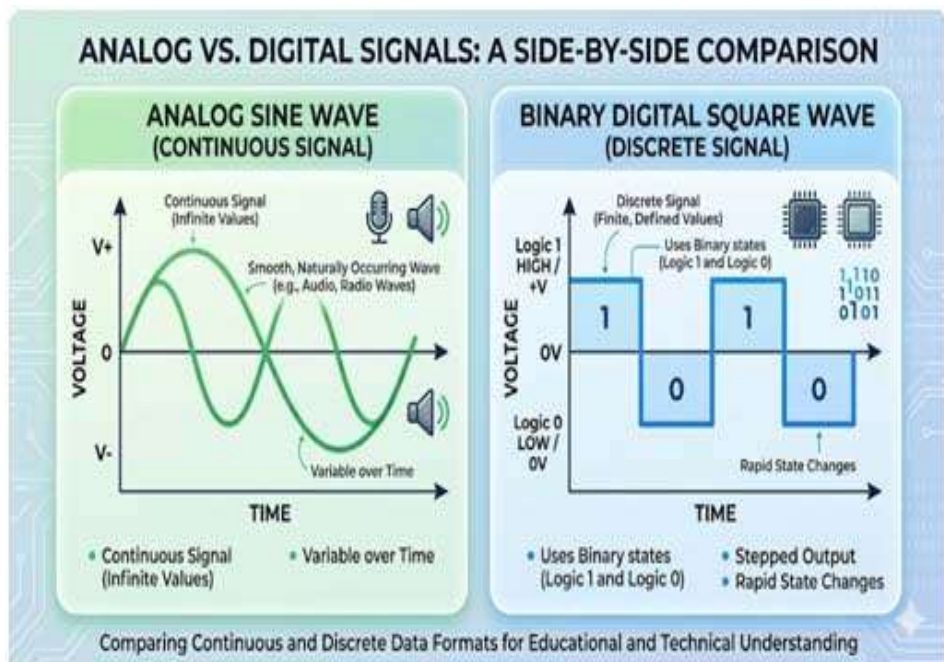


Fig. 1.1 : Analog vs. Digital Waveforms.

1.3.4. Advantages of Digital Systems

Digital systems have become dominant in modern technology due to several advantages :

1. High Noise Immunity

Small disturbances do not affect binary states unless thresholds are crossed.

2. **Reliable Storage**

Digital data can be stored and retrieved accurately without degradation.

3. **Ease of Processing**

Binary logic simplifies computation and automation.

4. **Programmability**

Digital systems can be programmed and reconfigured easily.

5. **Compact Design**

Integrated circuits allow millions of digital components in small chips.

6. **High Speed**

Digital circuits operate at extremely high switching speeds.

Because of these advantages, digital systems are widely used in computing, communication, control systems, and embedded technologies.

☒ **Check Your Progress 1.1.**

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. Define an analog system.

.....

2. State two differences between analog and digital signals.

.....

3. Why are digital systems more resistant to noise?

.....

4. Give two examples each of analog and digital systems.

.....

NOTES

1.4. COMPONENTS OF A DIGITAL SYSTEM

After understanding the difference between analog and digital systems, the next step is to examine how a digital system is structured. Every digital system, regardless of its size or complexity, follows a basic functional architecture. Whether it is a calculator, a computer, a smartphone, or an embedded controller, it performs three essential tasks: it accepts input, processes data, and produces output. In addition, it requires storage to retain information and instructions.

A digital system can therefore be broadly divided into four main components:

- Input Devices
- Processing Unit
- Output Devices
- Storage Elements

These components work together in a coordinated manner to perform meaningful operations.

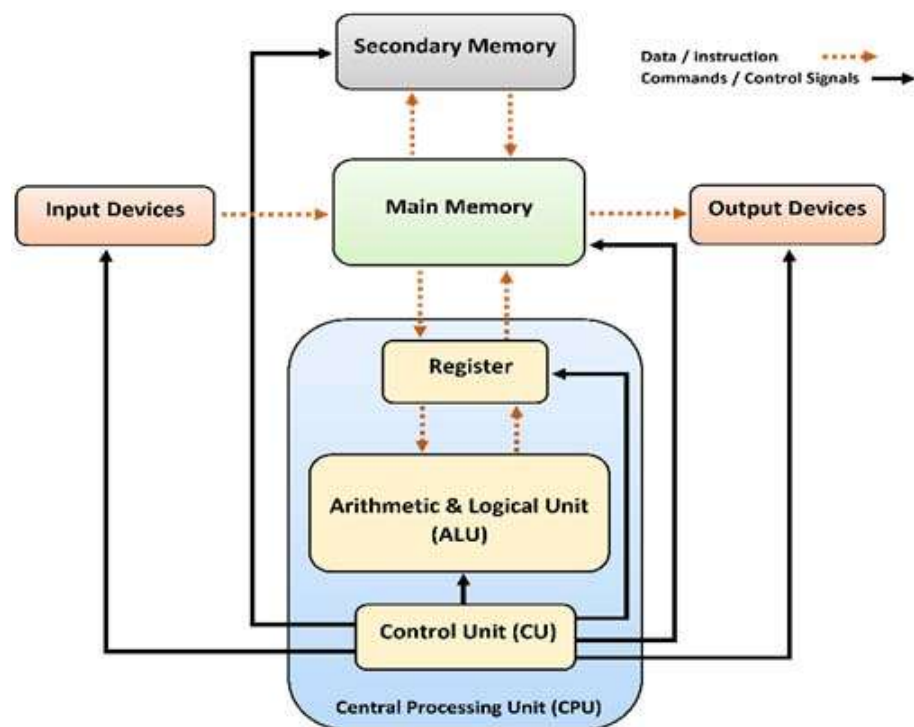


Fig. 1.1 : Computer architecture block diagram.

1.4.1. Input Devices

Input devices are used to provide data and instructions to the digital system. They act as the interface between the user and the machine.

Examples include:

- Keyboard
- Mouse
- Scanner

- Microphone
- Sensors

In digital electronics, input signals are often provided in the form of binary data or electrical pulses. Sensors in embedded systems convert physical quantities (temperature, pressure, motion) into electrical signals that can be processed digitally.

The primary function of input devices is to convert real-world information into a machine-readable format.

1.4.2. Processing Unit

The processing unit is the core component of a digital system. It performs logical and arithmetic operations on the input data.

In computer systems, this unit is known as the Central Processing Unit (CPU). It typically consists of:

- Arithmetic Logic Unit (ALU) – performs arithmetic and logical operations
- Control Unit (CU) – directs and coordinates operations
- Registers – temporary storage locations

The processing unit interprets instructions, performs calculations, and controls the overall operation of the system.

Without the processing unit, a digital system cannot transform input data into meaningful output.

1.4.3. Output Devices

Output devices present processed information to the user in a readable or usable form.

Examples include :

- Monitor
- Printer
- Speaker
- LED indicators
- Actuators in embedded systems

The output may be visual, audio, or mechanical depending on the application. In digital electronics, output signals are generated in binary form and then converted into human-understandable formats.

1.4.4. Storage Elements

Storage elements are responsible for retaining data and instructions. They allow the system to store information either temporarily or permanently.

Storage can be classified as:

1. **Primary Storage** – RAM, Registers (temporary storage)
2. **Secondary Storage** – Hard disk, SSD, Flash memory (permanent storage)

NOTES

In digital electronics, storage elements are implemented using memory devices composed of flip-flops and integrated circuits.

Storage plays a crucial role because processing requires both data and instructions to be available when needed.

 Check Your Progress 1.2.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. List the four main components of a digital system.

.....

.....

.....

2. What is the function of the processing unit?

.....

.....

.....

3. Differentiate between primary and secondary storage.

.....

.....

.....

4. Give two examples of input and output devices.

.....

.....

.....



Activity 1.1. ANALOG VS. DIGITAL IN YOUR PERSONAL SPACE

Title :

Observing Continuous and Discrete Signals at Home Objective: To observe and categorize everyday household objects based on whether they use continuous (analog) or discrete (digital) signals.

Material Needed :

A notebook, a pen, and your immediate surroundings.

Activity Description :

1. The Analog Hunt: Walk around your room and locate two devices that use a continuous scale or physical movement (e.g., a clock with hands or a traditional thermometer). Write them down.

2. The Digital Hunt: Locate two devices that display information using instant numerical steps (e.g., a microwave timer or smartphone). Write these down.
3. The Comparison: For one analog and one digital device, write a sentence describing exactly what happens when the value changes.

Reflection :

Ask yourself: Why have most appliances shifted to digital screens? You should notice that reading exact values on digital devices is faster and less prone to human error than guessing a needle's position on an analog dial. Digital states eliminate guesswork!

1.5. SUMMARY

In this unit, you have learned the fundamentals of number systems and data representation in digital electronics. The unit began by distinguishing analog and digital systems and introduced the binary, octal, decimal, and hexadecimal number systems, explaining how they are interconverted.

You studied the various techniques for representing numbers, including 1's complement, 2's complement, and binary fixed-point formats. Arithmetic operations and error conditions such as overflow and underflow were explored, providing insight into how digital circuits manage computation errors.

The unit also covered the IEEE 754 floating point standard, a key format for representing very large or very small real numbers. The final section introduced error detection and correction codes, including ASCII, EBCDIC, Gray Code, Excess-3 Code, BCD, and the use of Parity Bits and Hamming Code for maintaining data integrity.

By understanding these concepts, you have developed a strong foundation for advanced study in digital logic design, data communication, and microprocessor systems.

1.6. GLOSSARY

- **Analog Signal-** A continuous signal that varies over time.
- **Digital Signal-** A discrete signal with two states: 0 and 1.
- **Binary-** A number system with base 2 (digits 0 and 1).
- **Octal-** A number system with base 8 (digits 0 to 7).
- **Hexadecimal-** A number system with base 16 (digits 0-9 and A-F).
- **1's Complement-** A method of representing negative numbers by inverting all bits.
- **2's Complement-** A method of representing negative numbers by inverting all bits and adding 1.
- **Fixed-Point-** Binary representation where the position of the binary point is fixed.

NOTES

- **Floating-Point-** Binary representation where the binary point can "float" (IEEE 754 format).
- **Parity Bit** - A bit added to data to make the total number of 1's even or odd.

1.7. ANSWERS TO CHECK YOUR PROGRESS – POSSIBLE ONE

1. Check Your Progress 1.1

2. An analog system is a system that processes information using continuous signals that vary smoothly over time. These signals can take any value within a given range.
3. Two differences between analog **and digital signals**.
 - Analog signals are continuous, whereas digital signals are discrete (binary values such as 0 and 1).
 - Analog signals are more affected by noise, while digital signals are less affected by noise and distortion.
4. Digital systems use only two distinct levels (0 and 1), so small disturbances or noise do not usually change the signal value. Therefore, digital signals can be transmitted and processed more accurately.
5. Two examples each of analog and digital systems.
 Analog systems:
 - Analog clock
 - FM radio**Digital systems:**
 - Computer
 - Digital calculator

Check Your Progress 1.2

1. The four main components of a digital system.
 - Input unit
 - Processing unit
 - Storage unit
 - Output unit
2. The processing unit performs calculations, processes data, and controls the operations of the digital system according to the given instructions.
3. Difference between primary and secondary storage.
 - Primary storage is temporary memory used directly by the CPU, while secondary storage is permanent memory used for long-term data storage.
 - Primary storage is faster and smaller in capacity, whereas secondary storage is slower but has larger storage capacity.
4. Two examples of input and output devices.
 Input devices:
 - Keyboard

- Mouse
- Output devices:
- Monitor
 - Printer

NOTES

1.8. SUGGESTED READINGS – OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectionline.com/User/UnitUrls?unitId=3825	

1.9. REFERENCE

1. Mano, M. M., & Ciletti, M. D. (2013). *Digital Design: With an Introduction to the Verilog HDL*. Pearson.
2. Tocci, R. J., Widmer, N. S., & Moss, G. L. (2017). *Digital Systems: Principles and Applications*. Pearson.

1.10. EXERCISE

PART 1

A. Multiple choice question

1. Which of the following is an example of a set?
 - (a) A group of beautiful flowers
 - (b) The collection of vowels in the English alphabet
 - (c) The tallest buildings in the world
 - (d) The richest people on Earth
2. What is the representation of the set of even numbers less than 10 in roster form?
 - (a) {2,4,6,8} $x < 10$
 - (b) { $x|x$,is an even number $x < 10$ }
 - (c) { $2n|n \in \mathbb{N}, n < 5$ }
 - (d) All of the above
3. If $A = \{1,2,3,4\}$, which of the following is a proper subset of A?
 - (a) {1,2,3,4}
 - (b) {1,2}
 - (c) {1,2,3,4,5}
 - (d) $\emptyset$

NOTES

4. Which of the following is a countable set?
 - (a) The set of natural numbers
 - (b) The set of real numbers
 - (c) The set of points on a line
 - (d) The set of irrational numbers
5. What is the cardinality of the power set of $S=\{a,b,c\}$?
 - (a) 3
 - (b) 6
 - (c) 8
 - (d) 9

Answers with Explanations:

1. **Answer : (b)** The collection of vowels in the English alphabet

Explanation : A set must be well-defined. The vowels in the English alphabet (a,e,i,o,u) form a well-defined collection, whereas "beautiful" or "tallest" is subjective and not well-defined.

2. **Answer : (d)** All of the above

Explanation : The set of even numbers less than 10 can be represented in multiple ways: as a list (roster form), using a property (set-builder notation), or with a formula involving n . All options are correct representations.

3. **Answer : (b)** $\{1,2\}$

Explanation : A proper subset contains some, but not all, elements of the original set. $\{1,2,3,4\}$ is an improper subset, $\{1,2,3,4,5\}$ is not a subset, and $\emptyset$ set is an improper subset but not proper in this context.

4. **Answer : (a)** The set of natural numbers

Explanation : A countable set can be listed in a one-to-one correspondence with natural numbers. Natural numbers themselves are countable, whereas real numbers, points on a line, and irrational numbers are uncountable.

5. **Answer : (c)** 8

Explanation : The cardinality of the power set of a set with n elements is 2^n . Since S has 3 elements, the power set has $2^3=8$ subsets.

B. Short answer type question

1. Define the term "Discrete" in the context of digital electronics.
2. Give two real-world examples of analog systems that have been replaced by digital systems.
3. What is the function of a DAC (Digital-to-Analog Converter)?
4. Briefly explain why digital signals are less affected by electrical noise than analog signals.

PART 2**C. Long answer type question (200 words)**

1. What is a set? Provide a brief definition with one example.
2. Write the set of vowels in the English alphabet using both roster form and set-builder notation.
3. What is a subset? Provide one example of a subset of the set $A = \{1,2,3,4\}$
4. Differentiate between countable and uncountable sets with one example of each.
5. Define types of sets with suitable examples.

PART 3**D. Long answer type question (300 words)**

1. Explain properties of sets also give importance of sets in mathematics and real-world applications.
2. Discuss various methods of representing a set, including roster form, set-builder notation, and Venn diagrams.
3. Explain the difference between proper and improper subsets. Discuss the concept of a power set and demonstrate how to calculate the power set of a set with three elements.
4. Explain structure of countable and uncountable sets. Provide examples of countable sets (e.g., natural numbers, integers) and uncountable sets (e.g., real numbers).
5. Explain how the concept of countability relates to the idea of mapping elements and how it influences advanced mathematical theories.

NOTES

UNIT – 2

DATA TYPES AND NUMBER SYSTEMS

NOTES

UNIT STRUCTURE

- 2.1. Introduction**
- 2.2. Objective**
- 2.3. Data Types in Digital Systems**
 - 2.3.1. Numeric Data**
 - 2.3.2. Alphanumeric Data**
 - 2.3.3. Boolean Data**
 - 2.3.4. Multimedia Data**
- 2.4. Positional Number Systems**
 - 2.4.1. Concept of Base (Radix)**
 - 2.4.2. Place Value Principle**
 - 2.4.3. Weight of Each Position**
- 2.5. Decimal vs Binary Representation**
 - 2.5.1. Structure of Decimal System**
 - 2.5.2. Structure of Binary System**
 - 2.5.3. Bit, Nibble, Byte Concepts**
 - 2.5.4. Importance in Digital Electronics**
- 2.6. Glossary**
- 2.7. Answers to check your progress – possible one**
- 2.8. Suggested readings - OER**
- 2.9. Reference**
- 2.10. Exercise**

2.1. INTRODUCTION

We explored the physical architecture of digital systems and learned that they operate exclusively on discrete signals—specifically, high and low voltages. But a profound question arises: how can two simple voltage levels (ON and OFF) represent the vast complexity of human information? How does a computer know if a specific electrical pulse is a mathematical number, a letter in a text message, or a pixel in a high-definition photograph?

NOTES

The answer lies in how we categorize data and the mathematical language we use to represent it. In this unit, we will begin by looking at the various Data Types in digital systems. You will see that everything from your bank balance (Numeric Data) to your name (Alphanumeric Data) and even your favorite songs (Multimedia Data) must ultimately be translated into a format the machine can digest.

Once we understand what data is, we will explore how it is mathematically structured. We will dive into Positional Number Systems, breaking down the concepts of "Base" (or Radix) and "Place Value." Even though you have been using the Decimal system your entire life, analyzing its mathematical structure will make it incredibly easy to understand the language of computers: the Binary System. We will contrast the human-friendly Base-10 system with the machine-friendly Base-2 system, and introduce you to the fundamental building blocks of digital storage: bits, nibbles, and bytes. By the end of this unit, you will have mastered the basic vocabulary required to communicate mathematically with any digital processor. Let us dive into the language of machines!

2.2. OBJECTIVES:

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

- Identify and categorize the different data types processed by digital systems.
- Explain the foundational concepts of positional number systems, including Base (Radix) and place value.
- Compare the structural differences between the Decimal and Binary number systems.
- Define digital storage units such as bits, nibbles, and bytes, and explain their importance in electronics.

2.3. DATA TYPES IN DIGITAL SYSTEMS

Before a computer can process information, that information must be classified. Digital systems are highly structured, and they treat different types of data according to specific formatting rules.

2.3.1. Numeric Data

This is the most fundamental data type. Numeric data consists of numbers used for arithmetic calculations, such as addition or multiplication. This includes whole numbers (integers) and fractional numbers (floating-point numbers). In digital hardware, numeric data is strictly represented in binary format so the Arithmetic Logic Unit (ALU) can process it directly.

2.3.2. Alphanumeric Data

Humans communicate using alphabets and symbols, not just numbers. Alphanumeric data includes letters (A-Z), numbers (0-9) used as text (like a zip code), and special characters (@, #, \$). Because

computers only understand 1s and 0s, standard coding schemes like ASCII (American Standard Code for Information Interchange) or Unicode are used to assign a unique binary sequence to every single character on your keyboard.

2.3.3. Boolean Data

Named after mathematician George Boole, Boolean data is the simplest data type. It represents logical states and can only have one of two values: True (1) or False (0). This data type is the absolute backbone of digital decision-making, driving the logic gates we will study in the next unit.

2.3.4. Multimedia Data

Today's computers process massive amounts of multimedia: audio, images, and video. While complex, these are still just collections of binary data. An image is broken down into tiny dots called "pixels," and each pixel is assigned a binary number representing its specific color. Audio is sampled and converted into numerical voltage values. Ultimately, everything becomes binary.

Data Type	Primary Use	Example	Representation Method
Numeric	Mathematical calculations	45, -3.14	Direct Binary / 2's Complement
Alphanumeric	Text and communication	"Hello!", "A12"	ASCII / Unicode
Boolean	Logical decision making	True, False	1 (High) / 0 (Low)
Multimedia	Visual and Audio media	.JPG, .MP3	Pixel grids / Sample rates

Table 2.1 : Overview of Digital Data Types.

☒ Check Your Progress 2.1.

Notes : (a) Space is given below for your answer.
 (b) Check your answer with the one given at the end of this unit.

- Which coding scheme is commonly used to represent alphanumeric data in computers?

- What are the two possible values of Boolean data?

2.4. POSITIONAL NUMBER SYSTEMS

To understand binary, we must first understand the mechanics of counting. Both the human decimal system and the computer binary system are "Positional Number Systems."

2.4.1. Concept of Base (Radix)

The "Base" or "Radix" of a number system dictates the total number of unique symbols (digits) used in that system before you have to combine them to make larger numbers. For example, humans use Base-10 (Decimal) because we have 10 fingers. We use ten symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. Once we pass 9, we don't have a new single symbol; we combine '1' and '0' to make '10'.

2.4.2. Place Value Principle

In a positional system, the value of a digit depends not just on the symbol itself, but on its position within the number. For instance, the symbol '5' means five units in the number 15, but it means fifty units in the number 51. The position determines the multiplier.

2.4.3. Weight of Each Position

The multiplier for each position is called its "Weight," and it is calculated as **Base**^[Position]. The position numbering starts at 0 from the rightmost digit (the Least Significant Digit) and increases as you move left.

Let's analyze the decimal number 345:

- The '5' is in position 0. Its weight is $10^0 = 1$. ($5 * 1 = 5$)
- The '4' is in position 1. Its weight is $10^1 = 10$. ($4 * 10 = 40$)
- The '3' is in position 2. Its weight is $10^2 = 100$. ($3 * 100 = 300$)

When you add the weighted values together ($300 + 40 + 5$), you get the total value of 345. This exact same mathematical principle applies to the binary system used by computers.



Activity 2.1. DECONSTRUCTING PLACE VALUES

Title : The Mathematical Breakdown of Your Birth Year

Objective : To practically apply the concepts of Base, Position, and Weight to a familiar Base-10 number.

Material Needed : A notebook and a pen.

Activity Description:

1. **Write the Target Number :** Write down the four-digit year you were born (for example, 2001).
2. **Assign Positions :** Above each digit, write its position index. Remember, start from the far right with Position 0, moving left to 1, 2, and 3.

3. **Calculate the Weights :** Below each digit, write out the full mathematical expansion using the Base-10 weight formula (Digit $\times 10^{\{\text{Position}\}}$).

Example for 2001: $(2 \times 10^3) + (0 \times 10^2) + (0 \times 10^1) + (1 \times 10^0)$

4. **Solve :** Multiply them out and add them up to prove it equals your birth year.

Reflection:

Review your breakdown. Notice how the position completely changes the value of identical digits (like the two '0's in 2001 having different mathematical weights). By proving this rule on paper with the Decimal system, you are now perfectly prepared to apply this exact same logic to Base-2 (Binary) in the next section!

2.5. DECIMAL VS BINARY REPRESENTATION

Now that we understand positional weights, we can bridge the gap between human language and computer language.

2.5.1 Structure of Decimal System

As discussed, the Decimal system has a Radix of 10. It uses symbols 0-9, and the positional weights increase by powers of 10 (1, 10, 100, 1000). While perfect for human brains, it is terrible for electronics. Building a processor that can reliably distinguish between 10 different voltage levels (e.g., 1 Volt = 1, 2 Volts = 2) is expensive and prone to errors from electrical noise.

2.5.2 Structure of Binary System

The Binary system solves this problem. It has a Radix of 2. It uses only two symbols: 0 and 1. The positional weights increase by powers of 2 (1, 2, 4, 8, 16).

Let's look at the binary number **101**:

- The rightmost '1' has a weight of $2^0 = 1$. ($1 \times 1 = 1$)
- The middle '0' has a weight of $2^1 = 2$. ($0 \times 2 = 0$)
- The leftmost '1' has a weight of $2^2 = 4$. ($1 \times 4 = 4$)

Adding them up ($4 + 0 + 1$), we find that the binary number 101 equals 5 in decimal!

2.5.3. Bit, Nibble, Byte Concepts

In binary terminology :

- **Bit:** A single binary digit (0 or 1). It is the smallest unit of data.
- **Nibble:** A grouping of 4 bits (e.g., 1011).

NOTES

- **Byte:** A grouping of 8 bits. It is the standard unit of memory in modern computing. One byte can represent 256 different values (2^8).

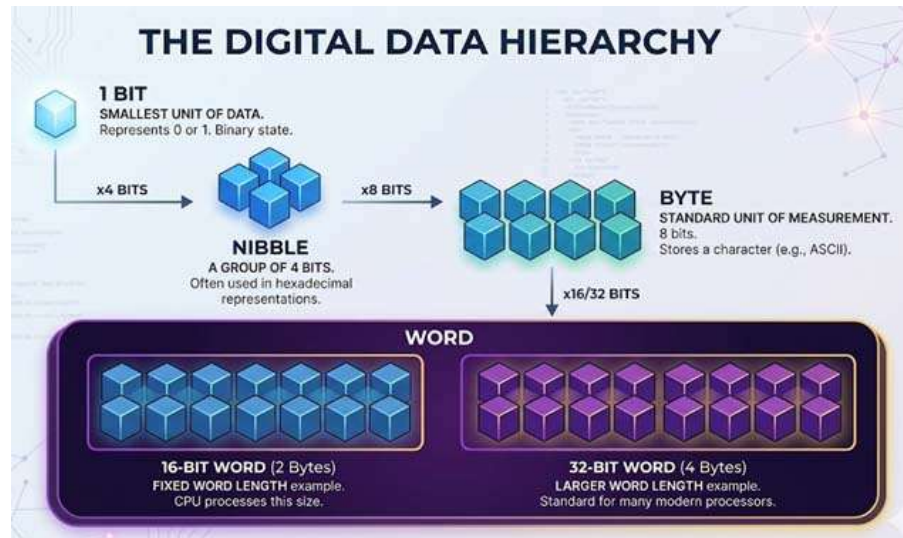


Fig. 2.1 : The Digital Data Hierarchy.

2.5.4. Importance in Digital Electronics

Binary is used exclusively in digital electronics because it perfectly maps to the physical hardware. A '1' represents a transistor being ON (high voltage, usually 5V), and a '0' represents it being OFF (low voltage, 0V). This high-contrast, two-state system is incredibly stable, extremely fast, and highly immune to electrical interference.

Feature	Decimal System	Binary System
Base (Radix)	10	2
Symbols Used	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0, 1
Positional Multipliers	1, 10, 100, 1000...	1, 2, 4, 8, 16...
Hardware Mapping	Impractical for electronics	Perfect for ON/OFF switches

Table 2.2 : Decimal vs. Binary Comparison.

☒ Check Your Progress 2.2.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. What is the decimal equivalent of the binary number 10?

.....

.....

2. How many bits are there in a single Byte?

.....

.....

.....

2.6. SUMMARY

In this unit, we decoded the basic vocabulary that allows digital systems to process the world. We began by classifying the different Data Types, discovering that whether a computer is calculating complex math (Numeric), rendering a text document (Alphanumeric), making a logical choice (Boolean), or playing a movie (Multimedia), every single piece of information is ultimately broken down into binary code.

To understand how binary works, we stepped back to analyze Positional Number Systems. We learned that the value of any number is dictated by its Base (Radix) and the mathematical weight of its position. By deconstructing our familiar Base-10 Decimal system, we saw how position multipliers ($10^0, 10^1, 10^2$) give value to digits. We then applied this exact mathematical rule to the Base-2 Binary system, where positional multipliers ($2^0, 2^1, 2^2$) allow us to calculate value using only 0s and 1s. Finally, we defined the standard measurements of digital storage—bits, nibbles, and bytes—and highlighted why the two-state binary system is the undisputed standard for achieving fast, error-free electronic processing.

2.7. GLOSSARY

- **Alphanumeric** : Data consisting of both letters and numbers, typically represented by codes like ASCII.
- **Base (Radix)** : The number of unique digits or symbols used in a positional number system.
- **Binary System** : A base-2 number system using only the digits 0 and 1.
- **Bit** : A "Binary Digit," representing a single 1 or 0; the smallest unit of digital data.
- **Byte** : A standardized group of 8 bits, capable of representing 256 unique values.

2.8. ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 2.1

1. The **ASCII** (American Standard Code for Information Interchange) scheme, or Unicode, is commonly used.
2. The two possible values of Boolean data are **True (1)** and **False (0)**.

NOTES

Check Your Progress 2.2

1. The decimal equivalent of binary 10 is **2**.
(Calculated as: $1 * 2^1 + 0 * 2^0 = 2$).
2. There are **8 bits** in a single Byte.

2.9. SUGGESTED READINGS – OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectonline.com/User/UnitUrls?unitId=3826	

2.10. REFERENCE

1. Mano, M. M., & Ciletti, M. D. (2013). *Digital Design*. Pearson.
2. Floyd, T. L. (2015). *Digital Fundamentals*. Pearson Education.

2.11. EXERCISE

PART – 1

A. Multiple Choice Questions

1. Which data type is strictly used to represent logical states of True or False?
 - (a) Numeric Data
 - (b) Alphanumeric Data
 - (c) Boolean Data
 - (d) Multimedia Data
2. What is the Radix (Base) of the binary number system?
 - (a) 2
 - (b) 8
 - (c) 10
 - (d) 16
3. A grouping of exactly 4 bits is commonly referred to as a:
 - (a) Byte
 - (b) Nibble
 - (c) Word
 - (d) Pixel
4. In the decimal number 582, what is the positional weight of the digit '8'?
 - (a) 10^0

- (b) 10^1
- (c) 10^2
- (d) 10^3
- 5. Why do computers use the binary system instead of the decimal system?
 - (a) It requires more symbols, allowing for more complex data.
 - (b) It maps perfectly to the ON/OFF states of electronic transistors.
 - (c) Humans process binary numbers faster than decimal numbers.
 - (d) It was chosen arbitrarily by early computer scientists.

Answers with explanations:1. **Answer : (c)**

Explanation : Boolean data is a specialized data type used in logic algebra that restricts values to only two possible states: True (represented by logic 1) and False (represented by logic 0).

2. **Answer : (a)**

Explanation : The binary system uses exactly two unique symbols (0 and 1) to represent all values, meaning its Base or Radix is 2.

3. **Answer : (b)**

Explanation : In digital terminology, a single binary digit is a bit, a group of four bits is called a nibble, and a group of eight bits is called a byte.

4. **Answer : (b)**

Explanation : Starting from the right (position 0), the digit '2' is at 10_0 . Moving one place to the left, the digit '8' sits in position 1, making its weight 10_1 (or 10).

5. **Answer : (b)**

Explanation : Electronic circuits are incredibly stable when dealing with just two extremes (High Voltage = 1, Low Voltage = 0). Designing hardware to recognize ten different voltage levels for a decimal system would be expensive and highly susceptible to noise errors.

B. Short answer type questions

1. Distinguish between Numeric data and Alphanumeric data.
2. Define the term "Radix" in the context of number systems.
3. Calculate the decimal value of the binary number 110.
4. Briefly explain how an image (Multimedia Data) is fundamentally stored in a digital system.

PART – 2**C. Long answer type question (200 words)**

1. Describe the concept of "Place Value" and "Positional Weight" in number systems. Use the decimal number 409 to demonstrate how position determines the actual mathematical value of a digit.
2. Compare the structural mechanics of the Decimal System with the Binary System. Focus on the number of symbols used, the base multiplier, and how numbers "roll over" when counting upwards.
3. Explain the relationship between bits, nibbles, and bytes. Why did the tech industry standardize the "Byte" as the primary unit for measuring memory and data storage?

PART – 3**D. Long answer type question (300 words)**

1. Provide a comprehensive breakdown of the four main Data Types (Numeric, Alphanumeric, Boolean, Multimedia) handled by digital systems. Explain how diverse, human-readable information—such as a complex mathematical formula, an email containing special characters, and a digital photograph—can all be translated into the same underlying format (binary 0s and 1s) for processing by a computer.
2. Analyze the significance of the Binary number system in the field of electronics. Discuss the physical hardware limitations that make the implementation of a Decimal-based electronic processor impractical. Explore how the binary states of '0' and '1' perfectly map to voltage thresholds and the ON/OFF switching states of transistors, thereby ensuring high processing speeds and excellent noise immunity.
3. Walk through the mathematical process of decoding a positional number system. Start by explaining the universal formula: $\text{Value} = \sum (\text{Digit} * \text{Base}^{\text{Position}})$. First, apply this formula to prove that the decimal number '27' equals twenty-seven. Then, use the exact same formula, changing only the Base multiplier, to prove that the binary number '11011' also represents the decimal value of twenty-seven. Show all your mathematical steps clearly.

UNIT – 3

LOGIC GATES

NOTES

UNIT STRUCTURE

- 3.1. Introduction**
- 3.2. Objective**
- 3.3. Logic Gates and Their Truth Tables**
 - 1.3.1. AND, OR, NOT Gates**
 - 1.3.2. NOR, NAND, XOR Gates**
- 3.4. Universal Gates and Applications**
- 3.5. Summary**
- 3.6. Glossary**
- 3.7. Answers to check your progress**
- 3.8. Suggested readings - OER**
- 3.9. Reference**
- 3.10. Exercise**

3.1. INTRODUCTION

In our previous unit, we decoded the language of computers. We learned that all data, whether it is a math problem or a multimedia file, is ultimately broken down into binary 0s and 1s. We also discovered that these 1s and 0s correspond directly to the physical High (ON) and Low (OFF) voltages in an electronic circuit. But storing a 1 or a 0 is not enough to make a machine "smart." To actually process data, the computer needs to make decisions based on those 1s and 0s.

Have you ever wondered how a simple calculator knows that $1 + 1 = 2$, or how a digital thermostat decides to turn on the heater only when the room is cold AND the system is enabled? This decision-making capability is the magic of "Logic Gates." Logic gates are the fundamental building blocks of all digital systems. They are physical electronic circuits (built from microscopic transistors) that take one or more binary inputs and produce a single binary output based on a strict logical rule.

In this unit, we will explore the "alphabet" of digital hardware. We will start with the three primary gates: AND, OR, and NOT. You will learn how to map out their decision-making rules using a mathematical tool called a "Truth Table." From there, we will look at derived gates like NAND, NOR, and the highly unique Exclusive-OR (XOR) gate. Finally, we will uncover a fascinating engineering shortcut: the Universal Gates. You will see how engineers can build the most complex supercomputer

in the world using only one type of gate, drastically reducing manufacturing costs. Let us dive into the logical core of the digital world!

3.2. OBJECTIVES

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

- Identify the symbols and functions of basic logic gates (AND, OR, NOT).
- Construct truth tables for basic, derived, and exclusive logic gates.
- Understand the operations of NAND, NOR, and XOR gates.
- Explain the concept of "Universal Gates" and demonstrate why they are highly valued in hardware manufacturing.

3.3. LOGIC GATES AND THEIR TRUTH TABLES

A logic gate evaluates the voltage levels of its inputs and decides the voltage level of its output. We map these possibilities using a "Truth Table"—a grid that lists all possible combinations of inputs and the resulting output.

A comprehensive guide to standard gates in digital logic.

LOGIC GATE	SCHEMATIC SYMBOL	BOOLEAN EXPRESSION	TRUTH TABLE (Inputs/Output)										
AND		$Y = A \cdot B$ (or AB)	<table><tr><th>Input A</th><th>Output Y</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td></tr></table>	Input A	Output Y	0	0	1	0	0	1	1	1
Input A	Output Y												
0	0												
1	0												
0	1												
1	1												
OR Gate		$Y = A + B$	<table><tr><th>Input A</th><th>Output Y</th></tr><tr><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	Input A	Output Y	0	0	0	1	1	0	1	1
Input A	Output Y												
0	0												
0	1												
1	0												
1	1												
NOT Gate (Inverter)		$Y = \bar{A}$ (or A')	<table><tr><th>A</th><th>Y</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	Y	0	1	1	0				
A	Y												
0	1												
1	0												
NAND Gate		$Y = (A \cdot B)'$	<table><tr><th>Input A B</th><th>Output Y</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td></tr></table>	Input A B	Output Y	0	1	1	1	0	0	1	0
Input A B	Output Y												
0	1												
1	1												
0	0												
1	0												
NOR Gate		$Y = (A + B)'$	<table><tr><th>Input A B</th><th>Output Y</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	Input A B	Output Y	0	1	1	0	0	0	1	1
Input A B	Output Y												
0	1												
1	0												
0	0												
1	1												
XOR Gate (Exclusive-OR)		$Y = A \oplus B$ (or $AB' + A'B$)	<table><tr><th>Input A B</th><th>Output Y</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	Input A B	Output Y	0	0	1	1	0	1	1	0
Input A B	Output Y												
0	0												
1	1												
0	1												
1	0												

Fig. 3.1 : Basic Logic Gates Reference Chart.

3.3.1. AND, OR, NOT Gates

These three are the absolute foundation of Boolean logic.

- **The AND Gate :** This gate requires all of its inputs to be High (1) for the output to be High (1). Think of it like a safety launch system for a rocket: Key A *AND* Key B must be turned simultaneously. If either is missing, the rocket does not launch (0). Mathematically, it performs logical multiplication ($Y = A \cdot B$).
- **The OR Gate :** This gate requires just one of its inputs to be High (1) for the output to be High. Think of a car's interior light: it turns on if the driver's door is open *OR* the passenger's door is open. Mathematically, it performs logical addition ($Y = A + B$).
- **The NOT Gate (Inverter) :** Unlike the others, the NOT gate has only one input. Its job is to completely reverse the signal. If you feed it a 1, it outputs a 0. If you feed it a 0, it outputs a 1. It is indicated by a bar over the variable ($Y = \bar{A}$).

3.3.2. NOR, NAND, XOR Gates

By combining the basic gates, we create derived gates with specific behaviors.

- **The NAND Gate:** This is an AND gate followed by a NOT gate. It outputs a Low (0) *only* when all its inputs are High (1). In every other scenario, it outputs a 1.
- **The NOR Gate:** This is an OR gate followed by a NOT gate. It outputs a High (1) *only* when all its inputs are Low (0). If even a single 1 enters the gate, the output is forced to 0.
- **The XOR Gate (Exclusive-OR):** The XOR gate is a strict "difference detector." It outputs a High (1) only when its inputs are strictly different from each other (one is 1, the other is 0). If the inputs are the same (both 0s or both 1s), the output is 0. This gate is the mathematical heart of digital addition, as it perfectly mimics the rule that $1 + 1 = 0$ (with a carry over).

Gate Name	Logical Equation	High Output (1) Condition	Low Output (0) Condition
AND	$Y = A \cdot B$	ALL inputs must be 1	ANY input is 0
OR	$Y = A + B$	ANY input is 1	ALL inputs must be 0
NOT	$Y = \overline{A}$	Input must be 0	Input must be 1
NAND	$Y = \overline{A \cdot B}$	ANY input is 0	ALL inputs must be 1
XOR	$Y = A \oplus B$	Inputs must be DIFFERENT	Inputs must be the SAME

Table 3.1 : Summary of Logic Gate Behaviors.

☒ Check Your Progress 3.1.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. Which logic gate is often referred to as an "Inverter"?

.....

2. What will be the output of an XOR gate if both of its inputs are 1?

.....

NOTES

3.4. UNIVERSAL GATES AND APPLICATIONS

In digital electronics, manufacturing complexity directly equates to higher costs. If a company building a microprocessor has to manufacture a silicon chip that contains millions of AND gates, OR gates, and NOT gates, they have to design three entirely different microscopic transistor structures on the same chip. This is difficult and expensive.

To solve this, hardware engineers rely on the concept of **Universal Gates**. A universal gate is a logic gate that can be used—in various combinations with itself—to replicate the behavior of any other logic gate. There are exactly two universal gates: the **NAND gate** and the **NOR gate**.

This means that if a factory only knows how to build NAND gates, they can still build an entire computer! By wiring a NAND gate's two inputs together, it turns into a NOT gate. By combining three NAND gates in a specific pattern, they act exactly like an OR gate. The universality of NAND and NOR gates makes them incredibly valuable in the mass production of Integrated Circuits (ICs) and memory chips. Most Flash memory (like the USB drive on your desk) is built entirely out of cascaded NAND gates because designing one repeating structure saves massive amounts of silicon space and engineering time.

 Check Your Progress 3.2.

Notes : (a) Space is given below for your answer.
(b) Check your answer with the one given at the end of this unit.

- 1. Name the two universal gates.
.....
.....
.....
- 2. Why are universal gates so valuable in hardware manufacturing?
.....
.....
.....

 Activity 3.1. THE LOGIC OF YOUR HOUSEHOLD

Title : Identifying Logic Gates in Everyday Physical Systems
Objective : To map the theoretical behavior of AND, OR, and XOR gates to real-world electrical systems in your immediate environment.

Material Needed : A notebook, a pen, and access to the light switches in your home.

Activity Description :

1. **The OR Logic :** Find a room in your house with multiple lamps or a ceiling fan with multiple pull chains. If flipping Switch A turns on light in the room, OR flipping Switch B turns on light, you have identified physical OR logic. Write down this example.
2. **The AND Logic :** Think of an appliance that requires two actions to work. For example, a microwave will only start heating if the timer is set (Input A) AND the door is securely closed (Input B). Write down one appliance in your home that uses physical AND logic.
3. **The XOR Logic :** Find a hallway or a staircase with two light switches that control the exact same bulb (a two-way switch). Notice that if both switches are "Down" (0,0), the light is off. If you flip one "Up" (1,0), the light turns on. If both are "Up" (1,1), the light turns off! This perfectly mirrors the Exclusive-OR gate. Document this behavior.

Reflection :

Review your notes. You should realize that logic gates are not just abstract computer concepts; they are simply formalized rules for cause and effect. By recognizing how multiple physical conditions (door closed, button pressed) result in a single outcome (machine starts), you are already thinking like a digital logic designer!

3.5. SUMMARY

In this unit, we successfully bridged the gap between raw binary data and actual computational decision-making. We introduced Logic Gates, the fundamental hardware building blocks that evaluate binary inputs to produce logical outcomes. We started with the three foundational gates: the AND gate (requiring all inputs to be High), the OR gate (requiring any input to be High), and the NOT gate (which strictly inverts the signal). We learned how to document these strict rules using Truth Tables, which provide a complete visual map of every possible input and output scenario.

Moving forward, we explored the derived gates. We saw how adding an inversion bubble creates the NAND and NOR gates, completely flipping the behavior of their base gates. We also analyzed the XOR gate, noting its unique ability to act as an inequality detector, outputting a High signal only when its inputs disagree. Finally, we uncovered the industrial secret of Universal Gates. We learned that the NAND and NOR gates possess the remarkable ability to mimic any other logic function. This universality is a cornerstone of modern electronics, allowing engineers to design cheaper, denser, and highly efficient

NOTES

microprocessors using a single repeating gate structure. With this logical alphabet mastered, you are now ready to combine these gates into complex Boolean algebraic equations in our next unit.

3.6. GLOSSARY

- **Logic Gate** : An electronic circuit that performs a basic Boolean logical operation on one or more inputs to produce a single output.
- **Truth Table**: A mathematical table used to represent the logical function of a gate, showing all possible input combinations and their resulting outputs.
- **Inverter**: Another name for the NOT gate; it reverses the logical state of its input.
- **Universal Gate**: A gate (specifically NAND or NOR) that can be used alone to implement any Boolean logic function.
- **Exclusive-OR (XOR)**: A gate that outputs true (1) only if its inputs are strictly different from one another.

3.7. ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 3.1

1. The **NOT gate** is often referred to as an Inverter.
2. If both inputs of an XOR gate are 1, the output will be **0** (because the inputs are not different).

Check Your Progress 3.2

1. The two universal gates are **NAND** and **NOR**.
2. They are valuable because a manufacturer only needs to design and mass-produce one single type of gate to build an entire processing circuit, which drastically reduces manufacturing complexity and costs.

3.8. SUGGESTED READINGS – OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectonline.com/User/UnitUrls?unitId=3827	

3.9. REFERENCE

1. Floyd, T. L. (2015). *Digital Fundamentals*. Pearson Education.
2. Tocci, R. J., Widmer, N. S., & Moss, G. L. (2017). *Digital Systems: Principles and Applications*. Pearson.

PART – 1

NOTES

Explanation : The term NAND stands for "NOT-AN(d)" It is constructed by taking the output of a standard AND gate and running it directly through a NOT gate (represented by the small bubble on the symbol).

B. Short answer type questions

1. Define what a "Truth Table" is in digital logic.
2. Write the Boolean mathematical expression for a 2-input OR gate.
3. Briefly explain the functional difference between an OR gate and an XOR gate.
4. Why is the XOR gate often called an "inequality detector"?

PART – 2

C. Long answer type question (200 words)

1. Describe the behavior of the NAND gate using a practical real-world analogy. Draw the truth table for a 2-input NAND gate to support your explanation.
2. Explain the concept of Universal Gates. Why is the ability to construct any logic circuit using solely NAND gates an advantage for a hardware manufacturing company?
3. Discuss the XOR gate. Beyond simple logic puzzles, explain why this specific gate is absolutely critical for Arithmetic Logic Units (ALUs) when performing binary addition.

PART – 3

D. Long answer type question (300 words)

1. Provide a comprehensive comparative analysis of the AND, OR, and NOT gates. Discuss their logical conditions for outputting a '1', provide the Boolean equation for each, and explain how these three fundamental gates serve as the building blocks for all higher-level decision-making in a digital computer. Use truth tables for 2-input variables to prove your points.
2. Detail the process of deriving the NAND and NOR gates from the foundational gates. Start by explaining the role of the "inversion bubble" used in digital schematics.
3. Evaluate the application of logic gates in automated safety systems. Imagine you are designing a safety circuit for an industrial metal press. The press should only operate if the operator presses the start button (Input A) AND the safety guard is down (Input B). However, the machine must instantly stop if the emergency override button is pressed (Input C). Design the logical flow using AND, OR, and NOT gates to ensure the machine operates safely. Detail your logic step-by-step.

UNIT – 4

BOOLEAN ALGEBRA AND SIMPLIFICATIONS

NOTES

UNIT STRUCTURE

- 4.1. Introduction
- 4.2. Objective
- 4.3. Introduction to Boolean Algebra
 - 4.3.1. Basic Boolean Laws
 - 4.3.2. DeMorgan's Theorems
- 4.4. Simplification Techniques
 - 4.4.1. Karnaugh Map (K-Map)
 - 4.4.2. Sum of Product (SOP) and Product of Sum (POS) Forms
- 4.5. Summary
- 4.6. Glossary
- 4.7. Answers to check your progress
- 4.8. Suggested readings - OER
- 4.9. Reference
- 4.10. Exercise

4.1. INTRODUCTION

In the previous unit, we explored the physical building blocks of digital electronics: the Logic Gates. We learned how AND, OR, and NOT gates make basic decisions based on high and low voltages. However, if you are designing a complex system, like a smartphone processor, you aren't just connecting two or three gates; you are connecting millions. If you blindly connect gates together to solve a complex problem, you will likely use far more hardware than you actually need. Extra gates mean a bigger microchip, more power consumption, more heat, and a higher manufacturing cost.

How do we solve this? Before engineers build a circuit with physical silicon, they design it using mathematics. But this isn't the standard algebra you learned in high school; this is a special mathematical system called Boolean Algebra, introduced by the mathematician George Boole in 1854. In this unique algebra, variables can only ever hold two values: 0 and 1.

NOTES

In this unit, we will learn the rules of this digital mathematics. We will explore the Basic Boolean Laws, such as the Idempotent and Complement laws, which often allow us to instantly delete redundant gates from our designs. We will also examine the famous DeMorgan's Theorems, a powerful set of rules that lets us switch between AND and OR logic—a trick that is incredibly useful when manufacturing hardware. Finally, we will look at structural ways to simplify logic, including the Sum of Product (SOP) form and a highly visual, grid-based technique known as the Karnaugh Map (K-Map). By the end of this unit, you will be equipped to take a messy, expensive logic circuit and mathematically reduce it to its sleekest, most efficient form. Let's get started on saving hardware!

4.2. OBJECTIVES:

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

- Define the fundamental laws and rules of Boolean Algebra.
- Apply DeMorgan's Theorems to transform and simplify complex logical expressions.
- Construct and evaluate expressions in Sum of Product (SOP) and Product of Sum (POS) forms.
- Utilize Karnaugh Maps (K-Maps) as a visual tool to minimize digital logic circuits.
- Understand how mathematical simplification directly translates to reduced hardware costs.

4.3. INTRODUCTION TO BOOLEAN ALGEBRA

Boolean Algebra is the mathematics of digital systems. Just like standard algebra uses variables (like x and y) and operations (like addition and multiplication), Boolean algebra uses variables (like A and B) and logical operations (OR, AND, NOT). However, because Boolean variables can only be 0 or 1, the rules are delightfully different.

4.3.1 Basic Boolean Laws

The manipulation and simplification of digital logic circuits rely on established **Boolean Laws**. These laws allow designers to rearrange variables and eliminate redundant terms to optimize hardware.

1. **Commutative Law** : The order of variables in an AND or OR operation does not change the result ($A+B=B+A$ and $A \cdot B=B \cdot A$).
2. **Associative Law** : The grouping of variables does not affect the result ($A+(B+C)=(A+B)+C$).
3. **Distributive Law** : This law allows the factoring of expressions ($A(B+C)=AB+AC$ and $A+BC=(A+B)(A+C)$).
4. **Idempotent Law** : Operations with the same variable result in the variable itself ($A+A=A$ and $A \cdot A=A$).

5. **Absorption Law** : This simplifies terms where one variable is absorbed into another ($A+AB=A$ and $A(A+B)=A$).
6. **Identity and Null Laws** : $A+0=A$, $A \cdot 1=A$ (Identity), and $A+1=1$, $A \cdot 0=0$ (Null).
7. **Inverse Law** : $A+\bar{A}=1$ and $A \cdot \bar{A}=0$.

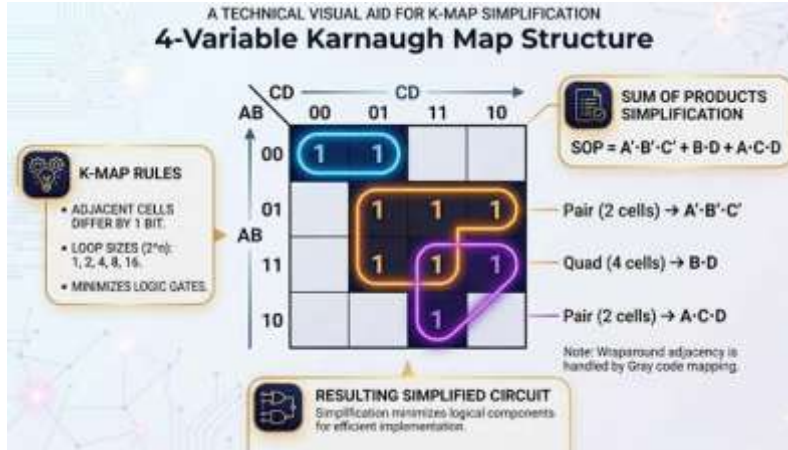


Fig. 4.1 : 4-Variable Karnaugh Map Structure.

4.4.2. DeMorgan's Theorems De Morgan's Theorems

The most powerful tools in Boolean algebra for finding the complement of a function.

- **First Theorem:** The complement of a sum is equal to the product of the complements ($\overline{A+B}=\bar{A} \cdot \bar{B}$).
- **Second Theorem:** The complement of a product is equal to the sum of the complements ($\overline{A \cdot B}=\bar{A} + \bar{B}$).

These laws are essential for converting between gate types and simplifying the implementation of complex logical functions in real-world circuits.

☒ Check Your Progress 4.1.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. According to the Null Law, what does $A + 1$ equal?

.....

2. Briefly state the phrase used to remember DeMorgan's Theorems.

.....

NOTES

**Activity 4.1 : THE DISTRIBUTIVE LAW IN YOUR DAILY LIFE**

Title : Simplifying Your Own Daily Logic

Objective : To recognize how the Boolean Distributive Law naturally applies to everyday human decision-making and reduces "mental" hardware.

Material Needed : A notebook and a pen.

Activity Description :

1. **The Complex Scenario:** Write down this weekend plan: *"I will take my umbrella if it is raining AND I am going to the grocery store, OR I will take my umbrella if it is raining AND I am going to the cinema."*
2. **Identify the Variables :** Let $A =$ "It is raining". Let $B =$ "Going to the grocery store". Let $C =$ "Going to the cinema". The logic equation for taking the umbrella (Y) is: $Y = (A.B) + (A.C)$.
2. **Apply the Distributive Law :** Look at the equation. The variable A is repeated. Mathematically "pull out" the common variable A , transforming the equation into: $Y = A.(B + C)$.
3. **Translate Back to English :** Write out the new, simplified sentence based on your minimized equation.

Reflection :

Your new sentence should read: *"I will take my umbrella if it is raining AND I am going to EITHER the grocery store OR the cinema."* Notice how much shorter and more efficient the second sentence is? You just saved your brain from evaluating the "raining" condition twice. In digital electronics, pulling out that common variable physically eliminates an entire AND gate from the circuit board!

4.5. SIMPLIFICATION TECHNIQUES

While applying Boolean laws manually is great for simple equations, human errors easily slip in when equations get long. Therefore, engineers use structured and visual techniques to guarantee the simplest circuit design.

4.5.1. Karnaugh Map (K-Map)

While algebraic simplification is useful, it can be non-intuitive for complex functions. The **Karnaugh Map (K-Map)** is a graphical tool that provides a systematic method for simplification. A K-map uses a grid of cells where each cell represents a unique combination of input variables (a **minterm** or **maxterm**). For n variables, a K-map contains 2^n cells.

Rules for K-Map Simplification:

- Groups must contain only 1s (for SOP) or only 0s (for POS).
- The number of cells in group must be a **power of 2** (1, 2, 4, 8, etc.).
- Groups must be as large as possible to achieve maximum simplification.
- Cells can be wrapped around the edges (top-to-bottom and left-to-right).

- **Don't Care Conditions (X):** These represent input combinations that cannot occur in a real circuit; they can be treated as 0 or 1 to help create larger groups.

4.5.2. Sum of Product (SOP) and Product of Sum (POS) Forms

Boolean functions are typically expressed in two standardized formats to ensure consistency and ease of design.

Sum of Products (SOP): In this form, product terms (called **minterms**) are ORed together. An SOP expression is used to define the conditions under which the circuit output is **Logic 1**.

For example, $Y = \bar{A}BC + AB\bar{C}$. **Product of Sums (POS):** Here, sum terms (called **maxterms**) are ANDed together. A POS expression defines the conditions where the output is **Logic 0**.

For example, $Y = (A+B+C) \cdot (\bar{A} + \bar{B} + C)$.

A **Canonical form** contains all variables in every term, whereas a **Standard form** may have simplified terms with fewer literals.

☒ Check Your Progress 4.2.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. Why are the rows and columns of a K-Map arranged using Gray Code?

.....

2. What do you group together in a K-Map to find a minimized SOP expression?

.....

4.6. SUMMARY

In this unit, we uncovered the mathematical tools that engineers use to optimize digital circuits. We began by introducing Boolean Algebra, a unique mathematical system where variables are strictly limited to 0 and 1. We explored the Basic Boolean Laws, such as Identity, Null, and Idempotent, which allow us to algebraically manipulate and shrink logic equations. We paid special attention to DeMorgan's Theorems, learning that "breaking the line and changing the sign" gives us the vital ability to convert AND-based logic into OR-based logic and vice versa.

Furthermore, we recognized that manual algebraic simplification is prone to human error, leading us to explore structural simplification

NOTES

techniques. We defined the two standard equation formats: Sum of Product (SOP) and Product of Sum (POS). Finally, we introduced the Karnaugh Map (K-Map), a brilliant visual grid that uses adjacent cell properties to effortlessly cancel out redundant variables. By grouping 1s into pairs, quads, or octets on a K-Map, we can extract the most minimal SOP expression possible. Mastering these simplification techniques is absolutely crucial, because a minimized mathematical expression directly translates to a circuit that requires fewer physical gates, consumes less power, costs less money, and operates with a lower propagation delay.

4.7. GLOSSARY

- **Boolean Algebra** : A branch of mathematics in which the values of the variables are the truth values true (1) and false (0).
- **DeMorgan's Theorems** : Two mathematical rules that describe how to negate an entire logical sum or product.
- **Karnaugh Map (K-Map)** : A graphical tool used to simplify Boolean algebra expressions without using complex algebraic equations.
- **Sum of Product (SOP)**: A Boolean expression format consisting of multiple AND terms combined with a final OR operation.
- **Minterm**: A product term in SOP form that contains every variable of the function, either in its normal or complemented form.

4.8. ANSWERS TO CHECK YOUR PROGRESS – POSSIBLE ONE

Check Your Progress 4.1

1. According to the Null Law, $A + 1 = 1$ always equals 1.
2. The phrase is: "Break the line, change the sign."

Check Your Progress 4.2

1. They use Gray Code so that only one variable changes state between any two adjacent cells, allowing redundant variables to cancel out when grouped.
2. You group the 1s (ones) together to find an SOP expression.

4.9. SUGGESTED READINGS – OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectonline.com/User/UnitUrls?unitId=3828	

4.10. REFERENCE

1. Digital Logic Applications and Design – John M. Yarbrough – 981-240-062-1 – 2001

2. Digital Principles and Design – Donald D. Givone – 978-0-07-052906-9 – 2002
3. HDL Programming (VHDL and Verilog) – Nazeih M. Botros – 978-8126514781 – 2008
4. Digital Circuits and Design – D. P. Kothari and J. S. Dhillon – 978-9332543539 – 2016
5. Fundamentals of Logic Design – Charles H. Roth, Jr. – Cengage Learning
6. Electronics Lab Manual – K. A. Navas – 978-8120351424 – 2015

4.11. EXERCISE

PART – 1

A. Multiple Choice Questions

1. Which of the following gates are classified as "Universal Gates" because they can realize any Boolean function?

(a) AND and OR	(b) NAND and NOR
(c) XOR and XNOR	(d) NOT and AND
2. According to De-Morgan's First Law, the complement of the sum of two or more variables is equal to:

(a) The sum of the variables	(b) The product of the complements
(c) The product of the complements of those variables	(d) The complement of the product of the variables
3. In a Karnaugh Map (K-map), the number of cells for a system with 'n' input variables is calculated as:

(a) n^2	(b) $2n$	(c) 2^n	(d) $n+2$
-----------	----------	-----------	-----------
4. Which Boolean law states that the order in which variables are ORed or ANDed does not affect the result (e.g., $A + B = B + A$)?

(a) Associative Law	(b) Distributive Law
(c) Commutative Law	(d) Absorption Law
5. A "minterm" is defined as a:

(a) Sum term in a POS expression	(b) Product term in an SOP expression where the function output is '1'
(c) Term with the minimum number of variables	(d) Complement of a maxterm

Answers with Explanations :

1. **Answer : (b)** NAND and NOR

Explanation : NAND and NOR gates are called universal gates because they can be used to produce all other basic Boolean

NOTES

functions, such as AND, OR, and NOT, allowing for any logic circuit to be built using only one type of these gates.

2. **Answer : (c)** The product of the complements of those variables

Explanation : De-Morgan's First Law states that $(A.B.C)^{\bar{}} = A^{\bar{}}B^{\bar{}}C^{\bar{}}$ meaning the complement of a sum is equal to the product of the complements.

3. **Answer : (c)** 2^n

Explanation : The number of cells in a K-map must equal the number of possible input combinations, which is determined by 2^n where n is the number of variables.

4. **Answer : (c)** Commutative Law

Explanation : The Commutative Law asserts that the sequence of variables does not change the result of the logical operation.

5. **Answer : (b)** Product term in an SOP expression where the function output is '1'

Explanation : Each product term in a Sum of Products (SOP) form is called a minterm, representing a specific combination where the output is high.

B. Short answer type question

1. State the Boolean Complement Law for both addition and multiplication.
2. Define what a "Minterm" is in digital logic.
3. Briefly explain why K-Maps use Gray code for their row and column labeling.
4. Write out the Product of Sum (POS) form for the variables A, B, and C.

PART 2

C. Long answer type question (200 words)

1. What is a set? Provide a brief definition with one example.
2. Write the set of vowels in the English alphabet using both roster form and set-builder notation.
3. What is a subset? Provide one example of a subset of the set $A=\{1,2,3,4\}$
4. Differentiate between countable and uncountable sets with one example of each.

PART 3

D. Long answer type question (300 words)

1. Explain properties of sets also give importance of sets in mathematics and real-world applications.
2. Define types of sets with suitable examples.



DR. C. V. RAMAN UNIVERSITY (C.G.)

SLM FEEDBACK FORM

PROGRAMME : _____
COURSE : _____
BLOCK : _____
UNIT : _____

Dear Student,

While studying the units of this block, you may have found certain portions of the text difficult to comprehend. We wish to know your difficulties and suggestions, in order to improve the course. Therefore, we request you to fill up and send us the following questionnaire, which pertains to this block. If you find the space provided insufficient, kindly use a separate sheet.

Please mail to aisectcontent@aisect.org / for online submission please scan QR code:

Questionnaire

1. Name : _____

2. Enrolment No. :

--	--	--	--	--	--	--	--	--	--	--

3. How many hours did you need for studying the units? :

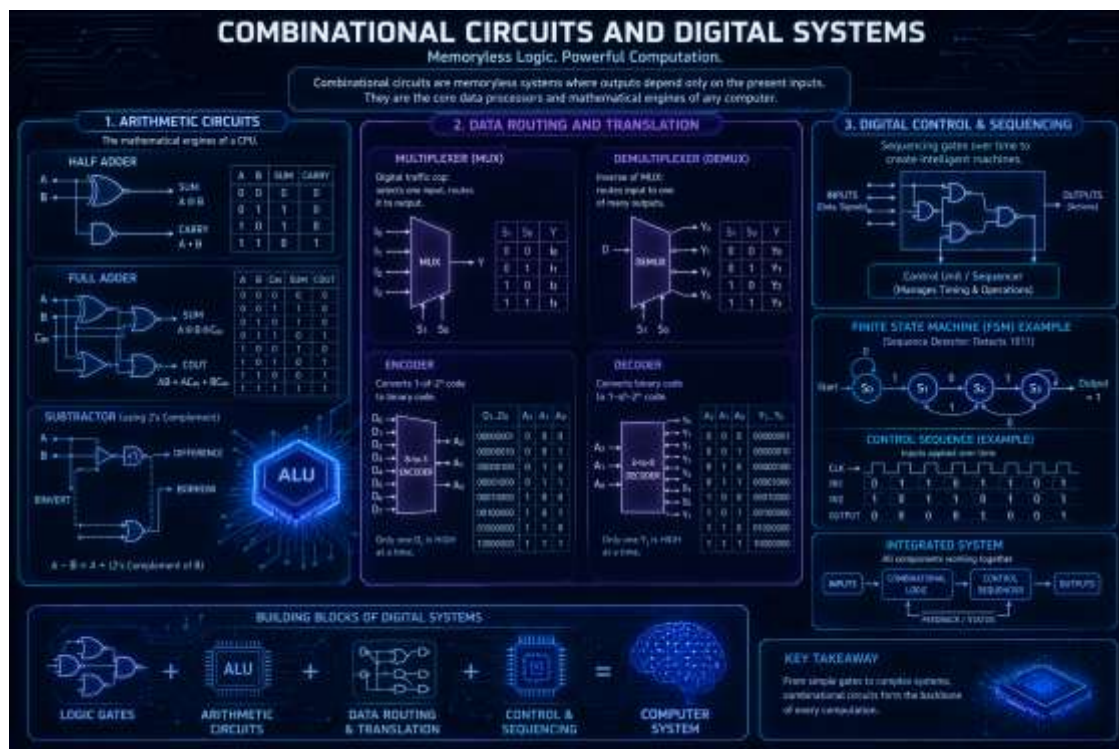
--

4. Please give your reactions to the following items based on your reading of the block:

Items	Excellent	Very Good	Good	Poor	Give specific Examples (If poor)
Presentation Quality					
Language and Style					
Illustrations					
Used (diagrams, tables, etc.)					
Conceptual Clarity					
Check Your Progress Questions					
Suggested Readings - OER					

5. Any other comments:

Block - II	COMBINATIONAL LOGIC DESIGN	43 - 84
Unit 5	Combinational Circuits : Adder and Subtractors	45
Unit 6	Combinational Circuits: Multiplexer and Demultiplexer	55
Unit 7	Encoders and Decoders	65
Unit 8	Digital Control Circuits and Sequencing	73



This section explores digital circuits where the output is determined strictly by the present state of its inputs—they possess absolutely no memory. As memoryless systems, combinational circuits are the core data processors and mathematical engines of any computer. We will begin by engineering mathematical marvels like Adders and Subtractors, constructing the full-bit logical arithmetic paths that CPUs require for calculation. Following this, we transition to data routing and translation. You will analyze the operation of **Multiplexers** (MUX), often called digital traffic cops, which take multiple data inputs and route only one specified signal to a single output line based on selector switches. Its inverse, the Demultiplexer (DeMUX), will also be covered. This block will also dissect Encoders and Decoders, the translators that bridge different computational codes, transforming human decimal input into machine binary and vice versa. Finally, we integrate these components to explore Digital Control and Sequencing. You will understand how individual gates are sequenced together over time to create an integrated machine capable of complex algorithms, acting as the centralized conductor that coordinates the thousands of disparate logic components on a motherboard into one cohesive computing engine.

UNIT – 5

COMBINATIONAL CIRCUITS: ADDERS AND SUBTRACTORS

NOTES

UNIT STRUCTURE

- 5.1 Introduction
- 5.2 Objective
- 5.3 Overview of Combinational Circuits
 - 5.3.1 Differences from Sequential Logic
- 5.4 Arithmetic Combinational Circuits
 - 5.4.1 Adders and Subtractors
 - 5.4.2 Binary Parallel Adders
- 5.5 Summary
- 5.6 Glossary
- 5.7 Answers to check your progress
- 5.8 Suggested readings - OER
- 5.9 Reference
- 5.10 Exercise

5.1. INTRODUCTION

The beginning of **Block 2 : Combinational Logic Design**. Over the past few units, we have been studying the raw alphabet and mathematics of digital electronics. We learned how basic logic gates (AND, OR, NOT) make simple binary decisions, and we learned how Boolean algebra helps us simplify those decisions to save hardware. But logic gates on their own don't actually *do* much.

Think of logic gates like raw bricks and mortar. Now, it is time to start building houses. In this unit, we will begin combining these basic gates to create complex functional circuits that perform actual, useful work for a computer. Specifically, we will look at "Combinational Circuits."

We will start by defining exactly what makes a circuit "combinational" and how it differs from the "sequential" circuits we will study later in the course. Then, we dive right into the mathematical heart of the Central Processing Unit (CPU): Arithmetic Circuits. If a computer is fundamentally just a giant calculator, it needs hardware to calculate. We will construct Half Adders and Full Adders, which are the exact transistor arrangements responsible for binary addition. We will also look at Subtractors, and finally, we will explore how engineers cascade

these tiny 1-bit adder circuits together to create Binary Parallel Adders, enabling the computer to process large numbers. Let's start building the calculator!

5.2. OBJECTIVES

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

- Define what a combinational circuit is and explain its fundamental operating principle.
- Distinguish between combinational logic and sequential logic.
- Describe the logic, truth tables, and Boolean equations for Half Adders and Full Adders.
- Explain the operation of Half Subtractors and Full Subtractors.
- Evaluate how individual Full Adders are connected to form Binary Parallel Adders for multi-bit arithmetic.

5.3. OVERVIEW OF COMBINATIONAL CIRCUITS

In digital design, every circuit falls into one of two major categories: Combinational or Sequential.

A **Combinational Circuit** is a highly predictable logic circuit where the output at any given moment depends *strictly and entirely* on the current inputs present at that exact moment.

Imagine a combination lock on a safe. If you input the correct digits (the current input), the safe opens (the output). It doesn't matter if you tried 100 wrong combinations yesterday; the lock doesn't remember the past. It only evaluates what you are inputting right now. In a combinational logic circuit, binary data flows in one direction: from the input, through the logic gates, directly to the output. There are no memory elements, and there is no feedback loop sending output data back to the input.

5.3.1. Differences from Sequential Logic

To fully grasp combinational circuits, it helps to briefly contrast them with what they are not: Sequential Circuits (which we will study in depth in Block 3).

- **Memory:** Combinational circuits have zero memory. Sequential circuits have built-in memory elements to store past data.
- **Feedback:** Combinational circuits have no feedback paths. Sequential circuits use feedback loops where the output loops back to influence the next input.
- **Clock Dependency:** Combinational circuits react instantly (ignoring minor gate delays) to any change in input. Sequential circuits are usually governed by a strict ticking "clock" signal that synchronizes when changes can happen.

COMBINATIONAL CIRCUITS: ADDERS AND SUBTRACTORS

Feature	Combinational Circuits	Sequential Circuits
Output Dependence	Current inputs only	Current inputs + Past states (Memory)
Feedback Loop	No	Yes
Memory Elements	Absent	Present (Flip-flops, Latches)
Examples	Adders, Multiplexers, Encoders	Counters, Shift Registers, RAM

NOTES

Table 5.1: Combinational vs. Sequential Circuits.

Check Your Progress 5.1.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

- Does the output of a combinational circuit depend on previous input states?

.....

- Mention one structural element present in sequential logic that is absent in combinational logic.

.....



Activity 5.1. COMBINATIONAL VS SEQUENTIAL IN YOUR HOME

Title : The "No Memory" Hardware Test

Objective : To distinguish between combinational and sequential behaviors in everyday household electrical items.

Material Needed : A notebook, a pen, and your immediate environment.

Activity Description:

- Find a Combinational Device :** Walk to a standard light switch that controls a single room light. Flip it up; the light turns on. Flip it down; the light turns off. It doesn't matter how many times you flipped it yesterday. The light's state strictly depends on the current position of the switch. This is physical Combinational logic. Write this down.

NOTES

2. **Find a Sequential Device :** Now, look at a smart device, like a television or a modern laptop, that has a single push-button for power. If the TV is OFF, pressing the button turns it ON. If the TV is ON, pressing that exact same button turns it OFF. The input action is identical, but the output changes because the TV remembers its previous state. This is physical Sequential logic. Write this down.

Reflection:

Review your findings. Notice how combinational devices act more like direct physical overrides, while sequential devices act more like toggles that require an internal memory of what just happened. In digital hardware, building the TV button requires complex memory circuits, while building the simple light switch only requires direct logic gates!

5.4. ARITHMETIC COMBINATIONAL CIRCUITS

The Arithmetic Logic Unit (ALU) inside a processor handles all mathematical operations. The foundation of this unit relies on combinational circuits built specifically to add and subtract binary bits.

5.4.1. Adders and Subtractors

Addition in binary follows simple rules: $0 + 0 = 0$, $0 + 1 = 1$, $1 + 0 = 1$, and $1 + 1 = 10$ (which is a Sum of 0 with a Carry of 1). We build hardware to perfectly replicate these rules.

- **The Half Adder :** A Half Adder is a combinational circuit designed to add exactly two single bits (A and B). It produces two outputs: a Sum (S) and a Carry (C).
 - Looking at the logic, the Sum output behaves exactly like an Exclusive-OR gate: it is 1 only when the inputs are different. ($S = A \oplus B$).
 - The Carry output behaves exactly like an AND gate: it is 1 only when both inputs are 1. ($C = A \cdot B$).
 - *Limitation:* A Half Adder cannot accept a carry-in from a previous mathematical operation.
- **The Full Adder :** To add large multi-bit numbers, we need a circuit that can add three bits: A , B , and a Carry-In (C_{in}) from a previous column. This is the Full Adder. It still produces a Sum (S) and a Carry-Out (C_{out}). Internally, a Full Adder is generally constructed by connecting two Half Adders together using an OR gate for their carry outputs.

LOGIC DIAGRAM OF HALF ADDER

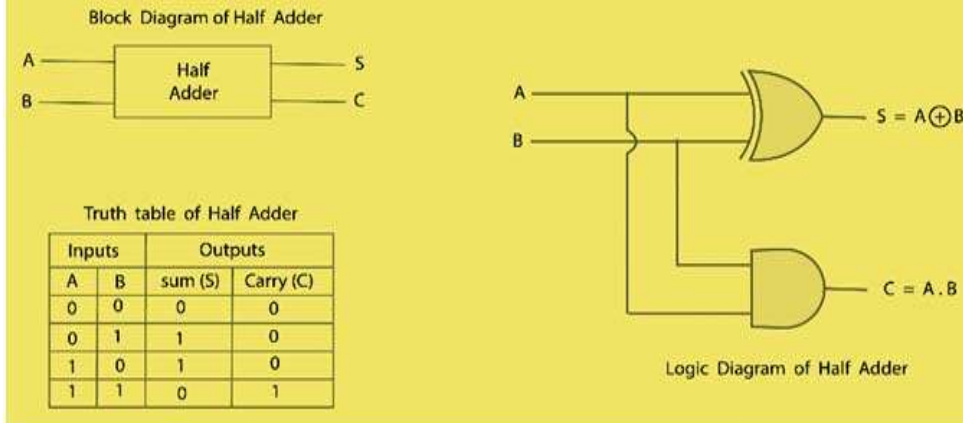


Figure 5.1 : Logic diagram of half adder.

- Subtractors** : Similar to adders, a **Half Subtractor** subtracts two bits to produce a Difference and a Borrow. A **Full Subtractor** subtracts two bits while also accounting for a borrow required by a previous lower-significant bit.

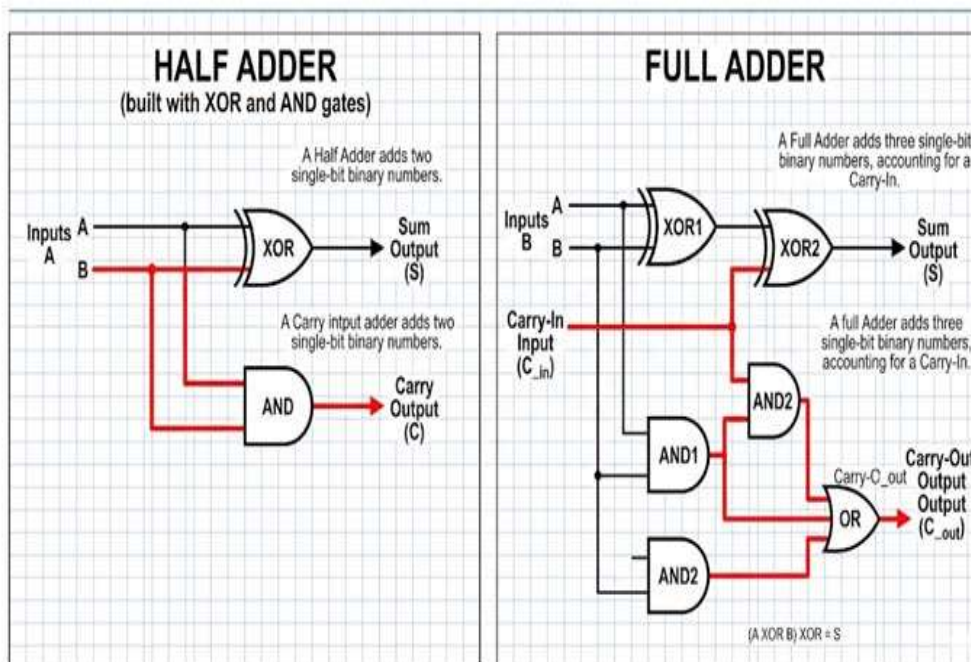


Figure 5.2 : Half Adder vs. Full Adder Logic Schematics.

5.4.2. Binary Parallel Adders

A single Full Adder is great, but it only adds one column of bits. How does a computer add two 4-bit numbers, like 1010 + 0011?

Engineers solve this by cascading multiple Full Adders together in a chain, creating a **Binary Parallel Adder**. In a 4-bit parallel adder, four

NOTES

Full Adders are placed side-by-side. The C_{out} of the first adder is wired directly into the C_{in} of the second adder, the second feeds the third, and so on.

When the input numbers are applied, the first bit adds and "ripples" its carry to the next stage, which calculates and ripples its carry to the next. Because the carry signal has to propagate through the chain like a wave, this specific combinational design is famously known as a **Ripple Carry Adder**.

☒ **Check Your Progress 6.1.**

Notes : (a) Space is given below for your answer.
(b) Check your answer with the one given at the end of this unit.

- 1. What logic gate generates the Sum (S) output in a Half Adder?
.....
.....
.....
- 2. Why is a standard Binary Parallel Adder often referred to as a "Ripple Carry Adder"?
.....
.....
.....

5.5. SUMMARY

In this unit, we took a massive leap from abstract logic gates to functional computer hardware by introducing Combinational Circuits. We defined a combinational circuit as a logic system where the current output is determined entirely and instantly by the current input values, with absolutely no memory of past events or feedback loops. This distinct "memoryless" behavior separates it from the sequential circuits we will study later.

We then applied this concept to computer arithmetic. We explored the Half Adder, a simple circuit utilizing an XOR gate for the Sum and an AND gate for the Carry, designed to add two single bits. Recognizing its limitation in handling carried numbers from previous columns, we introduced the Full Adder, which accommodates three inputs (A, B, and C_{in}). Finally, we discovered how the processing power of a computer is scaled up by cascading these individual Full Adders side-by-side to create a Binary Parallel Adder (or Ripple Carry Adder). This ingenious combinational architecture allows digital systems to add

complex, multi-bit binary numbers almost instantaneously, forming the absolute backbone of the CPU's calculating power.

NOTES

5.6. GLOSSARY

- **Combinational Circuit:** A digital logic circuit whose output depends solely on the present combination of inputs.
- **Half Adder:** An arithmetic combinational circuit that adds two single binary digits and produces a sum and a carry bit.
- **Full Adder:** An arithmetic combinational circuit that adds three input bits (two significant bits and a previous carry) to produce a sum and a carry output.
- **Parallel Adder:** A digital circuit formed by cascading multiple full adders to compute the sum of two multi-bit binary numbers simultaneously.
- **Ripple Carry:** The phenomenon in parallel adders where the carry bit must sequentially propagate from the lowest significant bit through to the highest.

5.7. ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 5.1.

1. **No**, the output of a combinational circuit strictly depends on the *current* inputs only.
2. **Memory elements** (or feedback loops) are present in sequential logic but absent in combinational logic.

Check Your Progress 5.2.

1. The **XOR (Exclusive-OR) gate** generates the Sum.
2. It is called a Ripple Carry Adder because the carry signal from the lowest bit must "ripple" (propagate sequentially) through every single full adder stage to reach the final output.

5.8. SUGGESTED READINGS – OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectionline.com/User/UnitUrls?unitId=3829	

5.9. REFERENCE

1. Mano, M. M., & Ciletti, M. D. (2013). *Digital Design*. Pearson.

2. Tocci, R. J., Widmer, N. S., & Moss, G. L. (2017). *Digital Systems: Principles and Applications*. Pearson.

5.10. EXERCISE

PART – 1

A. Multiple Choice Questions

1. The primary characteristic of a combinational logic circuit is that:
 - (a) It uses a clock signal to synchronize operations.
 - (b) Its output depends only on the present inputs.
 - (c) It contains flip-flops to store data
 - (d) Its output loops back to the input.
2. A Half Adder consists of which two logic gates?
 - (a) OR and NOT
 - (b) AND and NAND
 - (c) XOR and AND
 - (d) XOR and OR
3. How many inputs does a Full Adder have?
 - (a) 2
 - (b) 3
 - (c) 4
 - (d) 1
4. In a 4-bit Ripple Carry Adder, how many Full Adders are required?
 - (a) 2
 - (b) 4
 - (c) 8
 - (d) 16
5. Why can't a series of Half Adders be used to add two 4-bit numbers?
 - (a) They process data too slowly.
 - (b) They consume too much electrical power.
 - (c) They do not have an input pin to receive a Carry from a previous bit.
 - (d) They cannot produce a Sum output.

Answers with explanations:

1. Answer : (b)

Explanation : Combinational circuits are "memoryless." Their state is dictated 100% by whatever signals are currently present at their input terminals.

2. **Answer : (c)**

Explanation : The Sum is handled by the XOR gate (detecting when bits are different, e.g., $1+0=1$), and the Carry is handled by the AND gate (detecting when both bits are 1, e.g., $1+1=10$).

3. **Answer : (b)**

Explanation : A Full Adder requires three inputs: Input A, Input B, and a Carry-In C_{in} from the previous mathematical column.

4. **Answer : (b)**

Explanation : A 4-bit parallel adder requires exactly one adder stage per bit column, meaning 4 Full Adders are cascaded together.

5. **Answer : (c)**

Explanation : When adding multi-bit numbers, column 1 might generate a carry that must be added to column 2. Since Half Adders lack a C_{in} pin, they cannot process this incoming carry, making them useless for multi-bit addition.

B. Short answer type questions

1. Define the term "Ripple Carry" in the context of binary adders.
2. Write the Boolean logic equation for the Sum S output of a Half Adder.
3. What is the fundamental difference between a Half Subtractor and a Full Subtractor?
4. Briefly explain why combinational circuits do not require clock signals.

PART – 2

C. Long answer type question (200 words)

1. Compare and contrast Combinational logic with Sequential logic. Focus on the concepts of "memory," "feedback," and "current state versus past state" behavior. Provide one real-world analogy for each.
2. Describe the construction of a Full Adder. Explain how you can build a single Full Adder using two Half Adders and an OR gate. Why is this specific configuration necessary for CPU arithmetic?
3. Discuss the inherent delay problem associated with Ripple Carry Adders. When cascading many Full Adders together (e.g., a 32-bit adder), why does the final bit take longer to compute than the first bit?

NOTES**D. Long answer type question (300 words)**

1. Provide a comprehensive breakdown of the Half Adder circuit. Draw out the truth table showing inputs \$A\$ and \$B\$, alongside outputs Sum (\$S\$) and Carry (\$C\$). Systematically explain how the Boolean logic for the Sum maps perfectly to an XOR gate, while the logic for the Carry maps perfectly to an AND gate. Discuss the practical limitations of the Half Adder that ultimately led engineers to develop the Full Adder.
2. Analyze the architecture of a 4-bit Binary Parallel Adder. Explain the flow of data from the moment the input vectors (e.g., \$A_3A_2A_1A_0\$ and \$B_3B_2B_1B_0\$) are applied to the circuit. Detail the critical path of the Carry signal \$C_{out}\$ to \$C_{in}\$ as it propagates from the Least Significant Bit (LSB) up to the Most Significant Bit (MSB). Conclude by discussing how this cascading hardware configuration successfully mimics the traditional "pen and paper" method of human addition.
3. You are tasked with designing the logic for a smart home security alarm. The alarm (Output) should trigger if the System is Armed (Input A) AND either the Window is Broken (Input B) OR the Door is Opened (Input C). First, determine if this is a combinational or sequential circuit and justify your answer. Second, write the Boolean expression for this alarm logic. Finally, explain how you would physically construct this circuit using basic logic gates, detailing the data path from the sensors to the alarm bell.

UNIT – 6

COMBINATIONAL CIRCUITS: MULTIPLEXER AND DEMULTIPLEXER

NOTES

UNIT STRUCTURE

- 6.1. Introduction**
- 6.2. Objective**
- 6.3. Multiplexer and Demultiplexer**
 - 6.3.1. Functional Overview**
 - 6.3.2. Applications in Digital Circuits**
- 6.4. Applications in Digital Circuits**
 - 6.4.1. Data Routing and Selection**
- 6.5. Summary**
- 6.6. Glossary**
- 6.7. Answers to check your progress**
- 6.8. Suggested readings - OER**
- 6.9. Reference**
- 6.10. Exercise**

6.1. INTRODUCTION

In the previous unit, we built Arithmetic Combinational Circuits, which act as the calculating brain of the computer. However, a computer does not just perform math; it spends a massive amount of its time managing the flow of information. Imagine the motherboard of a modern computer: you have a mouse, a keyboard, a hard drive, a microphone, and a webcam, all trying to send data to the Central Processing Unit (CPU).

If we wired every single peripheral directly into the CPU with its own dedicated data lanes, the processor would need thousands of microscopic pins, and the motherboard would be an impossibly tangled, expensive mess of copper wires. We need a way to control this traffic. We need digital "traffic cops."

In this unit, we will explore specific data-routing combinational circuits designed to solve this exact problem: the Multiplexer and the Demultiplexer. We will learn how a Multiplexer acts like a funnel, taking multiple streams of input data and intelligently filtering them down to one single output wire. We will also examine its reverse counterPART,

the Demultiplexer, which takes a single stream of data and routes it to multiple specific destinations. By understanding the math behind "Select Lines," you will see how engineers use these components to save space, reduce costs, and efficiently route data across complex digital systems. Let us dive into the logistics of digital data flow!

6.2. OBJECTIVES

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

- Define the primary functions of Multiplexers (MUX) and Demultiplexers (DEMUX).
- Evaluate the critical role of "Select Lines" in routing digital data.
- Calculate the mathematical relationship between input lines and select lines using the 2^n rule.
- Explain the internal logic gate design of basic MUX and DEMUX circuits.
- Identify real-world applications of data routing, such as Time Division Multiplexing (TDM) and pin-count reduction in hardware.

6.3. MULTIPLEXER AND DEMULTIPLEXER

In digital systems, multiplexers and demultiplexers are categorized as **data transmission** circuits. These components act as digital switches that route information between various points in a circuit, allowing multiple signals to share a single transmission line or resource.

6.3.1. Functional Overview

I. Multiplexer (MUX) A **multiplexer**, abbreviated as **MUX**, is a combinational circuit that selects one of many input signals and directs it to a single output line. The selection of a specific input is controlled by a set of **select lines** (also called control lines).

- **Selection Logic:** If there are n select lines, the multiplexer can handle a maximum of 2^n **data inputs**.
- **Example: 4-to-1 Multiplexer:** This device comprises 4 input bits (d_0 through d_3), 1 output bit (q), and **2 control bits** (A and B).
 - When the control bits $AB=00$, the internal logic enables the gate associated with d_0 , allowing that specific input to transmit to the output while restricting all other inputs.
- **Purpose:** The primary objective of a MUX is to perform **high-speed switching** in digital applications.

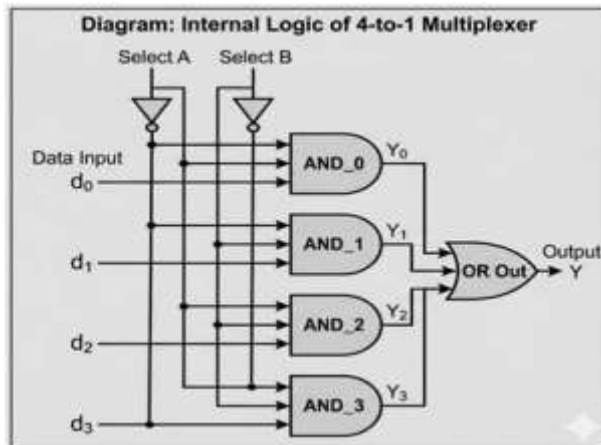


Fig. 6.1 : Internal logic of 4to1 multiplexer.

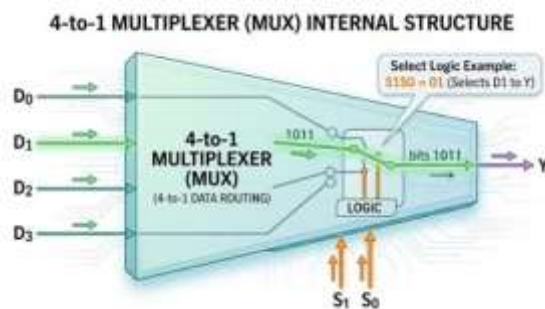


Fig. 6.2 : 4-to-1 Multiplexer (MUX) Architecture.

II. Demultiplexer (De-MUX)

The Demultiplexer performs the exact reverse operation. It is a "One-to-Many" device. It receives information on a single input line and transmits this same information over one of several possible output lines. Imagine a package sorting facility. One conveyor belt of packages comes in (the single input), and the automated sorter looks at the zip code (the select lines) to push the package down one of many specific delivery chutes (the multiple outputs).

- **The Math:** A DEMUX follows the same mathematical rule. A 1-to-4 DEMUX takes 1 input and routes it to 4 outputs, requiring exactly 2 select lines to dictate which output receives the data.

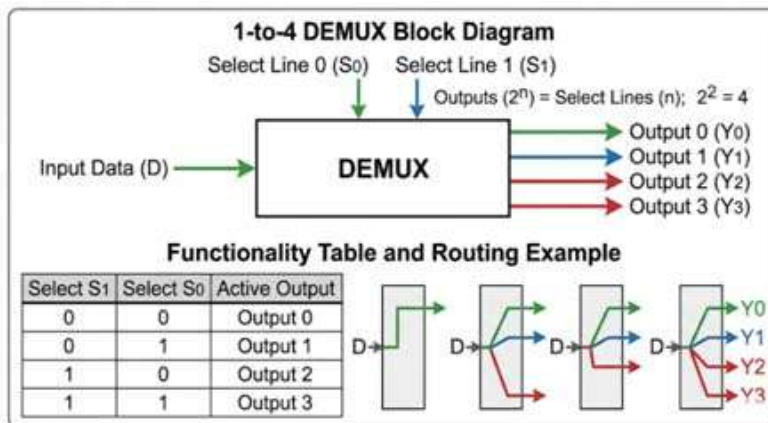


Fig. 6.3 : 1-to-4 DEMUX.

NOTES

Feature	Multiplexer (MUX)	Demultiplexer (DEMUX)
Primary Function	Data Selector	Data Distributor
Logic Structure	Many-to-One	One-to-Many
Select Lines Role	Chooses which input to read	Chooses which output to write to
Common Application	Sharing a single transmission line	Routing a signal to a specific chip

Table 6.1: MUX vs. DEMUX.

6.3.2. Design and Select Lines

The magic of both circuits lies in a set of control inputs known as **Select Lines**. These lines act as the steering wheel for the data.

- **The 2ⁿ Math:** The number of select lines is strictly tied to the number of data lines. If you have \$n\$ select lines, you can control up to 2ⁿ data lines.
 - For a **4-to-1 MUX**(2² = 4), you need exactly 2 select lines (let's call them *S₁* and *S₀*).
 - If the processor sends the binary code 00 to the select lines, Input 0 is connected to the output.
 - If the processor sends 10, Input 2 is connected to the output.

Internally, a Multiplexer is built using a combination of AND, OR, and NOT gates. Each data input is paired with its own AND gate. The select lines are wired (using NOT gates where necessary) so that only *one* AND gate is fully enabled at a time. The outputs of all these AND gates then flow into a single OR gate, which provides the final output.

✓ Check Your Progress 6.1.

- Notes :** (a) Space is given below for your answer.
(b) Check your answer with the one given at the end of this unit.

1. How many select lines are required for an 8-to-1 Multiplexer?
.....
.....
.....
2. Which circuit is known as a "Data Distributor"?
.....
.....
.....



Activity 6.1. ROUTING LOGIC IN YOUR LIVING ROOM

NOTES

Title : The "Many-to-One" Physical Audit

Objective : To identify the practical application of multiplexing in everyday household electronics.

Material Needed : A notebook, a pen, and access to your television or home audio system.

Activity Description:

1. **The Investigation :** Go to your primary television or stereo receiver. Look at the back or side panel where cables plug in (HDMI 1, HDMI 2, AV In, USB). Count how many physical input sources are currently plugged in. Write this number down.
2. **Identify the MUX Output :** Turn on the device. Notice that regardless of how many things are plugged in, the screen or speakers (the Output) can only process one source at a time.
3. **Identify the Select Lines :** Pick up the remote control and press the "Source" or "Input" button to switch from HDMI 1 to HDMI 2.

Reflection :

Write a brief reflection connecting this experience to the unit. When you press the "Source" button on your remote, you are essentially acting as the human operator changing the binary state of the "Select Lines." Your remote control sends a code (like 01 for HDMI 2), which forces the internal Multiplexer logic gates to block all other signals and route only HDMI 2 to the screen. You use MUX circuits every single day!

6.4. APPLICATIONS IN DIGITAL CIRCUITS

Multiplexers and demultiplexers are essential for managing data flow in complex architectures:

1. **Data Bus Sharing :** Multiplexing is used to allow multiple external sources to share a common data bus in a computer system. Latches are often placed between the external source and the data bus to temporarily store data when the bus is occupied by another source.
2. **Communication Systems :** They are used to transmit multiple signals over a single line (Time Division Multiplexing), maximizing the efficiency of the transmission medium.
3. **High-Speed Routing :** In digital applications, they function as electronic switches that can change data paths almost instantaneously, which is critical for CPU instruction processing and memory access.
4. **Function Realization:** Multiplexers can also be used to implement Boolean logic functions directly without needing a complex arrangement of individual gates.

NOTES

6.4. APPLICATIONS IN DIGITAL CIRCUITS

Multiplexers and Demultiplexers are not just theoretical constructs; they are essential for the physical manufacturing and operation of modern technology.

6.4.1. Data Routing and Selection

- **Time Division Multiplexing (TDM)** : This is one of the most vital applications of MUX/DEMUX circuits, PARTICULARly in telecommunications. Imagine a submarine fiber-optic cable crossing the Atlantic Ocean. It is incredibly expensive to lay one cable, so telecom companies cannot afford to give every phone call its own physical wire. Instead, they use TDM. A MUX rapidly cycles its select lines, taking a tiny slice of Audio A, then a tiny slice of Audio B, then Audio C, and sending them one after the other down the single cable. At the other side of the ocean, a synchronized DEMUX distributes the slices back into separate audio channels. The switching happens so fast that human ears never hear the interruption.
- **Pin Count Reduction** : Integrated Circuits (ICs), like microprocessors, are strictly limited by physical space. If an IC needs to monitor 16 different internal sensors, adding 16 physical pins to the chip makes it large and expensive. By placing a 16-to-1 MUX inside the chip, the processor only needs 1 data pin and 4 select pins (1+4 = 5 pins total) to monitor all 16 sensors, saving valuable physical real estate.
- **Computer Memory (RAM) Routing** : When the CPU wants to read a specific piece of data from the RAM, it uses multiplexers to route the requested data from the massive grid of memory cells down to the specific data bus leading back to the processor.

☑ Check Your Progress 6.2.

Notes : (a) Space is given below for your answer.
(b) Check your answer with the one given at the end of this unit.

1. What does TDM stand for in telecommunications?
.....
.....
.....
2. Briefly explain how a MUX saves physical space on a microchip.
.....
.....
.....

6.5. SUMMARY

In this unit, we explored the critical role of data-routing combinational circuits, specifically Multiplexers (MUX) and Demultiplexers (DEMUX). We learned that as digital systems grow more complex, it becomes physically and economically impossible to provide dedicated wires for every single component.

To solve this, the Multiplexer acts as a "many-to-one" data selector, funneling multiple input streams down to a single output line. We discovered that the internal logic is controlled by "Select Lines," which dictate exactly which input is granted access based on the mathematical rule of 2^n . We then examined the Demultiplexer, a "one-to-many" data distributor that takes a single stream of information and routes it to a specific destination. Finally, we looked at real-world applications, noting how synchronized MUX and DEMUX circuits enable Time Division Multiplexing (TDM) to share expensive communication cables, and how these circuits drastically reduce the required number of physical pins on a microprocessor. These components are the unsung heroes of digital logistics, ensuring data gets exactly where it needs to go.

6.6. GLOSSARY

- **Multiplexer (MUX):** A combinational logic circuit that selects one of many input signals and forwards it to a single output line. Also known as a Data Selector.
- **Demultiplexer (DEMUX):** A combinational circuit that routes a single input signal to one of several output lines. Also known as a Data Distributor.
- **Select Lines:** The control inputs on a MUX or DEMUX that determine which specific input/output path is currently active.
- **Time Division Multiplexing (TDM):** A technique for transmitting multiple signals over a single communication line by assigning each signal a rapid, alternating time slot.

6.7 ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 6.1

1. A Multiplexer is commonly called a **Data Selector**.
2. An 8-to-1 Multiplexer requires **3 select lines** (because $2^3 = 8$).

Check Your Progress 6.2

1. TDM stands for **Time Division Multiplexing**.
2. A MUX saves space by allowing multiple internal signals to share a single output pin, drastically reducing the total number of physical metal pins required on the exterior of the chip.

NOTES

NOTES

6.8. SUGGESTED READINGS - OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectonline.com/User/UnitUrls?unitId=3830	

6.9. REFERENCE

1. Mano, M. M., & Ciletti, M. D. (2013). *Digital Design*. Pearson.
2. Floyd, T. L. (2015). *Digital Fundamentals*. Pearson Education.

6.10. EXERCISE**PART – 1****A. Multiple Choice Questions**

1. A circuit that routes data from a single input to one of many outputs is called a:
 - (a) Multiplexer
 - (b) Demultiplexer
 - (c) Full Adder
 - (d) Half Subtractor
2. If a Multiplexer has 4 select lines, how many data inputs can it manage?
 - (a) 4
 - (b) 8
 - (c) 16
 - (d) 32
3. What internal logic gate is primarily used in a MUX to "enable" or "disable" specific input lines based on the select line status?
 - (a) XOR Gate
 - (b) NOR Gate
 - (c) AND Gate
 - (d) NOT Gate
4. In Time Division Multiplexing (TDM), what allows multiple signals to share the same wire without getting mixed up?
 - (a) They are sent at different voltages.
 - (b) They are sent in alternating time slots.
 - (c) They are converted to analog signals.
 - (d) They are compressed into a ZIP file.
5. A 4-to-1 MUX has how many final output lines?

- (a) 1
- (b) 2
- (c) 4
- (d) 8

NOTES

Answers with explanations:1. **Answer : (b)**

Explanation : A Demultiplexer (DEMUX) is a "one-to-many" device that distributes a single input across multiple possible outputs.

2. **Answer : (c)**

Explanation : Using the required formula $2^n = \text{inputs}$. If $n = 4$, then $2^4 = 16$ data inputs.

3. **Answer : (c)**

Explanation : The AND gate acts like a digital valve. If one input to an AND gate is 0 (disabled by the select line), the output is strictly 0. If it is 1 (enabled), the data passes through.

4. **Answer : (b)**

Explanation : TDM rapidly cycles the select lines of a MUX, slicing the signals up and sending them one by one in precise time slots so they can share a single physical cable.

5. **Answer : (a)**

Explanation : By definition, a Multiplexer is a "many-to-one" circuit. Regardless of how many inputs it has (4, 8, 16, or 32), it always funnels down to exactly 1 output line.

B. Short answer type questions

1. Define the relationship between select lines and data inputs in a DEMUX.
2. Why is a Multiplexer called a "Data Selector"?
3. State one practical reason why engineers try to minimize the "pin count" on a microprocessor.
4. Draw a conceptual block diagram of a 2-to-1 Multiplexer, labeling inputs, select lines, and output.

PART – 2**C. Long answer type question (200 words)**

1. Describe the internal logical operation of a 4-to-1 Multiplexer. Detail the role of the 4 data inputs (D_0, D_1, D_2, D_3), the 2 select lines (S_1, S_0), and the single output (Y). Use a basic truth table to explain how the binary state of the select lines strictly dictates which data input reaches the output.
2. Explain the concept of Time Division Multiplexing (TDM) conceptually. How does rapidly switching the select lines on a MUX allow multiple phone calls to travel over a single

NOTES

transatlantic submarine cable without interfering with each other?

3. Discuss the inherent symmetry between a Multiplexer and a Demultiplexer. Why are these two circuits almost always used together in long-distance communication systems?

PART – 3**D. Long answer type question (300 words)**

1. Provide a comprehensive analysis of how Data-Routing Combinational Circuits reduce hardware complexity. Imagine designing a complex printed circuit board without using Multiplexers. Detail the physical wiring and manufacturing consequences of attempting to route 32 separate sensor signals directly into a microprocessor. Then, explain step-by-step how inserting a 32-to-1 Multiplexer drastically reduces the required pin count on the Central Processing Unit, making the board smaller, cheaper, and less prone to wire-routing errors.
2. Deconstruct the logic of a 1-to-4 Demultiplexer. Draw the truth table featuring the two select lines (S_1, S_0), the single data input (D), and the four output lines (Y_0, Y_1, Y_2, Y_3). Explain the Boolean logic required to build this circuit using NOT gates and AND gates. Discuss a practical application for this specific DEMUX, such as a computer routing a single stream of audio data to one of four different speaker zones in a smart house.
3. Analyze a complete end-to-end digital communication channel utilizing both MUX and DEMUX. Consider a security room with 8 different digital security cameras (inputs) but only one single long-distance coaxial cable connecting the building to a remote recording server. Detail how an 8-to-1 MUX at the camera site, synchronized perfectly with a 1-to-8 DEMUX at the server site, successfully transmits all 8 video feeds over the single wire. Discuss the vital importance of the clock signal ensuring that the select lines on *both* ends change at the exact same time.

UNIT – 7

ENCODERS AND DECODERS

NOTES

UNIT STRUCTURE

- 7.1. Introduction
- 7.2. Unit Objective
- 7.3. Encoder and Decoder
 - 7.3.1. Types and Functions
 - 7.3.2. Implementation Examples
- 7.4. Code Converters
- 7.5. Summary
- 7.6. Glossary
- 7.7. Answers to check your progress
- 7.8. Suggested readings - OER
- 7.9. Reference
- 7.10. Exercise

7.1. INTRODUCTION

In our previous unit, we looked at Multiplexers and Demultiplexers, the "traffic cops" that route data through shared wires. But routing data is only half the battle. A digital system also needs to *translate* data.

Think about how you interact with a computer. You press a key with a decimal number '9' on it, or an alphabet letter 'A'. You understand these symbols perfectly. The Central Processing Unit (CPU), however, only understands binary logic (0s and 1s). How does your physical button press turn into a language the computer can process? And conversely, when the computer wants to show you the result of a calculation, how does it translate its internal binary code back into a readable number on a digital screen?

This critical translation is handled by two specialized combinational circuits: **Encoders** and **Decoders**. In this unit, we will explore how Encoders act as the computer's entry point, compressing human-friendly inputs into dense binary codes. We will then examine Decoders, which act as the exit point, expanding those binary codes back into actionable outputs or readable displays. Finally, we will look at Code Converters, which are specialized circuits that translate one type of machine code into another to ensure different PARTs of a system can communicate flawlessly. Let's start decoding the digital language!

NOTES

7.2. OBJECTIVES

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

- Define the primary functions of Encoders and Decoders in digital systems.
- Differentiate between standard encoders and priority encoders.
- Analyze the mathematical relationship between inputs and outputs (2^n to n and vice versa).
- Explain practical implementation examples, such as keyboard encoding and memory decoding.
- Evaluate the role of Code Converters in bridging different digital formats, such as Binary to Gray code.

7.3. ENCODER AND DECODER

Encoders and Decoders are inverse operations of one another. They are the essential translators sitting between the human operator and the machine's processor.

7.3.1. Types and Functions

- **The Encoder:** An encoder is a combinational circuit that converts an active input signal into a coded binary output. It is a "many-to-fewer" device. It typically takes up to 2^n input lines and compresses them into n output lines.
 - *Example:* An **8-to-3 Octal to Binary Encoder** has 8 input lines (representing numbers 0 through 7) and 3 output lines. If the input line for '5' is activated, the encoder outputs the binary code 101.
 - **Priority Encoder:** A standard encoder has a major flaw: if a user accidentally presses two inputs at the exact same time, the output generates garbage logic. A **Priority Encoder** solves this by assigning a rank to each input. If inputs '3' and '7' are pressed simultaneously, the priority encoder ignores '3' and only outputs the binary code for '7'.
- **The Decoder:** A decoder does the exact opposite. It converts coded binary information into multiple recognizable output lines. It is a "fewer-to-many" device, taking n inputs and expanding them to a maximum of 2^n outputs.
 - *Example:* A **3-to-8 Decoder** takes a 3-bit binary input (like 110) and activates exactly one of its 8 output lines (in this case, line 6). All other output lines remain inactive.

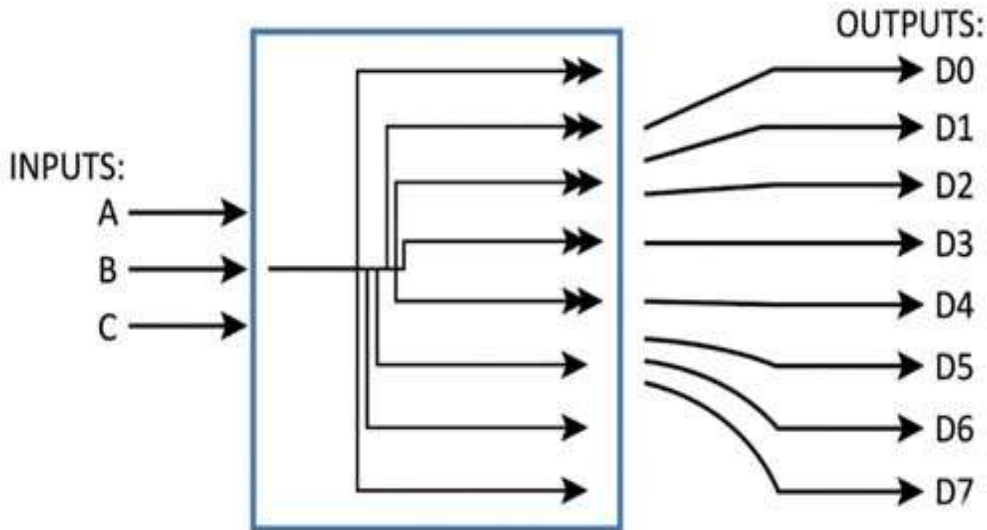


Fig. 7.1 : 3-to-8 Line Binary Decoder.

NOTES

7.3.2. Implementation Examples

- **Keyboard Encoding:** When you type on a calculator, pressing a decimal digit (0-9) sends a signal to a **Decimal-to-BCD Encoder**. This circuit immediately translates your single keystroke into a 4-bit Binary Coded Decimal (BCD) so the processor can perform math on it.
- **Memory Address Decoding:** Inside a computer, RAM consists of millions of memory locations. When the CPU wants to read data, it sends out a binary address. A decoder receives this binary address and activates the one single, specific wire connected to the correct memory chip. Without decoders, addressing memory would be physically impossible.

☒ Check Your Progress 7.1.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. What type of encoder prevents errors when two input buttons are pressed simultaneously?
.....
.....
.....
2. A decoder with 4 input lines can have a maximum of how many output lines?
.....
.....
.....

NOTES

**Activity 7.1. DECODING YOUR APPLIANCES**

Title : The 7-Segment Decoder Hunt

Objective : To recognize the physical manifestation of digital decoders in everyday life.

Material Needed : A notebook and access to digital appliances (microwave, digital clock, oven).

Activity Description :

1. **Find a Digital Display :** Look at the digital clock on your microwave or a standard digital alarm clock. Look closely at the numbers.
2. **Observe the Segments :** You will notice that each number is actually made up of 7 distinct, straight LED bars (segments) arranged in a figure-eight pattern.
3. **Analyze the Logic :** The internal computer of the microwave doesn't "know" what a number looks like. It just holds a 4-bit binary value (e.g., 0010 for the number 2). To show you a '2', a specific circuit must read 0010 and decide to turn on the top, top-right, middle, bottom-left, and bottom segments, while keeping the others off.

Reflection :

Write down your observations. The circuit performing this visual magic is called a **BCD to 7-Segment Decoder**. It acts as the ultimate bridge between machine logic (binary) and human readability (visual numbers). Every time you look at a digital clock, you are looking at the direct output of a combinational decoder!

7.4. CODE CONVERTERS

Sometimes, the translation isn't between humans and computers, but between two different digital subsystems that "speak" different binary dialects. This is where **Code Converters** come in.

A code converter is a specialized combinational logic circuit designed to change data presented in one binary code format into another binary code format.

- **Binary to Gray Code Converter:** In standard binary, incrementing from 3 (011) to 4 (100) requires three separate bits to flip simultaneously. In mechanical systems (like robotic arms or rotating shafts), if these bits don't flip at the *exact* same microsecond, the system might read an erratic intermediate value, causing a fatal error. **Gray Code** is a special binary sequence where *only one single bit* changes state between any two successive numbers. Engineers use a Binary-to-Gray Code Converter to ensure smooth, error-free data transmission in mechanical sensors.
- **BCD to Excess-3 Converter:** This converts standard BCD into "Excess-3" code (where the value 3 is added to every decimal digit

before converting to binary). Excess-3 is often used in older arithmetic circuits because it makes subtraction operations using "complements" much easier for the hardware.

Code converters are simply made by mapping the truth table of the input code to the truth table of the desired output code, using Karnaugh Maps (K-Maps) to find the simplified logic, and wiring up the necessary AND, OR, and XOR gates.

NOTES**☑ Check Your Progress 7.2.**

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. What is the primary characteristic of Gray Code that makes it useful for mechanical sensors?

.....

2. What type of circuit is required to change standard binary numbers into Gray code?

.....

7.5. SUMMARY

In this unit, we explored the critical role of translation in digital electronics through Encoders, Decoders, and Code Converters. We defined Encoders as combinational circuits that compress active, human-triggered inputs into dense binary codes, serving as the essential first step when data enters a computer system. We also noted the importance of Priority Encoders, which use built-in logic to handle simultaneous inputs gracefully without crashing the system.

Conversely, we examined Decoders, which expand dense binary instructions back into multiple individual control lines. Decoders are the backbone of hardware management, translating binary addresses to select specific memory banks or lighting up 7-segment displays for human viewing. Finally, we looked at Code Converters, specialized circuits that translate one form of binary into another (such as Binary to Gray Code) to ensure different mechanical and digital subsystems can communicate without logic errors. Together, these circuits ensure that data is always in the correct format, at the correct time, for both the machine and the user.

7.6. GLOSSARY

- **Encoder** : A digital circuit that converts multiple input lines into a smaller number of coded binary output lines.
- **Decoder** : A digital circuit that converts a coded binary input into multiple distinct output lines.
- **Priority Encoder** : A type of encoder that responds to the highest-priority input when two or more inputs are active simultaneously.
- **Code Converter** : A logic circuit that translates data from one digital code format to another (e.g., Binary to Gray code).
- **Gray Code** : A binary numeral system where two successive values differ in only one bit, used to prevent errors in electro-mechanical switches.

7.7. ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 7.1

1. A **Priority Encoder**.
2. Using the 2^n rule, $2^4 = 16$. It can have a maximum of **16 output lines**.

Check Your Progress 7.2

1. Its primary characteristic is that **only one bit changes state** at a time between consecutive numbers, which prevents reading errors during mechanical transitions.
2. A **Code Converter** (specifically, a Binary to Gray Code Converter).

7.8. SUGGESTED READINGS - OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectonline.com/User/UnitUrls?unitId=3831	

7.9. REFERENCE

1. Mano, M. M., & Ciletti, M. D. (2013). *Digital Design*. Pearson.
2. Tocci, R. J., Widmer, N. S., & Moss, G. L. (2017). *Digital Systems: Principles and Applications*. Pearson.

7.10. EXERCISE

PART – 1

A. Multiple Choice Questions

1. A combinational circuit that changes a decimal input (0-9) from a keypad into a 4-bit binary output is called a(n):
 - (a) Multiplexer
 - (b) Decoder
 - (c) Encoder
 - (d) Demultiplexer

2. If the inputs 2 and 5 are triggered simultaneously on a Priority Encoder, which binary code will be generated at the output?
 - (a) The code for 2
 - (b) The code for 5
 - (c) The code for 7 (2+5)
 - (d) An error code of all zeros
3. A 4-to-16 line decoder has how many input lines?
 - (a) 2
 - (b) 4
 - (c) 8
 - (d) 16
4. Which code converter is commonly used to prevent errors in rotating mechanical shafts?
 - (a) BCD to 7-Segment
 - (b) Binary to Excess-3
 - (c) Gray to Binary
 - (d) Binary to Gray Code
5. The primary difference between a Multiplexer and a Decoder is:
 - (a) A MUX routes data, while a Decoder activates specific lines based on a binary code.
 - (b) A MUX expands data, while a Decoder compresses data.
 - (c) A Decoder requires a clock signal, while a MUX does not.
 - (d) There is no difference; they are the same circuit.

Answers with explanations :

1. **Answer : (c)**
Explanation : An encoder takes a single active input (like a keypress) and compresses it into a smaller binary code output.
2. **Answer : (b)**
Explanation : A priority encoder always prioritizes the input with the highest numerical value, ignoring the lower values to prevent logic corruption.
3. **Answer : (b)**
Explanation : A decoder expands n inputs to 2^n outputs. Here, 4 inputs expand to 2^4 (16) outputs.
4. **Answer : (d)**
Explanation : Because Gray code only changes one bit at a time, converting standard binary into Gray code is essential for reading the **position** of mechanical sensors safely.
5. **Answer : (a)**
Explanation : Multiplexer acts as a switch passing a data stream from input to output. A Decoder acts as a translator, turning a binary word into a physical hardware activation.

B. Short answer type questions

1. Define the primary function of a Code Converter.
2. What is the mathematical relationship between inputs and outputs for a standard Encoder?

NOTES

3. Briefly explain why an ordinary encoder fails if two buttons are pressed at once.
4. Give one practical example of where a Decoder is used inside a computer.

PART – 2**C. Long answer type question (200 words)**

1. Describe the necessity of a Priority Encoder over a standard encoder. Use a real-world analogy, such as an elevator control panel where multiple floor buttons might be pushed simultaneously, to explain how priority logic resolves conflicts.
2. Explain how a BCD to 7-Segment Decoder functions. Discuss the transition from the internal 4-bit machine language to the external visual representation that humans can read.
3. Discuss the concept of Gray Code. Why is standard binary problematic for electro-mechanical position sensors (like a robotic arm joint), and how does Gray Code solve this issue?

PART – 3**D. Long answer type question (300 words)**

1. Provide a comprehensive comparative analysis of Encoders and Decoders. Detail their architectural differences, focusing heavily on the flow of data ($2n$ to n vs. n to $2n$). Construct a scenario involving a user typing a number on a keyboard, the CPU processing it, and the screen displaying it. Map out exactly where the Encoder is used in this scenario, where the Decoder is used, and why both are absolutely necessary for the system to function.
2. Outline the step-by-step process of designing a Code Converter. Assume you are tasked with building a 3-bit Binary to Gray Code converter. Explain how you would start by drawing a truth table mapping the standard binary inputs (000 to 111) to their corresponding Gray code outputs. Discuss how you would use Karnaugh Maps to derive the simplified Boolean expressions for each output bit, and state which specific logic gate (e.g., XOR) dominates this specific conversion circuit.
3. Analyze the application of Decoders in computer memory addressing. Imagine a microprocessor trying to access a specific byte of data in a RAM module that contains 256 different memory locations. Explain how an 8-to-256 line Decoder is utilized in this process. Detail how the 8-bit binary address sent by the CPU acts as the input to the decoder, and how the decoder subsequently activates one—and strictly one—physical wire to unlock the correct memory cell. Discuss the hardware efficiency of this method.

UNIT – 8

DIGITAL CONTROL CIRCUITS AND SEQUENCING

NOTES

UNIT STRUCTURE

- 8.1. Introduction
- 8.2. Objective
- 8.3. Arithmetic Units Refresher
 - 8.3.1. Basics of Arithmetic Logic Units (ALUs)
- 8.4. Program Control
 - 8.4.1. Instruction Sequencing
 - 8.4.2. Control Circuit Functionality
- 8.5. Summary
- 8.6. Glossary
- 8.7. Answers to check your progress
- 8.8. Suggested readings - OER
- 8.9. Reference
- 8.10. Exercise

8.1. INTRODUCTION

In our previous units, we built the foundational blocks of a computer. We constructed Arithmetic Circuits (Adders and Subtractors) that give the computer the ability to calculate. We then built Data-Routing Circuits (Multiplexers and Demultiplexers) to direct traffic, and Translation Circuits (Encoders and Decoders) to bridge human input with machine binary.

However, a collection of adders, multiplexers, and decoders sitting on a motherboard does not automatically do anything. They are completely inert. If you want to add two numbers, how do the numbers actually get from the computer's memory into the Adder circuit? How does the Multiplexer know exactly *when* to switch its select lines? How does the system know whether to add, subtract, or just move data during any given microsecond?

This is where the magic of **Digital Control Circuits and Sequencing** comes into play. Think of all the combinational circuits we built as the musicians in an orchestra. They are highly skilled and capable of

NOTES

making beautiful music, but without a conductor, they will just play a chaotic mess of notes. The **Control Unit (CU)** is the conductor.

In this unit, we will first refresh our understanding of the Arithmetic Logic Unit (ALU)—the mathematical heart of the machine. Then, we will explore how the Control Unit reads software instructions, decodes them, and sequences a series of strict electrical signals to coordinate every single component on the motherboard. By the end of this unit, you will understand how static logic gates are sequenced together over time to create a living, breathing, computing machine.

8.2. OBJECTIVES

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

- Explain the fundamental architecture and role of the Arithmetic Logic Unit (ALU) within a microprocessor.
- Identify and explain the significance of ALU status flags (Carry, Zero, Sign, Overflow) in decision-making.
- Define the concept of Program Control and explain how a processor governs the flow of data.
- Deconstruct the Fetch-Decode-Execute cycle and map the journey of an instruction from memory to execution.
- Differentiate between Hardwired Control Units and Microprogrammed Control Units, evaluating the physical tradeoffs of each design.
- Appreciate how digital sequencing synchronizes thousands of disparate logic gates into a single, cohesive computing engine.

8.3. ARITHMETIC UNITS REFRESHER

Before we can understand how a computer controls its operations, we must clearly define what it is operating *on*. The central processing hub for all mathematical and logical operations is the Arithmetic Logic Unit.

8.3.1. Basics of Arithmetic Logic Units (ALUs)

The Arithmetic Logic Unit (ALU) is a highly complex combinational digital circuit that performs integer arithmetic and bitwise logical operations. It is the fundamental building block of the Central Processing Unit (CPU) found in everything from digital thermometers to supercomputers.

ALU V-SHAPED BLOCK DIAGRAM VISUAL AID

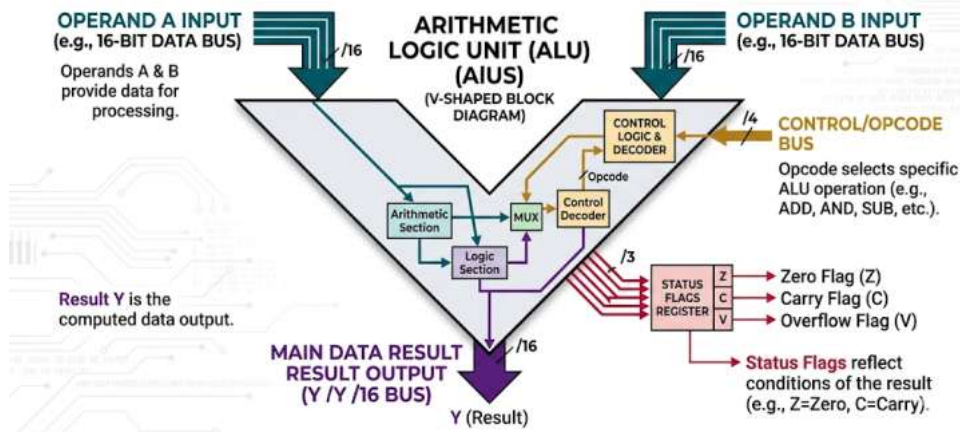


Fig. 8.1 : Standard ALU Block Diagram.

1. The Inputs and Outputs

An ALU typically has three distinct types of input and two distinct types of output :

- **Data Inputs (Operands):** The ALU receives two binary numbers (often called Operand A and Operand B). If it is a 32-bit ALU, each of these operands is a 32-bit wide bus.
- **Operation Code (Opcode):** This is a small set of control wires coming from the Control Unit that tells the ALU *what* to do with the operands. For example, an opcode of 001 might tell the ALU to ADD, while 101 might tell it to perform a bitwise AND.
- **Data Output (Result):** The resulting mathematical or logical answer is pushed out on this bus to be stored in a register or memory.
- **Status Flags:** This is a critically important output. These are individual wires that tell the rest of the processor about the *nature* of the result.

2. The Status Register (Flags)

The ALU doesn't just calculate answers; it provides context. The Status Flags are the primary way a computer makes decisions (like "If X is greater than Y, then do Z"). Common flags include:

- **Zero Flag (Z):** This wire becomes a logical 1 if the result of an ALU operation is exactly zero. (**Crucial** for checking if two numbers are equal by subtracting them).
- **Carry Flag (C):** This wire is set to 1 if an addition operation causes a carry-out beyond the maximum bit width of the ALU (e.g., adding two 8-bit numbers that result in a 9-bit answer).
- **Sign Flag (S) / Negative Flag (N):** In signed arithmetic, this flag simply mirrors the Most Significant Bit (MSB) of the result. If it's a 1, the answer is mathematically negative.

NOTES

- **Overflow Flag (V):** This wire is set if the result of an operation is too large (or too small) to be represented in the current signed bit format, indicating a fatal math error.

Opcode	Operation Category	Specific Function	Explanation
0000	Arithmetic	ADD	\$A + B\$ (Standard binary addition)
0001	Arithmetic	SUB	\$A - B\$ (Standard binary subtraction)
1000	Logical	AND	Bitwise logical AND of \$A\$ and \$B\$
1001	Logical	OR	Bitwise logical OR of \$A\$ and \$B\$
1010	Logical	XOR	Bitwise logical Exclusive-OR
1100	Shift/Rotate	SHL	Shift bits of \$A\$ left by 1 position

Table 8.1 : Typical ALU Operations.

Crucial Concept: The ALU is entirely combinational. It has no memory and makes no decisions on its own. It simply calculates whatever is on its input lines based on the opcode it receives at that exact microsecond. The intelligence of the computer lies in the circuit that feeds the ALU—the Control Unit.

☒ **Check Your Progress 8.1.**

Notes : (a) Space is given below for your answer.
(b) Check your answer with the one given at the end of this unit.

1. What does the Zero Flag (Z) indicate when it is set to 1?
.....
.....
.....
2. What are the three primary inputs required for an ALU to function?
.....
.....
.....

8.4. PROGRAM CONTROL

A computer program is nothing more than a long list of binary instructions stored in memory. The processor's job is to read these instructions one by one and execute them. This process is governed by the Program Control logic.

8.4.1. Instruction Sequencing

To sequence through a program without getting lost, the CPU relies on a set of internal memory cells called Registers.

- **The Program Counter (PC):** This is arguably the most important register in the computer. The Program Counter always holds the exact memory address of the *next* instruction that needs to be executed.
- **The Instruction Register (IR):** Once an instruction is fetched from main memory, it is temporarily stored here so the computer can figure out what it means.

The Fetch-Decode-Execute Cycle

Digital sequencing is entirely built around a relentless, repeating, three-step cycle dictated by the ticking of the system clock.

1. FETCH Stage:

- The Control Unit reads the memory address currently sitting in the Program Counter (PC).
- It sends this address over the Address Bus to the main memory (RAM).
- The RAM finds the binary instruction at that address and sends it back to the CPU over the Data Bus.
- The CPU saves this instruction into the Instruction Register (IR).
- *Crucial Step:* The Program Counter is instantly incremented by 1, so it is ready for the *next* cycle.

2. DECODE Stage:

- The binary instruction sitting in the IR is fed into a massive Decoder circuit inside the Control Unit.
- The Control Unit translates this dense binary code (e.g., 10010110) into physical electrical signals. It determines exactly what the instruction is asking (e.g., "Is this an ADD instruction? Is this a MOVE instruction?").
- It also determines where the required data is located (e.g., "I need the data from Register 1 and Register 2").

3. EXECUTE Stage:

- The Control Unit fires off the necessary electrical pulses to make the instruction happen.
- If it is a math instruction, it routes the data to the ALU, sets the ALU opcode to the correct math function, and routes the ALU's output back to a storage register.
- Once execution is complete, the cycle immediately loops back to step 1.

8.4.2. Control Circuit Functionality

We have established what the Control Unit does, but *how* is it physically built? There are two primary architectural philosophies for designing the internal logic of a Control Unit.

1. Hardwired Control Unit

NOTES

In a hardwired control unit, the logic is implemented using pure combinational circuits (AND gates, OR gates, NOT gates, Multiplexers, and Decoders) combined with sequential timing circuits (flip-flops and counters).

- **How it works :** The binary instruction from the Instruction Register is fed directly into a massive web of physical logic gates. The output of these gates directly connects to the control lines of the ALU, memory, and registers.
- **Advantages:** It is incredibly fast. Because it relies purely on electrical physics flowing through static logic gates, there is minimal delay. This is used in high-performance processors (like RISC architectures).
- **Disadvantages:** It is highly inflexible. If an engineer makes a mistake in the instruction set, or wants to add a new instruction later, they cannot just update software; they have to physically redesign and remanufacture the entire silicon chip.

2. Microprogrammed Control Unit

A microprogrammed control unit is essentially a tiny computer *inside* the computer. Instead of a massive web of logic gates, it uses a small, high-speed internal memory (Control Memory or ROM).

- **How it works:** The main CPU instruction (the "Macro-instruction") is translated into a sequence of smaller, lower-level instructions called "Micro-instructions." These micro-instructions are stored in a ROM chip inside the Control Unit. When a main instruction is decoded, it just triggers a mini-program that runs inside the control unit to flip the necessary switches.
- **Advantages:** It is highly flexible. Designing a complex processor is much easier because engineers are just writing micro-code rather than drawing millions of wires. If a bug is found, the microcode can sometimes be updated without destroying the physical silicon. This is heavily used in CISC architectures (like standard Intel processors).
- **Disadvantages:** It is generally slower than hardwired control because the control unit has to constantly fetch micro-instructions from its internal memory before it can execute the main instruction.

Feature	Hardwired Control	Microprogrammed Control
Physical Design	Massive web of logic gates	Small internal ROM containing microcode
Speed	Extremely Fast	Relatively Slower

Flexibility	Rigid; impossible to change after manufacturing	Flexible; microcode can be altered
Circuit Complexity	Highly complex to design and debug	Simpler structural design
Primary Use Case	RISC Processors (e.g., ARM chips in phones)	CISC Processors (e.g., x86 chips in laptops)

NOTES

Table 8.2: Hardwired vs. Microprogrammed Control.

✓ Check Your Progress 8.2.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. Which CPU register holds the address of the next instruction to be fetched?

.....

2. Why is a Microprogrammed Control Unit considered more flexible than a Hardwired one?

.....



Activity 8.1. THE HUMAN FETCH-DECODE-EXECUTE CYCLE

Title : Simulating the Processor Clock

Objective : To physicalize the abstract concept of Instruction Sequencing and the Fetch-Decode-Execute cycle.

Material Needed : 10 index cards, a pen, and three distinct areas on your desk (Memory, CPU, Output).

Activity Description:

1. **Prepare Memory :** Write five simple math problems on five index cards (e.g., Card 1: ADD 5+2, Card 2: SUB 10-3, Card 3: MULTIPLY 4x2). Stack them face down in the "Memory" area.
2. **Prepare the PC :** Take a blank index card and write Address: 1. Place it in the "CPU" area. This is your Program Counter.
3. **The Cycle - Tick 1 (FETCH) :** Look at your Program Counter (it says 1). Go to "Memory" and fetch Card 1. Bring it to the CPU

NOTES

area. Immediately cross out 1 on the Program Counter and write 2.

4. **The Cycle - Tick 2 (DECODE)** : Look at Card 1. Translate what it is asking you to do in your brain. You recognize the word ADD. You prepare to do addition.
5. **The Cycle - Tick 3 (EXECUTE)** : Perform the math in your head. Write the answer (7) on a blank index card and move it to the "Output" area.
6. **Repeat** : Loop back to Step 3, looking at your new Program Counter value (2).

Reflection:

You just acted as the Control Unit, the ALU, and the Clock. Notice how the entire system depends on strict sequencing. If you tried to execute the math before decoding the instruction, the system would fail. The computer processes billions of these index cards a second, but the logic remains exactly as simple as the exercise you just performed.

8.5. SUMMARY

In this exhaustive unit, we transitioned from static combinational logic to dynamic, sequenced computing. We began with a deep refresher on the Arithmetic Logic Unit (ALU), establishing it as the fundamental calculator of the processor, relying heavily on data inputs, opcodes, and crucially, Status Flags (Zero, Carry, Sign, Overflow) to inform the system about the results of its math.

We then shifted to the brain of the operation: The Control Unit and Program Control. We dissected the Fetch-Decode-Execute cycle, illustrating the relentless, clock-driven journey of an instruction. We highlighted the critical role of the Program Counter (PC) in keeping the processor oriented within the software's memory space, and the Instruction Register (IR) in holding the data for translation. Finally, we explored the physical implementation of the Control Unit, contrasting the blazing speed but rigid inflexibility of Hardwired Control (made of physical logic gates) against the adaptable, software-driven approach of Microprogrammed Control (using internal ROM and micro-instructions). Understanding this sequencing is what transforms a pile of silicon into a machine capable of running complex software.

8.6. GLOSSARY

- **Arithmetic Logic Unit (ALU):** A digital circuit used to perform arithmetic and logic operations. It represents the fundamental building block of the central processing unit of a computer.
- **Control Unit (CU):** A component of a computer's CPU that directs the operation of the processor. It tells the computer's memory, ALU, and input/output devices how to respond to the instructions.

- **Program Counter (PC):** A CPU register that indicates where a computer is in its program sequence; it holds the memory address of the next instruction to be fetched.
- **Instruction Register (IR):** A register inside the CPU that holds the instruction currently being executed or decoded.
- **Fetch-Decode-Execute Cycle:** The basic operation cycle of a computer, consisting of retrieving an instruction from memory, determining what it means, and carrying out the action.
- **Hardwired Control:** A control unit design utilizing physical logic gates and combinational circuits to generate control signals.
- **Microprogrammed Control:** A control unit design that uses a sequence of micro-instructions stored in internal memory to generate control signals.
- **Opcode (Operation Code):** The portion of a machine language instruction that specifies the operation to be performed.
- **Status Flag:** A hardware indicator inside the ALU that signifies a specific condition (like a zero result or an overflow error) has occurred during an operation.
- **Bus:** A communication system that transfers data between components inside a computer.

8.7. ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 8.1

1. When the Zero Flag is set to 1, it indicates that the mathematical or logical result of the most recent ALU operation was exactly zero (e.g., $\$5 - 5 = 0\$$).
2. The ALU requires Operands (the data to calculate), an Opcode (the instruction telling it *what* math to do), and a Clock/Enable signal to synchronize the operation.

Check Your Progress 8.2

1. The **Program Counter (PC)**.
2. A Microprogrammed Control Unit is more flexible because its control signals are generated by reading "micro-instructions" from a ROM chip, which can be modified or updated without having to physically redesign the silicon logic gates on the motherboard.

8.8. SUGGESTED READINGS – OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectonline.com/User/UnitUrls?unitId=3832	

NOTES

8.9. REFERENCE

1. Mano, M. M. (2008). *Computer System Architecture*. Pearson.
 2. Stallings, W. (2019). *Computer Organization and Architecture: Designing for Performance*. Pearson.
-

8.10. EXERCISE

PART – 1**A. Multiple Choice Questions**

1. The register responsible for storing the exact memory address of the next instruction to be fetched is the:
(a) Instruction Register
(b) Accumulator
(c) Program Counter
(d) Status Register
2. If an ALU adds two massive positive numbers together and the result becomes a negative number due to bit constraints, which flag will activate?
(a) Zero Flag
(b) Carry Flag
(c) Overflow Flag
(d) Equal Flag
3. In the Fetch-Decode-Execute cycle, what happens immediately after an instruction is placed into the Instruction Register?
(a) The instruction is executed by the ALU.
(b) The Program Counter is reset to zero.
(c) The instruction is decoded by the Control Unit.
(d) The data is saved back to the hard drive.
4. A Hardwired Control Unit is primarily built using:
(a) High-speed ROM
(b) Pure combinational and sequential logic gates
(c) Micro-instructions
(d) Hard drives
5. Which control unit architecture is most commonly associated with high-speed RISC processors?
(a) Microprogrammed Control
(b) Harvard Control
(c) Sequential Control
(d) Hardwired Control

Answers with explanations :

NOTES

1. **Answer : (c)**

Explanation : The Program Counter acts as the CPU's bookmark, always pointing to the location of the next pending instruction.

2. **Answer : (c)**

Explanation : The Overflow Flag is designed exactly for this: it detects when an arithmetic operation exceeds the maximum size limit of the signed binary format, resulting in erroneous math.

3. **Answer : (c)**

Explanation : Once fetched, the binary data means nothing until it is translated. The Decode phase must translate the dense binary into physical control signals before execution can occur.

4. **Answer : (b)**

Explanation : Hardwired control relies completely on physical hardware—AND, OR, NOT gates—to generate signals, rather than reading software from a ROM.

5. **Answer : (d)**

Explanation : Because RISC (Reduced Instruction Set Computer) processors prioritize maximum execution speed, they rely on the physically faster logic-gate architecture of Hardwired control units.

B. Short answer type questions

1. Define the purpose of the "Opcode" in digital computing.
2. List three common Status Flags found in an Arithmetic Logic Unit.
3. Briefly explain the difference between the Program Counter (PC) and the Instruction Register (IR).
4. State one major disadvantage of a Hardwired Control Unit.

PART – 2

C. Long answer type question (200 words)

1. Describe the architectural flow of data into and out of an Arithmetic Logic Unit (ALU). Detail the specific roles of Operand A, Operand B, the Opcode, the Result bus, and the Status Flags. Why are the Status Flags considered the "decision-making" mechanism for a computer program?
2. Outline the three major stages of the Fetch-Decode-Execute cycle. Describe the specific actions occurring on the physical data buses and address buses during the "Fetch" stage. Why is the system clock vital to this three-step process?
3. Compare and contrast Hardwired Control Units with Microprogrammed Control Units regarding execution speed

NOTES

and design flexibility. If you were engineering a processor for an aerospace application where a hardware bug could be fatal and impossible to physically repair in space, which architecture might you lean toward and why?

PART – 3**D. Long answer type question (300 words)**

1. Provide a highly detailed, step-by-step narrative of a microprocessor executing a single command: ADD Register 1 to Register 2. Begin your narrative at the precise moment the clock ticks to initiate the Fetch cycle. Trace the flow of the memory address from the Program Counter to the RAM, the return of the binary instruction to the Instruction Register, and the translation process during Decode. Finally, describe the physical signals the Control Unit must send to the ALU and the Registers during the Execute phase to successfully perform the addition. Conclude by explaining how the Program Counter prepares the system for the very next instruction.
2. Deeply analyze the philosophy of Control Unit design. The transition from Hardwired Control (using physical logic gates) to Microprogrammed Control (using a ROM to store micro-instructions) represented a massive paradigm shift in computer engineering. Explain how Microprogrammed Control essentially puts a "computer inside a computer." Discuss how this design abstracts the hardware, allowing engineers to write software (microcode) to dictate hardware behavior rather than physically soldering complex logic gate webs. Address the impact this had on the development of complex CISC processors.
3. Evaluate the vital importance of the Arithmetic Logic Unit's Status Register (Flags) in high-level computer programming. Consider a simple IF/ELSE software statement, such as IF (X == Y) THEN DO A. The CPU does not inherently understand equality. Explain how the compiler translates this high-level IF statement into a subtraction operation fed to the ALU. Detail how the Control Unit monitors the **Zero Flag** during this subtraction to determine if \$X\$ and \$Y\$ are indeed equal, and how the state of that single flag wire dictates which branch of software the processor will jump to next.



DR. C.V. RAMAN UNIVERSITY

Chhattisgarh, Bilaspur AN AICTE GROUP UNIVERSITY

Approved by: PCI | AICTE | NCTE | BCI | Member of: AU | Joint Committee: JUGC | BEB | AICTE | Recognized by: UGC | A NAAC Accredited University



DR. C. V. RAMAN UNIVERSITY (C.G.)

SLM FEEDBACK FORM

PROGRAMME : _____

COURSE : _____

BLOCK : _____

UNIT : _____

Dear Student,

While studying the units of this block, you may have found certain portions of the text difficult to comprehend. We wish to know your difficulties and suggestions, in order to improve the course. Therefore, we request you to fill up and send us the following questionnaire, which pertains to this block. If you find the space provided insufficient, kindly use a separate sheet.

Please mail to aisectcontent@aisect.org / for online submission please scan QR code:

Questionnaire

1. Name : _____

2. Enrolment No. :

--	--	--	--	--	--	--	--	--	--	--

3. How many hours did you need for studying the units? :

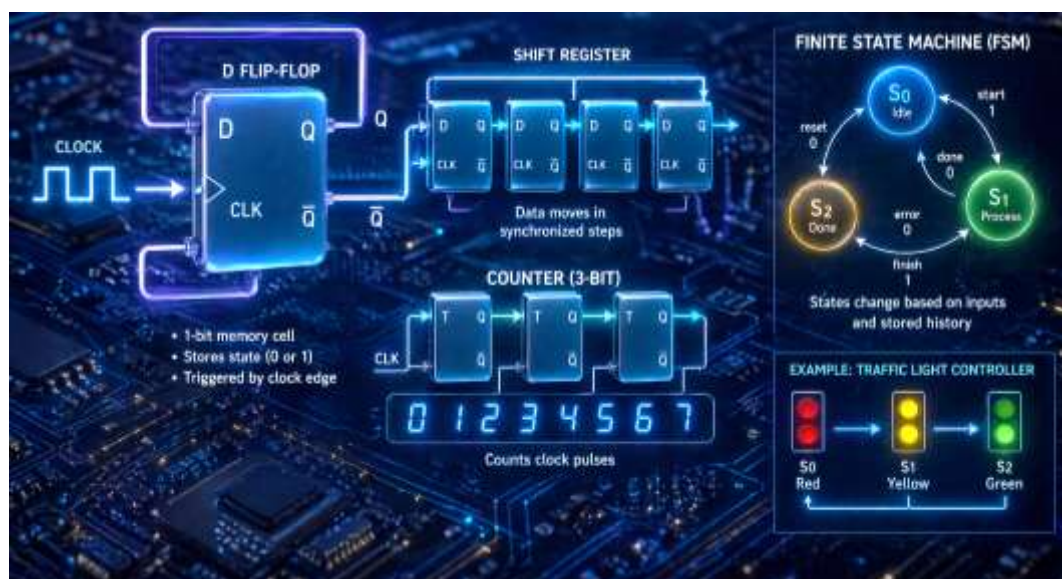
--

4. Please give your reactions to the following items based on your reading of the block:

Items	Excellent	Very Good	Good	Poor	Give specific Examples (If poor)
Presentation Quality					
Language and Style					
Illustrations					
Used (diagrams, tables, etc.)					
Conceptual Clarity					
Check Your Progress Questions					
Suggested Readings - OER					

5. Any other comments:

Block - III	SEQUENTIAL LOGIC AND STATE MACHINES	87 - 150
Unit 9	Sequential Logic and Basic Flip Flops	89
Unit 10	Advanced Flip-Flop Configurations and Triggering	105
Unit 11	Shift Registers	117
Unit 12	Counters and State Diagrams	131
Unit 13	State Machines	141



In this vital section, we introduce the missing ingredient that separates calculators from robust computers: memory. Unlike combinational circuits, which suffering from total amnesia when their inputs are removed, sequential circuits utilize feedback loops to capture and retain logic states indefinitely. We must introduce a new fundamental building block: the Flip-Flop. This synchronized multivibrator acts as a **Digital Delay Line** or 1-bit memory cell, trapping a logical 1 or 0. Its operation is governed by a master **Clock Signal**, which acts as the system's metronome. We will analyze how edge-triggering and precise timing ensure flip-flops sample data only on specified rising or falling clock edges. After mastering individual memory cells, we will group them to construct Shift Registers (conveyor belts for moving data words) and Counters (trackers of time and events). The block concludes with Finite State Machines (FSMs), arguably the most crucial design concept in digital intelligence. You will use state diagrams to visualize abstract algorithmic logic before translating those states into physical hardware (such as a vending machine or traffic light controller), ensuring your machine can intelligently sequence events based on its inputs and stored historical context.

UNIT – 9

SEQUENTIAL LOGIC AND BASIC FLIP FLOPS

NOTES

UNIT STRUCTURE

- 9.1. Introduction
- 9.2. Objective
- 9.3. Introduction to Sequential Logic Circuits
- 9.4. Flip Flops Basics
 - 9.4.1. D Flip Flop
 - 9.4.2. T Flip Flop
 - 9.4.3. S-R and J-K Flip Flops
- 9.5. Summary
- 9.6. Glossary
- 9.7. Answers to check your progress
- 9.8. Suggested readings - OER
- 9.9. Reference
- 9.10. Exercise

9.1. INTRODUCTION

A monumental turning point in your journey through the landscape of digital electronics. As we inaugurate Block 3, we are fundamentally shifting the way we think about computing hardware.

If we look back at the previous units comprising Block 2, our entire focus was dedicated to exploring Combinational Logic Circuits. We engineered mathematical marvels like Adders and Subtractors, and we designed intricate data-routing networks using Multiplexers and Decoders. However, despite their complexity, all combinational circuits share one severe, inherent limitation: they suffer from total amnesia. A combinational circuit has absolutely no concept of history, no awareness of time, and no ability to remember what happened even a fraction of a millisecond ago. If you present a set of inputs to an Adder, it will calculate the sum perfectly. But the instant you remove those inputs, the answer vanishes into thin air. The circuit reverts to its base state, oblivious to the fact that it ever performed a calculation.

For a computer to be genuinely useful—for it to run software, play a video game, or calculate the trajectory of a spacecraft—it must have the ability to remember. It needs an internal memory to store the results of

NOTES

step one so that it can apply them to step two. It needs to know not just what is happening *now*, but what happened *before*.

This critical requirement introduces us to the profound domain of **Sequential Logic Circuits**. In this unit, we will cross the threshold from memoryless machines into stateful systems. We will begin by deeply investigating the theoretical architecture of sequential logic, uncovering how engineers ingeniously take the output of a circuit and loop it right back into the input, creating a continuous "feedback loop." This loop is the physical manifestation of memory. We will examine the absolute necessity of the system "Clock," a ticking electrical heartbeat that coordinates millions of these feedback loops so that data does not crash into itself.

Once we have established the theoretical framework of time and state, we will zoom in to the microscopic level to build the actual hardware. We will dissect the foundational building blocks of digital memory: the **Flip-Flops**. You will learn how basic NAND and NOR logic gates are wired against one another to create stable circuits capable of trapping and holding a single binary bit (a '1' or a '0') indefinitely. We will rigorously explore the evolution of these memory cells, starting from the basic S-R (Set-Reset) Flip-Flop, moving to the data-driven D Flip-Flop, examining the oscillating T (Toggle) Flip-Flop, and culminating in the highly versatile J-K Flip-Flop.

By the time you complete this unit, you will no longer just be looking at circuits that react; you will be looking at circuits that remember. You will have built the fundamental foundations of RAM, computer registers, and the sequential state machines that run the modern world. Let us begin giving our hardware a memory.

9.2. OBJECTIVES

After successfully completing this unit, you will be able to:

- Critically distinguish between the functional paradigms of combinational logic circuits and sequential logic circuits.
- Define and analyze the concept of "State" within digital electronics, understanding how past inputs influence current and future outputs.
- Deconstruct the fundamental block diagram of a sequential logic system, paying special attention to the implementation and necessity of the feedback loop.
- Differentiate between synchronous and asynchronous sequential circuits, evaluating the advantages and timing challenges inherent in both architectures.
- Explain the critical role of the system clock signal as the central synchronizing metronome of a digital processor.
- Analyze the theoretical foundation, physical construction, and truth tables of fundamental 1-bit memory storage elements known as Flip-Flops.

- Contrast the operational mechanics, logic gates, and characteristic equations of the D (Data), T (Toggle), S-R (Set-Reset), and J-K Flip-Flops.

9.3. INTRODUCTION TO SEQUENTIAL LOGIC CIRCUITS

To understand how a computer remembers, we must first reconstruct our understanding of how a digital circuit is wired. We are moving from a straight-line data path to a circular, evolving data path.

1. The Fundamental Definition In the strictest engineering terms, a Sequential Logic Circuit is a complex digital system where the output state at any given moment is determined not only by the present combination of inputs but also by the past sequence of inputs. Because the circuit's current behavior is dictated by its history, the circuit must possess physical hardware capable of storing that history. This stored history is formally referred to as the "**State**" of the circuit.

2. The Architecture of Memory: The Feedback Loop How does one build "memory" using nothing but standard logic gates? The secret lies in a concept called **Feedback**. If you look at the block diagram of a sequential circuit, you will notice that it actually contains a standard combinational logic block (like the Adders or Multiplexers we already studied) at its core. However, it adds two critical new elements: Memory Elements and a Feedback Path.

The outputs generated by the combinational logic block are not just sent out into the world; a portion of these outputs is routed directly into a dedicated bank of Memory Elements. These Memory Elements hold onto these binary values. Then, these stored values are routed backward—via the feedback path—to serve as secondary inputs to the combinational logic block.

Therefore, the combinational logic block is simultaneously receiving:

1. **External Inputs:** New data coming from outside the system (e.g., a user pressing a keyboard key).
2. **Present State Inputs:** The stored, historical data looping back from the memory elements.

The combinational circuit computes a new result based on *both* of these inputs. The new result becomes the external output and also overwrites the old data in the memory elements, establishing the "**Next State**." This continuous evolution from Present State to Next State is the essence of digital computing.

3. Synchronous vs. Asynchronous Sequential Circuits Because sequential circuits rely on feedback loops, they are highly sensitive to timing. If the output loops back to the input too quickly, the circuit might process the same data twice. If different logic gates process data at slightly different speeds (known as propagation delay), the feedback loop can become chaotic, leading to a system

NOTES

crash. To manage this, engineers categorize sequential circuits into two distinct types based on how they handle timing:

- **Asynchronous Sequential Circuits:** In an asynchronous system, there is no master timekeeper. The state of the circuit changes instantaneously the moment any of the external input variables change.
 - *Mechanism:* They are primarily driven by the raw electrical propagation of the signals through the logic gates. If Input A changes, the logic gates react as fast as physics allows, and the new state cascades through the feedback loop.
 - *Advantages:* They can be incredibly fast because they do not have to wait for a specific scheduled time to update. They operate at the absolute physical speed limit of the silicon.
 - *Disadvantages:* They are notoriously difficult to design and debug. Because different logic gates have microscopic differences in manufacturing, signals might race each other through the circuit. If Signal A arrives at a gate a billionth of a second before Signal B (when they were supposed to arrive together), the circuit might briefly output garbage data or enter an endless, unstable loop. This is known as a "Race Condition." Due to this instability, complex modern processors rarely use pure asynchronous designs for their main calculating cores.
- **Synchronous Sequential Circuits:** In a synchronous system, absolute order is imposed by a master timekeeper called the **Clock**. A synchronous circuit will only update its internal state at specific, discrete, uniformly spaced intervals of time, regardless of when the external inputs change.
- **Mechanism :** The system uses memory elements that are specifically wired to ignore all incoming data until they receive a permissive "pulse" from the central Clock generator. Even if the external inputs are changing wildly, the memory elements lock their doors. Only when the clock "ticks" do the doors briefly open, allowing the combinational logic to read the inputs, calculate the new state, and lock the doors again.
- **Advantages:** This architecture completely eliminates the chaos of race conditions. By ensuring that the clock ticks slightly slower than the slowest logic gate in the system, engineers guarantee that all electrical signals have perfectly settled into steady 1s and 0s before the system is allowed to take its next step. This makes synchronous circuits highly predictable, stable, and much easier to design at a massive scale.
- **Disadvantages:** The maximum speed of the entire computer is dictated by the master clock (this is why you see CPU speeds advertised as 3.5 GHz—meaning the clock ticks 3.5 billion times per second). Furthermore, routing a clock wire to every

single memory element across a massive motherboard requires significant power and highly complex physical wiring trees.

- 4. The Anatomy of the Clock Signal** Because synchronous sequential circuits form the backbone of 99% of modern digital electronics, it is vital to intimately understand the Clock signal. A digital clock does not tell the time of day; it is simply a continuous, perfectly square electrical wave that alternates between a logic HIGH (1) and a logic LOW (0).

When engineers design synchronous memory elements (like the flip-flops we will study in the next section), they must decide exactly *when* during the clock wave the circuit is allowed to update. There are two primary triggering methods:

1. **Level Triggering:** The circuit is allowed to update its state continuously for the entire duration that the clock signal remains HIGH (or LOW). This is somewhat akin to opening a gate and letting traffic flow as long as the light is green. While simpler, it can still lead to timing issues if the inputs change while the clock is still HIGH.
2. **Edge Triggering:** This is the most robust and widely used method in modern computing. The circuit is strictly designed to update its state only at the exact microscopic instant the clock signal transitions from LOW to HIGH (Positive Edge-Triggered) or from HIGH to LOW (Negative Edge-Triggered). It does not matter how long the clock stays high or low; the action only occurs on the sharp, instantaneous edge of the wave. This provides laser-precise synchronization across billions of transistors.

By mastering the concept of the feedback loop and the synchronizing power of the system clock, you have theoretically solved the problem of digital memory. The next step is to take our soldering irons and build the actual hardware circuits capable of obeying these synchronous rules.

✓ Check Your Progress 9.1.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. What is the fundamental physical difference between a combinational logic circuit and a sequential logic circuit?

2. Why are asynchronous circuits rarely used for the main calculating cores of complex microprocessors?

NOTES

.....

.....

.....

9.4. FLIP FLOPS BASICS

Having established the theoretical need for feedback loops and the synchronizing pulse of a system clock, we must now build the physical hardware that executes these concepts. The most fundamental building block of all digital memory is a circuit known as a Flip-Flop.

In the formal language of electronic engineering, a flip-flop is defined as a bi-stable multivibrator. "Bi-stable" means the circuit has exactly two stable states: it can hold a logical 1 (often called the SET state) or a logical 0 (often called the RESET or CLEAR state). Once it is pushed into one of these states by an external input, the internal feedback loop acts like a lock. The circuit will hold that state indefinitely, perfectly "remembering" the bit of data, even after the original input signal is completely removed. It will only change its state when explicitly commanded to do so by a new set of inputs combined with a valid clock pulse.

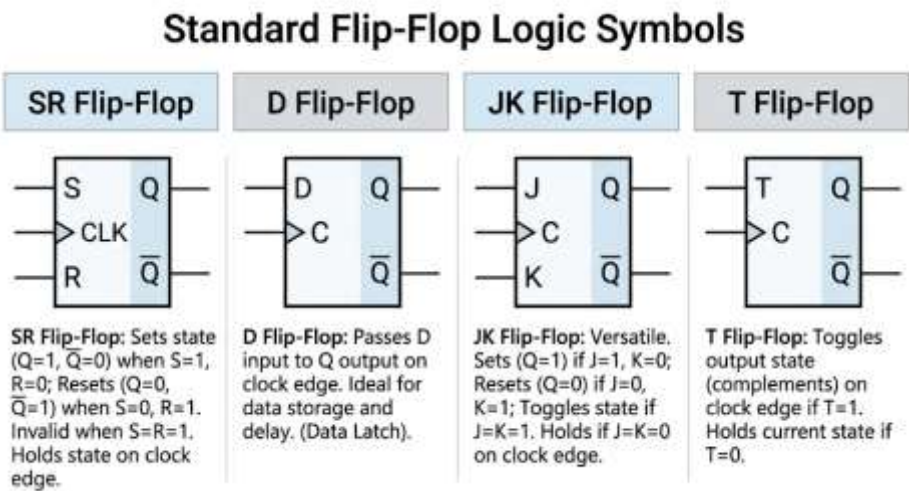


Fig. 9.1 : Standard Flip-Flop Logic Symbols.

Latches vs. Flip-Flops

Before analyzing the specific types, we must clarify a critical industry distinction between "Latches" and "Flip-Flops."

- **The Latch:** A latch is the raw, unclocked (or level-triggered) memory element. It is made simply by taking two logic gates (usually NAND or NOR gates) and cross-coupling them—wiring the output of the first gate into the input of the second, and the output of the second gate back into the input of the first. A latch is entirely transparent; if the inputs change, the stored state changes immediately.

- **The Flip-Flop:** A flip-flop is essentially a latch that has been upgraded with a synchronous, edge-triggered clocking mechanism. It acts as a gated vault. The inputs can change constantly on the outside, but the flip-flop will only read those inputs and update its internal latch at the exact microscopic edge (rising or falling) of the system clock.

Every flip-flop fundamentally outputs two signals:

1. The normal output, conventionally labeled as Q .
2. The inverted output, conventionally labeled as $\overline{Q}$ (pronounced "Q-bar"). By the laws of physics governing the cross-coupled gates, if Q is 1, $\overline{Q}$ must physically be forced to 0, and vice versa.

Let us now dissect the four major classifications of flip-flops, analyzing their unique logic structures and applications.

9.4.1. D FLIP-FLOP

The **D Flip-Flop** (where "D" stands for **Data** or sometimes **Delay**) is arguably the most important and widely used memory element in modern digital computing. It forms the backbone of CPU registers and system RAM.

1. The Logic and Operating Principle

The D flip-flop was engineered to solve a specific problem found in older flip-flop designs: the risk of invalid or contradictory inputs. The D flip-flop simplifies the input interface entirely. It has exactly one data input line, labeled D, and one Clock input (CLK).

The operating principle is elegant and absolute: **The output Q will simply take on the exact value of the input D at the next active clock edge.** * If $D = 1$ when the clock ticks, the flip-flop SETS, and Q becomes 1.

If $D = 0$ when the clock ticks, the flip-flop RESETS, and Q becomes 0.

Because there is only one data input, it is physically impossible to give the circuit conflicting commands. Internally, the D flip-flop takes the single D line and immediately splits it. One path goes directly into the internal latch logic, while the other path goes through a NOT gate (inverter) before entering the other side of the latch. This guarantees that the internal set and reset mechanisms are always perfectly complementary.

2. The Truth Table and Characteristic Equation

In sequential logic, we map the circuit's behavior using a characteristic table. We use the notation Q_n to represent the "Present State" (what is currently held in memory) and Q_{n+1} to represent the "Next State" (what the memory will become after the clock ticks).

Present State (Q_n)	Data Input (D)	Next State (Q_{n+1})	Action
-------------------------	----------------	--------------------------	--------

NOTES

0	0	0	Reset / Store 0
0	1	1	Set / Store 1
1	0	0	Reset / Store 0
1	1	1	Set / Store 1

Table 9.1: Characteristic Table of a D Flip-Flop.

Notice that the Next State ($Q_{\{n+1\}}$) completely ignores whatever the Present State (Q_n) was. It simply mirrors the D input. Therefore, the characteristic mathematical equation for a D flip-flop is wonderfully simple:

$$Q_{\{n+1\}} = D$$

3. Application: The Delay Element

Why is it sometimes called a "Delay" flip-flop? Because whatever data is sitting on the D line cannot pass through to the Q output until the clock ticks. If a 1 arrives at the input, it is delayed there until the rising edge of the clock allows it to pass. By chaining millions of these together, engineers create CPU Registers—highly synchronized staging areas where 64-bit words of data wait precisely for the processor's command to move.

9.4.2. T FLIP-FLOP

The **T Flip-Flop** (where "**T**" stands for **Toggle**) is a specialized memory element designed not to simply store specific data, but to alternate its state upon command.

1. The Logic and Operating Principle

The T flip-flop has a single input line labeled **T** and a Clock input. Unlike the **D** flip-flop, which forces the output to match the input, the **T** flip-flop's behavior is highly dependent on its *Present State*.

If $T = 0$ when the clock ticks, the flip-flop does nothing. It holds its current state.

If $T = 1$ when the clock ticks, the flip-flop **Toggles**. If it was holding a 0, it flips to a 1. If it was holding a 1, it flips to a 0.

Internally, a T flip-flop is usually constructed by taking a more complex flip-flop (like a J-K) and permanently tying its set and reset input pins together to form the single **T** line. The feedback loop ensures that the current output is fed back into the input logic to guarantee the circuit always knows what its opposite state should be.

2. The Truth Table and Characteristic Equation

Present State (Q_n)	Toggle Input (T)	Next State (Q_{n+1})	Action
0	0	0	Hold State
0	1	1	Toggle
1	0	1	Hold State
1	1	0	Toggle

NOTES

If you look closely at this table, the Next State output behaves exactly like an Exclusive-OR (XOR) logic gate evaluating the T input and the Present State. Thus, the characteristic equation is:

$$Q_{n+1} = T \oplus Q_n$$

3. Application: Digital Counters and Dividers

The T flip-flop is the undisputed king of counting circuits. Imagine you keep the T input permanently wired to a logical 1. Every single time the clock ticks, the flip-flop will toggle $0 \rightarrow 1 \rightarrow 0 \rightarrow 1$.

Because it takes *two* clock ticks to complete a full cycle of output (returning to 0), the output wave Q is oscillating at exactly half the frequency of the main clock. The T flip-flop is functionally a "Divide-by-2" frequency divider. By chaining several T flip-flops together—where the output of the first acts as the clock for the second—engineers build the Binary Counters found in digital clocks, timers, and CPU program counters.

9.4.3. S-R AND J-K FLIP-FLOPS

We now arrive at the historical grandfather of all memory circuits (the S-R) and its perfected, universal evolution (the J-K).

1. The S-R (Set-Reset) Flip-Flop

The S-R flip-flop is the foundational architecture built from basic cross-coupled NOR or NAND gates. It has two distinct input lines: S (Set) and R (Reset), alongside the Clock.

The operating rules are straightforward:

- If $S = 0$, $R = 0$: **Hold State**. The circuit remembers its previous value.
- If $S = 1$, $R = 0$: **Set**. The output Q is forced to 1.
- If $S = 0$, $R = 1$: **Reset**. The output Q is forced to 0.

However, the S-R flip-flop possesses a fatal engineering flaw. What happens if you tell the circuit to both Set and Reset at the exact same time?

- If $S = 1$, $R = 1$: **The Invalid State**. This condition forces both the Q and $\bar{Q}$ outputs to the exact same logic level (usually 0, depending on the gate type). This violates the fundamental law of flip-flops that Q and $\bar{Q}$ must be opposites. Furthermore, if the S and R inputs are returned to 0 simultaneously, the circuit enters a

SEQUENTIAL LOGIC AND STATE MACHINES

NOTES

chaotic "race condition," wildly fluctuating before settling into an unpredictable state. Because of this fatal flaw, pure S-R flip-flops are rarely used in modern synchronized logic design.

Q_n	S	R	Q_{n+1}	Action
0	0	0	0	Hold
0	0	1	0	Reset
0	1	0	1	Set
0	1	1	?	Invalid / Forbidden
1	0	0	1	Hold
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	?	Invalid / Forbidden

Table 9.3 : Characteristic Table of an S-R Flip-Flop

(Characteristic Equation: $Q_{n+1} = S + \{\overline{R}\}Q_n$, with the strict condition that $S \cdot R = 0$)

2. The J-K Flip-Flop: The Universal Savior

To eliminate the invalid state of the S-R flip-flop, engineers devised the J-K flip-flop. (The letters J and K have no universally accepted meaning, though they are often attributed to Jack Kilby, the inventor of the integrated circuit).

The J-K flip-flop operates exactly like an S-R flip-flop for three of its four input conditions. The **J** acts as the Set input, and the **K** acts as the Reset input. However, the brilliance of the J-K flip-flop lies in how it handles the 1,1 input condition.

Engineers took the outputs (**Q** and $\overline{Q}$) and fed them all the way back into the input logic gates alongside the **J** and **K** signals. Because of this massive feedback loop, the circuit always cross-references its own current state before acting.

- If **J=1 and K=1: Toggle**. Instead of crashing into an invalid state, the internal feedback logic turns the 1,1 command into a Toggle command. If the state was 1, the feedback forces the internal Reset to fire. If the state was 0, the feedback forces the internal Set to fire.

Q _n	J	K	Q _{n+1}	Action
0	0	0	0	Hold
0	0	1	0	Reset
0	1	0	1	Set
0	1	1	1	Toggle (0 to 1)
1	0	0	1	Hold
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	0	Toggle (1 to 0)

NOTES

Table 9.4 : Characteristic Table of a J-K Flip-Flop.

The characteristic equation for the J-K flip-flop beautifully incorporates the feedback mechanism:

$$Q_{\{n+1\}} = J \overline{Q_n} + \overline{K} Q_n$$

Because the J-K flip-flop safely utilizes all possible input combinations, it is considered a **Universal Flip-Flop**. By simply tying the inputs together in different ways, a J-K flip-flop can instantly be transformed into a D flip-flop or a T flip-flop, making it a highly valuable component in modular digital design.

✓ Check Your Progress 9.2.

- Notes :** (a) Space is given below for your answer.
 (b) Check your answer with the one given at the end of this unit.

- How does the D Flip-Flop prevent the "invalid state" problem found in the S-R Flip-Flop?

- What happens to a J-K Flip-Flop when both the \$J\$ and \$K\$ inputs are set to logic 1 during a clock pulse?

9.5. SUMMARY

In this foundational unit, we crossed the critical threshold from memoryless combinational logic into the stateful, time-aware domain of sequential logic. This transition represents the precise moment where a digital circuit evolves from a simple calculator into a functional computer capable of retaining history. We established that the defining characteristic of a sequential circuit is its reliance on a **Feedback Loop**,

NOTES

a physical wiring architecture where the outputs of a combinational logic block are routed back into its own inputs via specialized memory elements. This loop creates the concept of "State," allowing the system's future actions to be dictated not just by present external inputs, but by the accumulated history of past inputs.

We thoroughly analyzed the vital necessity of timing and synchronization in these feedback loops. We contrasted **Asynchronous Circuits**, which react instantly to input changes and are highly susceptible to chaotic "race conditions," with **Synchronous Circuits**, which bring absolute order to the system through the use of a master **Clock**. We explored how the clock acts as a relentless digital metronome, ensuring that memory updates only occur at specific, discrete intervals—specifically on the microscopic rising or falling edges of the clock signal.

Transitioning from theory to hardware, we dissected the fundamental bi-stable memory cells known as **Flip-Flops**. We differentiated transparent Latches from edge-triggered Flip-Flops and systematically evaluated the four primary architectures:

1. **The S-R (Set-Reset) Flip-Flop:** The primitive ancestor built from cross-coupled gates, plagued by a fatal "invalid state" when commanded to simultaneously set and reset.
2. **The D (Data) Flip-Flop:** The backbone of modern computer memory and CPU registers, designed with a single input to faithfully capture and delay a data bit until the next clock tick.
3. **The T (Toggle) Flip-Flop:** The oscillating memory element used predominantly in digital counters and frequency dividers, changing its state only when the toggle input is active.
4. **The J-K Flip-Flop:** The universal triumph of sequential design, utilizing massive internal feedback to solve the S-R's invalid state by gracefully converting simultaneous high inputs into a controlled toggle action.

By mastering these bi-stable multivibrators, you have acquired the theoretical and practical knowledge required to build registers, counters, and the complex finite state machines that govern modern microprocessors.

9.6. GLOSSARY

- **Asynchronous Sequential Circuit:** A logic circuit whose internal state changes immediately in response to changes in external inputs, without waiting for a synchronizing clock pulse. Highly fast but prone to timing errors.
- **Bi-stable Multivibrator:** The formal electronic classification for a flip-flop; a circuit possessing exactly two stable electrical states (HIGH and LOW) that it can hold indefinitely until triggered.

- **Clock Signal:** A continuous, highly precise square wave alternating between logic 0 and logic 1, used to synchronize the actions of all memory elements across a digital system.
- **D Flip-Flop (Data/Delay):** A type of edge-triggered memory element with a single data input. Its primary function is to capture the value of the input at the exact moment of the clock edge and hold it at the output.
- **Edge-Triggering:** A design mechanism where a memory circuit only reads its inputs and updates its state during the instantaneous microscopic transition of a clock signal (either moving from low-to-high or high-to-low).
- **Feedback Loop:** A physical circuit pathway that routes the output of a system back into its own input, creating the physical basis for digital memory and statefulness.
- **J-K Flip-Flop:** A universal, highly stable memory element that utilizes internal feedback to resolve the invalid state found in S-R flip-flops, converting conflicting high inputs into a toggle command.
- **Latch:** A basic, unclocked (or level-triggered) memory element built from cross-coupled logic gates. It is transparent to input changes as long as it is enabled.
- **Race Condition:** A catastrophic electronic timing error occurring when different logic signals taking different physical paths through a circuit arrive at a destination at slightly different times, causing unpredictable system behavior.
- **State:** The specific, stable binary condition (usually 1 or 0) currently held within the memory elements of a sequential circuit at any given moment in time.
- **Synchronous Sequential Circuit:** A logic circuit that relies on a master clock signal to ensure that all internal state changes happen simultaneously at discrete, predictable intervals.

9.7. ANSWERS TO CHECK YOUR PROGRESS

Check your Progress 9.1

1. **Answer :** A sequential logic circuit physically incorporates memory elements and a feedback loop to remember past states, whereas a combinational circuit has no memory and its output relies solely on the immediate, present inputs.
2. **Answer:** Asynchronous circuits are highly susceptible to "race conditions." Microscopic differences in physical gate manufacturing can cause signals to arrive out of order, leading the un-clocked feedback loops to enter unstable or chaotic states. Synchronous clocks solve this by forcing the system to wait until all signals have settled.

NOTES

Check your Progress 9.2

1. **Answer:** The D flip-flop prevents invalid states by utilizing only one single input pin (**D**). Internally, this input is split, with one path going through an inverter (NOT gate). This hardware design absolutely guarantees that the internal Set and Reset mechanisms always receive opposite logic levels, making a simultaneous Set/Reset command physically impossible.
2. **Answer:** Instead of crashing into an invalid state, the J-K flip-flop will enter the "Toggle" mode. It will read its current output state via internal feedback and invert it (changing a 1 to a 0, or a 0 to a 1).

9.8. SUGGESTED READINGS – OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectonline.com/User/UnitUrls?unitId=3833	

9.9. REFERENCE

- Mano, M. M., & Ciletti, M. D. (2013). *Digital Design: With an Introduction to the Verilog HDL*. Pearson.
- Floyd, T. L. (2015). *Digital Fundamentals*. Pearson Education.
- Tocci, R. J., Widmer, N. S., & Moss, G. L. (2017). *Digital Systems: Principles and Applications*. Pearson.

9.10. EXERCISE

PART – 1

A. Multiple Choice Questions

1. Which of the following is the defining feature of a sequential logic circuit?
 - (a) The presence of XOR logic gates
 - (b) The use of a feedback loop
 - (c) The ability to perform high-speed division
 - (d) The absence of a clock signal
2. A circuit that simply passes its single data input to its output at the exact moment of a clock edge is known as a:
 - (a) J-K Flip-Flop
 - (b) T Flip-Flop
 - (c) D Flip-Flop
 - (d) S-R Latch
3. In an S-R Flip-Flop, the input combination of S=1 and R=1 is strictly avoided because:

- (a) It causes the circuit to overheat and physically melt.
- (b) It forces the output into an unpredictable or invalid state.
- (c) It divides the clock frequency by two.
- (d) It clears the entire system memory to zero.
- 4. If a T Flip-Flop currently holds a state of 1 ($Q_n = 1$), and the T input is set to 1 when the clock triggers, what will the next state (Q_{n+1}) be?
 - (a) 0
 - (b) 1
 - (c) High Impedance
 - (d) The circuit will crash
- 5. Which flip-flop is universally considered the most versatile due to its ability to act as a Set/Reset device, a Data holder, and a Toggle device?
 - (a) S-R Flip-Flop
 - (b) D Flip-Flop
 - (c) Latch
 - (d) J-K Flip-Flop

Answers with explanations:

1. **Answer : (b)**
Explanation : The feedback loop is what gives the circuit "memory" by routing previous output states back into the input logic, which defines sequential circuits.
2. **Answer : (c)**
Explanation : The D (Data) flip-flop is specifically designed to capture the exact value on its D input pin upon a clock edge and hold it.
3. **Answer : (b)**
Explanation : Activating both Set and Reset simultaneously violates the logical rule that Q and $\bar{Q}$ must be opposites, throwing the cross-coupled gates into a forbidden logic conflict.
4. **Answer : (a)**
Explanation : The defining behavior of a T (Toggle) flip-flop when $T=1$ is to invert its current state. Therefore, a 1 toggles to a 0.
5. **Answer : (d)**
Explanation : The J-K flip-flop resolves all the flaws of earlier designs and can easily be wired to mimic the behavior of D, T, and S-R flip-flops.

B. Short answer type questions

1. Define the term "Edge-Triggered" in the context of a digital clock signal.
2. What is the fundamental difference between a Latch and a Flip-Flop?

NOTES

3. Write the characteristic Boolean equation for a D Flip-Flop.
4. Briefly explain why the T Flip-Flop is useful for building digital counters.

PART – 2**C. Long answer type question (200 words)**

1. Contrast Synchronous and Asynchronous sequential circuits. Detail the physical mechanism of how an asynchronous circuit reacts to an input change versus how a synchronous circuit reacts. Discuss the specific engineering threat posed by "race conditions" and explain exactly how the implementation of a master Clock signal neutralizes this threat across a massive motherboard.
2. Deconstruct the logic of the D Flip-Flop. Begin by explaining why the S-R Flip-Flop's "invalid state" was highly problematic for computer memory design. Detail how the internal architecture of the D Flip-Flop—specifically the use of a single input pin and an internal inverter—physically prevents conflicting logic signals from ever reaching the internal cross-coupled latch.
3. Analyze the characteristic table of the T Flip-Flop. Explain the logic of the Toggle function and how it relates mathematically to an Exclusive-OR (XOR) operation. Provide a practical example of how a T Flip-Flop could be utilized to slow down a high-speed clock signal to control a slower peripheral device (like a blinking LED).

PART – 3**D. Long answer type question (answer in 300+ words)**

1. Provide a structural analysis of the J-K Flip-Flop, detailing how its internal feedback loop prevents the invalid state of the S-R Flip-Flop and allows it to safely toggle when inputs are high.
2. Using the example of a digital combination lock, explain why sequential logic (utilizing memory and feedback loops) is strictly required over combinational logic to create a functional state machine.

UNIT – 10

ADVANCED FLIP-FLOP CONFIGURATIONS AND TRIGGERING

NOTES

UNIT STRUCTURE

- 10.1. Introduction
- 10.2. Objective
- 10.3. Advanced Flip-Flop Configurations
 - 10.3.1. Master-Slave Flip Flop
 - 10.3.2. Race Conditions
- 10.4. Edge-Triggered and Level-Triggered Circuits
 - 10.4.1. Level-Triggering (Latches)
 - 10.4.2. Edge-Triggering (Flip-Flops)
 - 10.4.3. Types of Edge Triggering
- 10.5. Summary
- 10.6. Glossary
- 10.7. Answers to check your progress
- 10.8. Suggested readings - OER
- 10.9. Reference
- 10.10. Exercise

10.1. INTRODUCTION

In Unit 9, we explored the individual building blocks of memory: the flip-flops. We learned how a single J-K or D flip-flop can "remember" exactly one bit of information. However, in the world of modern computing, a single bit is rarely enough to perform meaningful work. Computers process data in "words"—groups of 8, 16, 32, or 64 bits that represent numbers, characters, or instructions.

To handle these groups of bits, we must move from individual flip-flops to integrated structures known as **Registers** and **Counters**. Registers act as the high-speed temporary "scratchpads" inside the CPU, moving

NOTES

data around like a conveyor belt. Counters, on the other hand, are the precision "engineers" of the digital world; they keep track of time, count events, and ensure the processor executes instructions in the correct mathematical sequence.

In this unit, we will begin by learning how to chain flip-flops together to form Registers. We will look at how data can be shifted into these registers one bit at a time (Serial) or all at once (Parallel). Following this, we will dive into the world of Counters, starting with the simple "Ripple Counter" and moving into the more advanced "Synchronous Counter" architectures that eliminate timing delays. This unit bridges the gap between basic memory and the complex data-processing systems used in every smartphone and laptop today.

10.2. OBJECTIVES

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

- Define the fundamental role of registers in digital data storage.
- Explain the operation of different types of shift registers.
- Describe the working principle of asynchronous (ripple) counters.
- Analyze the design and timing of synchronous counters.
- Identify the applications of counters in frequency division and timing.
- Differentiate between various counter modulus and their sequences.

10.3. REGISTERS: STORAGE AND DATA MOVEMENT

A register is a group of flip-flops used to store a binary word. Since each flip-flop can store one bit, an n -bit register consists of n flip-flops. Beyond mere storage, registers are essential for "shifting" data, which is a fundamental operation used in multiplication, division, and serial communication.

10.3.1. Types of Shift Registers

Shift registers are classified based on how the data enters the register and how it leaves it. There are four primary configurations:

1. Serial-In/Serial-Out (SISO)

In a SISO register, data is fed in one bit at a time through a single wire. With every clock pulse, the bits shift one position to the right. After n clock pulses, the first bit finally reaches the output.

- **Use Case:** This is primarily used as a temporary delay line or for simple data storage where wiring space is limited.

2. Serial-In/Parallel-Out (SIPO)

Data enters one bit at a time, but once the register is full, all the stored bits are available simultaneously on multiple output pins.

- **Use Case:** This is the heart of communication devices. It takes serial data coming from a single fiber optic cable or USB wire and "unpacks" it into a parallel format that a computer's 64-bit CPU can understand.

3. Parallel-In/Serial-Out (PISO)

This is the reverse of SIPO. All bits are loaded into the flip-flops simultaneously at once. Then, on subsequent clock pulses, the data is shifted out bit-by-bit through a single output wire.

- **Use Case:** This is used to "pack" data to be sent over long distances. Your computer takes a whole byte of data and uses a PISO register to send it out through a serial port.

4. Parallel-In/Parallel-Out (PIPO)

Data is loaded all at once, and it is available at the output all at once. There is no shifting involved; it acts purely as a synchronized storage buffer.

- **Use Case :** These are used as internal registers (like the Accumulator) within a CPU to hold numbers during mathematical calculations.

10.3.2. Bidirectional Shift Registers

While basic registers shift in one direction, a bidirectional shift register uses additional logic gates (multiplexers) to allow data to shift either left or right based on a "Control" input. This is vital for mathematical operations, as shifting a binary number left by one position is the electrical equivalent of multiplying it by 2, while shifting it right is equivalent to dividing it by 2.

☒ Check Your Progress 10.1.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. How many flip-flops are required to construct a 16-bit register?

.....

2. Which type of register is used to convert serial data into parallel data?

.....

NOTES

3. What mathematical operation is performed by shifting a binary number one position to the left?

10.4. ASYNCHRONOUS (RIPPLE) COUNTERS

While registers are designed to move and store specific data patterns, **Counters** are specialized sequential circuits designed to cycle through a predetermined sequence of binary states. An asynchronous counter, commonly known as a "Ripple Counter," is the simplest form of this circuit.

10.4.1. The Operating Principle of Ripple Counters

The term "asynchronous" in this context means that the flip-flops within the counter are not all triggered by the same master clock signal. Instead, only the first flip-flop (the Least Significant Bit or LSB) receives the external system clock. Every subsequent flip-flop is triggered by the output of the flip-flop that precedes it.

To build a basic binary ripple counter, we use **T (Toggle) Flip-Flops** (or J-K flip-flops with **J** and **K** tied to HIGH).

1. The first flip-flop toggles on every falling edge of the main clock.
2. The second flip-flop toggles only when the first flip-flop's output changes from 1 to 0.
3. This creates a "ripple" effect, where the transition of the LSB eventually trickles down to the Most Significant Bit (MSB).

10.4.2. Propagation Delay and the "Race" Problem

Because the clock signal must "ripple" through each flip-flop one after another, a significant physical problem arises: **Accumulated Propagation Delay**.

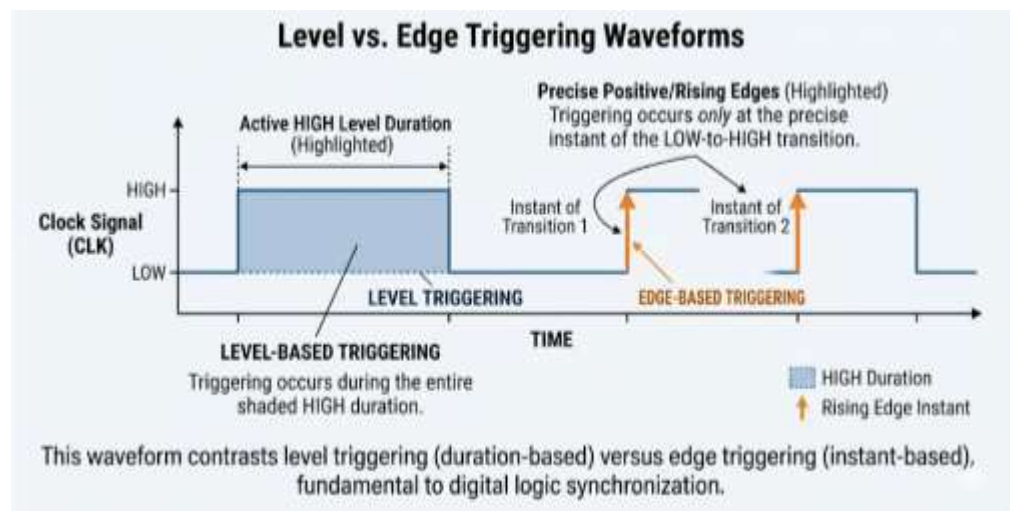


Figure 10.1 : Level vs. Edge Triggering Waveforms.

Every logic gate and flip-flop takes a few nanoseconds to react to an input. In a 16-bit ripple counter, the 16th flip-flop has to wait for 15 previous flip-flops to finish their transitions before it can move. At high speeds, this means the counter may briefly display "garbage" values because the first few bits have finished changing while the last few bits are still stuck in their old state. This limitation makes ripple counters unsuitable for high-speed modern microprocessors.

NOTES

✓ Check Your Progress 10.2.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. Why is an asynchronous counter also called a "Ripple Counter"?

.....

2. What is the main disadvantage of using asynchronous counters in high-speed systems?

.....

3. In a 4-bit ripple counter, which flip-flop (LSB or MSB) receives the external clock pulse?

.....

10.5. SYNCHRONOUS COUNTERS

To solve the delay issues of the ripple counter, engineers use **Synchronous Counters**. In this architecture, every single flip-flop in the counter is connected to the exact same master clock signal. Because they all "see" the clock at the same time, they all change their states simultaneously.

10.5.1. The Logic of Synchronous Counting

If all flip-flops are connected to the same clock, how do we prevent them all from toggling at once? The secret lies in the combinational logic connected to the J and K (or T) inputs.

- The 1st flip-flop toggles on every clock pulse.
- The 2nd flip-flop is only allowed to toggle if the 1st one is currently at 1.
- The 3rd flip-flop is only allowed to toggle if *both* the 1st and 2nd are at 1.

NOTES

This requires the use of AND gates to "look ahead" at the previous bits. By the time the next clock edge arrives, the logic gates have already decided which flip-flops should flip and which should stay the same.

10.5.2. Modulus Counters (MOD Counters)

A counter that cycles through all possible binary states is called a full-sequence counter. However, we often need counters that stop or reset early. For example, a digital clock needs a counter that resets after reaching 9 (to count seconds).

- **MOD-10 (Decade) Counter:** This counter has 10 states (0 through 9). When it hits 10 (1010 in binary), a NAND gate detects this pattern and immediately sends a "Clear" signal to all flip-flops, resetting them to 0000.
- **The Modulus (M):** The total number of unique states a counter passes through before returning to zero. A counter with n flip-flops has a maximum modulus of 2^n .

☒ Check Your Progress 10.3.

Notes : (a) Space is given below for your answer.
(b) Check your answer with the one given at the end of this unit.

1. What is the primary advantage of a Synchronous Counter over an Asynchronous Counter?
.....
.....
.....
2. How many unique states are in a MOD-6 counter?
.....
.....
.....
3. Which logic gate is typically used to reset a counter when it reaches a specific state?
.....
.....
.....

10.6. COUNTER APPLICATIONS AND FREQUENCY DIVISION

Counters are not just for "counting" in the traditional sense; they are the primary tools for managing time and frequency within a digital system.

10.6.1. Frequency Division

Every time a flip-flop toggles, its output frequency is exactly half of its input frequency. If you feed a 1 MHz clock into a 4-bit counter:

- The 1st output (LSB) will be 500 kHz.
- The 2nd output will be 250 kHz.
- The 3rd output will be 125 kHz.
- The 4th output (MSB) will be 62.5 kHz.

This allows a computer to take a single high-speed crystal oscillator and generate all the different "rhythms" needed for the keyboard, the USB ports, and the internal memory.

10.6.2. Digital Clocks and Timers

In a digital watch, a high-frequency crystal (usually 32,768 Hz) is fed into a long chain of counters. These counters divide the frequency down until they get exactly one pulse per second. This 1 Hz pulse then triggers a MOD-60 counter (for minutes) and eventually a MOD-12 or MOD-24 counter (for hours).

NOTES



Activity 10.1 : THE HUMAN RIPPLE COUNTER

Title : Simulating Propagation Delay

Objective : To visualize why ripple counters are slower than synchronous counters.

Materials : 4 PARTicipants (representing Flip-Flops A, B, C, and D) and a stopwatch.

Description :

1. The Ripple Setup:PARTicipant A watches the "Clock" (a leader clapping). When the leader claps, A taps B. Only when B is tapped can they tap C. Only when C is tapped can they tap D.
2. Observe: Notice the time it takes for the "signal" to reach PARTicipant D. This represents propagation delay.
3. The Synchronous Setup: Now, all 4 PARTicipants look directly at the leader. When the leader claps, everyone who needs to move moves at the exact same time.
4. Conclusion: Compare the speeds. The synchronous method is clearly faster and more coordinated, simulating why high-speed CPUs require synchronous logic.

10.7. SUMMARY

In this unit, we scaled our understanding of digital memory from individual flip-flops to the integrated systems that drive modern computing: Registers and Counters. We established that registers are critical temporary storage units, fundamental for holding digital words

NOTES

inside the CPU. We explored the four distinct shifting architectures—SISO, SIPO, PISO, and PIPO—and demonstrated how these configurations are used for everything from creating simple delay lines to packing parallel data for serial transmission over a USB cable. We also noted the mathematical significance of bidirectional shifting, where shifting bits left or right acts as high-speed hardware multiplication or division.

We then transitioned to the concept of sequential state cycling through Counters. We examined the Asynchronous (Ripple) Counter, noting its simple design using T flip-flops, but highlighting its fatal flaw: cumulative propagation delay. Because the clock signal must physically ripple through each logic gate, asynchronous counters are too slow and error-prone for modern, high-speed microprocessors.

To resolve this, we analyzed Synchronous Counters, where every flip-flop is triggered by the exact same clock pulse simultaneously, utilizing "look-ahead" combinational logic to determine state changes. Finally, we explored how Modulus (MOD) counters are engineered to truncate sequences (like a MOD-10 decade counter) and how counting circuits serve as the fundamental frequency dividers and timers that synchronize the entire computer system.

10.8. GLOSSARY

- **Asynchronous (Ripple) Counter:** A digital counter where the clock signal is only applied to the first flip-flop, and subsequent flip-flops are triggered by the output of the preceding one.
- **Frequency Division:** The process of using a flip-flop or counter to reduce a high-frequency clock signal down to a lower, mathematically related frequency.
- **Modulus (MOD):** The maximum number of unique, sequential states a counter cycles through before resetting to its initial state.
- **Parallel-In/Parallel-Out (PIPO):** A register architecture where all bits are loaded simultaneously and read simultaneously, acting as a direct storage buffer.
- **Propagation Delay:** The microscopic amount of time it takes for an electrical signal to physically travel through a logic gate or flip-flop and produce an output.
- **Register:** A connected group of flip-flops designed to store and manipulate a multi-bit binary word.
- **Serial-In/Parallel-Out (SIPO):** A register that takes a single stream of sequential data bits and converts them into a parallel output format.

- **Shift Register:** A specific type of register capable of shifting its stored binary data to the left or right upon receiving a clock pulse.
- **Synchronous Counter:** A counter where all flip-flops are connected to the same master clock, allowing all state changes to occur at the exact same instant.

10.9. ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 10.1.

1. 16 flip-flops are required, as each flip-flop stores exactly one bit.
2. A Serial-In/Parallel-Out (SIPO) register.
3. Shifting a binary number one position to the left is equivalent to multiplying the number by 2.

Check Your Progress 10.2

1. The clock pulse is applied only to the first flip-flop; the signal then physically "ripples" sequentially through the rest of the circuit.
2. Accumulated propagation delay, which can cause temporary incorrect outputs (race conditions) at high speeds.
3. The Least Significant Bit (LSB) flip-flop receives the external system clock.

Check Your Progress 10.3

1. All flip-flops update simultaneously, completely eliminating cumulative propagation delay and allowing for much faster clock speeds.
2. A MOD-6 counter has 6 unique states (typically 0 through 5).
3. A NAND gate is typically used to detect a specific binary pattern and trigger the asynchronous "Clear" pins on the flip-flops.

10.10. SUGGESTED READINGS - OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books aisectonline.com/User/UnitUrls?unitId=3834	

10.11. REFERENCE

1. Mano, M. M., & Ciletti, M. D. (2013). Digital Design: With an Introduction to the Verilog HDL. Pearson.
2. Floyd, T. L. (2015). *Digital Fundamentals*. Pearson Education.

10.12. EXERCISE

PART 1

A. Multiple choice question

1. A 4-bit shift register contains exactly how many flip-flops?
 - (a) 2
 - (b) 4
 - (c) 8
 - (d) 16
2. Which register configuration is best suited for converting data from a single serial wire into a parallel format for the CPU?
 - (a) SISO
 - (b) PIPO
 - (c) SIPO
 - (d) PISO
3. What is the primary cause of timing errors in asynchronous ripple counters?
 - (a) Using SRAM instead of DRAM
 - (b) Cumulative propagation delay across logic gates
 - (c) Faulty multiplexer connections
 - (d) Low system voltage
4. A counter designed to cycle through exactly ten unique states before resetting is called a:
 - (a) MOD-2 counter
 - (b) MOD-8 counter
 - (c) Decade counter
 - (d) PIPO counter
5. If a 100 kHz clock is fed into a single T flip-flop configured to toggle, what is the frequency of the output?
 - (a) 50 kHz
 - (b) 100 kHz
 - (c) 200 kHz
 - (d) 10 kHz

Answers with Explanations :

NOTES

1. **Answer : (b)**

Explanation : Each individual flip-flop is capable of storing exactly one bit of data, so a 4-bit register strictly requires 4 flip-flops.

2. **Answer : (c)**

Explanation : A Serial-In/Parallel-Out (SIPO) register takes a single sequential stream of bits and unpacks them to be read simultaneously on multiple output lines.

3. **Answer : (b)**

Explanation : Because the master clock only triggers the first flip-flop, subsequent flip-flops must wait for the physical output of the preceding one, causing a cascading delay.

4. **Answer : (c)**

Explanation : A decade counter cycles through exactly ten unique binary states (typically 0 through 9) before resetting, which defines a MOD-10 architecture.

5. **Answer : (a)**

Explanation : A toggling flip-flop requires two full clock cycles to complete one output cycle, effectively dividing the input frequency by exactly half.

B. Short answer type question

1. Define the primary function of a Shift Register in a digital system.
2. Explain the main operational difference between PIPO and PISO registers.
3. Why does a ripple counter suffer from cumulative propagation delay?
4. What does the Modulus (MOD) of a digital counter specifically indicate?
5. What mathematical operation is the electrical equivalent of shifting a binary number one position to the left?

PART 2

C. Long answer type question (200 words)

1. Describe the structural and architectural differences between an asynchronous counter and a synchronous counter, focusing specifically on clock signal distribution.
2. Detail the internal logic and operation of a MOD-10 decade counter. Explain exactly how it manages to reset itself before reaching its natural 4-bit hardware limit of 15.

NOTES**D. Long answer type question (300 words)**

1. Analyze the concept of frequency division using digital counters. Explain step-by-step how a high-speed crystal oscillator uses a chain of cascading flip-flops to ultimately generate a precise 1 Hz signal for a digital clock.
2. Compare and contrast the four primary types of shift registers (SISO, SIPO, PISO, PIPO). Provide a specific, real-world hardware application for each configuration to demonstrate its utility.

UNIT – 11

SHIFT REGISTERS

NOTES

UNIT STRUCTURE

- 11.1. Introduction
- 11.2. Objective
- 11.3. Introduction to Shift Registers
- 11.4. Types of Shift Registers
 - 11.4.1. Serial-In Serial-Out (SISO)
 - 11.4.2. Serial-In Parallel-Out (SIPO)
- 11.5. Summary
- 11.6. Glossary
- 11.7. Answers to check your progress
- 11.8. Suggested readings - OER
- 11.9. Reference
- 11.10. Exercise

11.1. INTRODUCTION

In the preceding units, we laid the foundation of digital memory by examining the flip-flop—a bistablemultivibrator capable of retaining a single bit of binary information (either a 0 or a 1). We also explored how grouping these individual memory cells together forms a standard register, allowing a computer to hold "words" of data (such as 8-bit bytes or 64-bit instruction codes) for immediate processing by the CPU.

However, in the dynamic environment of a microprocessor, static storage is only half of the equation. Data must not only be held; it must be constantly moved, manipulated, and transmitted. If a standard register is like a locked safe holding a document, a **Shift Register** is like a high-speed conveyor belt. It is a specialized, synchronized linear array of memory cells designed specifically to shift its stored binary data to adjacent cells upon the command of a clock pulse.

The invention of the shift register revolutionized digital electronics because it solved two massive engineering problems simultaneously: **Arithmetic manipulation** and **Data communication**.

First, within the Arithmetic Logic Unit (ALU) of a processor, moving a binary sequence one position to the left is the physical hardware equivalent of multiplying that number by two. Shifting it to the right divides it by two. This means shift registers allow CPUs to perform high-

NOTES

speed math without requiring complex, transistor-heavy adder networks.

Second, in the realm of communications, microprocessors generate data in parallel (say, 32 bits at once), but physical cables like USB, Ethernet, or HDMI often transmit data sequentially over a single wire. Shift registers act as the critical translators at the ports of your computer, converting parallel blocks of data into single-file lines for transmission, and vice versa.

In this unit, we will move beyond the simple concept of memory and dive into the kinetic mechanics of data shifting. We will explore the precise internal wiring that prevents data collision, the timing physics of the clock signal, and the architectural differences between serial and parallel data routing.

11.2. OBJECTIVES

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

- Define the fundamental architecture and operational mechanics of shift registers.
- Analyze the role of edge-triggered D flip-flops in cascading memory circuits.
- Evaluate the timing constraints, including setup time, hold time, and propagation delay.
- Explain the step-by-step shifting operation of Serial-In Serial-Out (SISO) registers.
- Describe the hardware configuration and networking applications of Serial-In Parallel-Out (SIPO) registers.

11.3. INTRODUCTION TO SHIFT REGISTERS

A shift register is formally defined as a synchronous sequential logic circuit comprising a linear cascade of flip-flops. To truly understand how it functions, we must deconstruct its architecture at the hardware level, looking at the choice of memory cells, the wiring configuration, and the strict physical laws of clock timing that govern its operation.

11.3.1. The Cascade Architecture and the D Flip-Flop

The term "cascade" refers to the specific physical layout of the circuit: the components are arranged in a daisy-chain. The main output of the first flip-flop is physically soldered to the primary input of the second flip-flop. The output of the second is wired to the input of the third, forming a continuous path.

While shift registers *can* theoretically be built using J-K or S-R flip-flops, modern digital design relies almost exclusively on the **D (Data) Flip-Flop**.

The D flip-flop is chosen because of its transparent predictability. It has only one data input pin (**D**). Whatever logic level (High or Low voltage) is present at the **D** pin right before the clock pulse triggers will be faithfully captured and pushed to the **Q** output pin immediately

after the trigger. It inherently eliminates the "invalid state" problem found in S-R flip-flops, ensuring that no matter what data is shifting through the register, the circuit will never crash or output undefined voltage levels.

If we want to build a 4-bit shift register, we align four D flip-flops: FF_0 , FF_1 , FF_2 and FF_3 . The Q output of FF_0 connects to the D input of FF_1 , and so on. The D input of FF_0 serves as the main entry point for new data entering the system, while the Q output of FF_3 serves as the final exit point.

11.3.2. The Master Clock and Edge-Triggering

The defining characteristic that makes this chain of flip-flops a "shift register" rather than a chaotic mess of moving voltages is the **Universal Clock Network**.

In a shift register, the clock input pins of *every single flip-flop* in the cascade are wired together in parallel to a single, master system clock. This means every memory cell evaluates its input and updates its output at the exact same physical instant.

Furthermore, these flip-flops must be **Edge-Triggered** (usually on the rising, or positive, edge of the clock signal).

- If the flip-flops were *level-triggered* (meaning they allow data to pass through as long as the clock voltage is HIGH), a single 1 entering the first flip-flop would blast entirely through the entire 4-bit register in a fraction of a nanosecond, destroying any existing data. This is known as "race-through."
- By using edge-triggering, the flip-flops are completely "deaf" to their inputs for 99% of the time. They only sample their inputs for a microscopic fraction of a nanosecond—exactly when the clock voltage jumps from 0V to 5V. This ensures that the data moves exactly *one* position per clock cycle, like a highly synchronized bucket brigade.

11.3.3. Timing Constraints and Physics of Shifting

To design a shift register that functions correctly at high frequencies (like the gigahertz speeds inside modern CPUs), engineers must calculate the strict timing physics of the transistors. There are three critical parameters:

1. **Setup Time** (t_{setup}): The data bit arriving at the D input must be stable at a solid 0V or 5V for a specific amount of time *before* the clock edge hits. If the voltage is still fluctuating when the clock triggers, the flip-flop might capture an invalid state (metastability).
2. **Hold Time** (t_{hold}): The data at the D input must remain stable for a brief fraction of a nanosecond *after* the clock edge hits, ensuring the internal transistors have enough time to permanently latch the new value.

NOTES

3. **Propagation Delay** (t_{pd}): Once the clock edge hits, it takes physical time for the electrons to travel through the silicon gates and change the voltage at the **Q** output.

For a shift register to work properly without data crashing into itself, the propagation delay of the first flip-flop *must* be greater than the hold time of the second flip-flop ($t_{pd} > t_{hold}$). If **FF₀** changes its output too fast, **FF₁** might accidentally grab the *new* bit instead of the *old* bit it was supposed to shift.

11.3.4. The Mathematical Power of Shifting

Beyond just moving data, shifting is a mathematical operation. Because computers use the Base-2 (Binary) number system, shifting a bit string alters its place value.

Consider the 4-bit binary number for the decimal value three: 0011.

- **Shift Left:** If we shift all bits one position to the left, adding a 0 to the empty space on the right, we get 0110. The binary number 0110 is equal to the decimal value six. We just performed hardware multiplication ($3 * 2 = 6$) using zero logic gates, only a single clock pulse!
- **Shift Right:** If we take 0110 and shift it one position to the right, we get 0011 (three). We just performed hardware division ($6 / 2 = 3$).

This principle is the foundation of high-speed algorithmic processing in digital signal processors (DSPs) and graphics cards.

☒ Check Your Progress 11.1.

Notes : (a) Space is given below for your answer.
(b) Check your answer with the one given at the end of this unit.

1. Why is the edge-triggered D flip-flop specifically chosen for constructing shift registers?
.....
.....
.....
2. What would happen to the data if the flip-flops in a shift register were level-triggered instead of edge-triggered?
.....
.....
.....
3. What mathematical operation is physically executed when a binary word is shifted one position to the right?
.....
.....
.....

11.4. TYPES OF SHIFT REGISTERS

The classification of shift registers is based entirely on the interface through which data is written into the memory cells (the input) and how that data is subsequently read by the rest of the digital system (the output).

NOTES

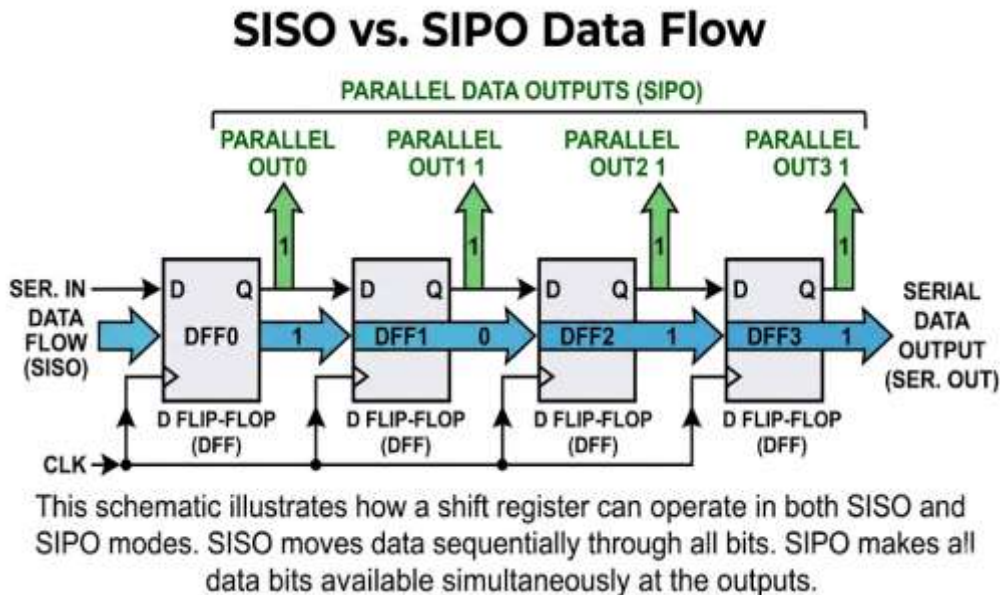


Figure 11.1 : SISO vs. SIPO Data Flow.

In digital electronics, data can be transported in two fundamental ways:

1. **Serial Transmission:** Data bits are sent sequentially, one after another, over a single physical wire. This is highly cost-effective and immune to the electromagnetic cross-talk that plagues multi-wire cables, making it ideal for long-distance communication (e.g., USB, Ethernet, fiber optics).
2. **Parallel Transmission:** Multiple data bits are sent simultaneously over multiple, separate physical wires (a "bus"). This method is exceptionally fast but requires massive amounts of wiring, making it suitable only for short distances, such as the internal datapath between a CPU and its RAM.

By mixing and matching these input and output methods, engineers create four primary shift register architectures:

- Serial-In Serial-Out (SISO)
- Serial-In Parallel-Out (SIPO)
- Parallel-In Serial-Out (PISO)
- Parallel-In Parallel-Out (PIPO)

In this unit, we will deeply analyze the first two variants, which form the backbone of sequential data buffering and serial-to-parallel conversion.

NOTES

11.4.1. Serial-In Serial-Out (SISO)

The Serial-In Serial-Out (SISO) shift register is the most structurally fundamental configuration. It is the electronic equivalent of a single-file pipeline. It accepts a stream of binary data one bit at a time on a single input line, shifts it through a linear chain of flip-flops, and outputs it one bit at a time on a single output line.

1. Architectural Wiring

Let us examine a 4-bit SISO register constructed from four D flip-flops, designated as FF_0 (the input stage), FF_1 , FF_2 , and FF_3 (the output stage).

- The external data source is connected exclusively to the D input of FF_0 .
- The Q output of FF_0 is wired directly to the D input of FF_1 .
- The Q output of FF_1 is wired to the D input of FF_2 .
- The Q output of FF_2 is wired to the D input of FF_3 .
- The Q output of FF_3 serves as the sole external output line for the entire circuit.
- A universal clock signal is connected in parallel to the clock inputs of all four flip-flops.

2. Clock-by-Clock Operational Analysis

To understand the precise mechanics, let us trace the insertion of the 4-bit binary word 1011 into an initially cleared SISO register. We will assume the data enters Left-to-Right, meaning the rightmost bit (the Least Significant Bit, 1) will enter first.

- **Initial State :** Before any clock pulses, the outputs of all flip-flops (Q_0, Q_1, Q_2, Q_3) are stable at **0000**.
- **Clock Pulse 1:** The first bit of our sequence (1) is applied to the D input of FF_0 . On the rising edge of the clock, FF_0 captures this 1. Simultaneously, FF_1 , FF_2 , and FF_3 capture the 0s that were previously held by their left-hand neighbors. The internal state is now **1000**.
- **Clock Pulse 2:** The second bit (1) is applied to the input. On the clock edge, FF_0 captures the new 1. The old 1 inside FF_0 is shifted into FF_1 . The state is now **1100**.
- **Clock Pulse 3:** The third bit (0) is applied. Everything shifts one position to the right. The state becomes **0110**.
- **Clock Pulse 4:** The final bit (1) is applied. The register shifts again. The internal state is now **1011**.

At this exact moment, the 4-bit word is fully loaded. However, because this is a SISO register, the CPU cannot simply "see" the entire word. Only the very last bit (the original 1 that has made it to FF_3) is visible on the output wire.

3. Data Retrieval and Digital Delay

To retrieve the full word 1011 from the SISO register, the system must apply *four additional clock pulses*, capturing the output of FF_3 one bit at a time as the data is pushed off the end of the cascade. Therefore, a generic N -bit SISO register requires exactly N clock cycles to write the data and N clock cycles to read the data.

Because the data entering a SISO register is physically delayed by a specific number of clock cycles before it exits, this circuit is predominantly used as a **Digital Delay Line**. Engineers can calculate the exact time delay Δt of the signal using the formula:

$$\Delta t = N \times T_{\{clk\}}$$

Where N is the number of flip-flops in the cascade and $T_{\{clk\}}$ is the time period of one clock cycle. If a 1 MHz clock (which has a period of 1 microsecond) drives an 8-bit SISO register, any data bit entering the circuit will be cleanly and perfectly delayed by exactly 8 microseconds before appearing at the output. This is crucial in audio processing and signal synchronization, where disparate data streams must be forcefully aligned in time.

NOTES

☒ Check Your Progress 11.2.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. In a 4-bit SISO register, why is the CPU unable to read the entire 4-bit word immediately after the fourth clock pulse?

.....

2. If an engineer cascades eight D flip-flops to build a SISO register, how many total clock pulses are required to completely load an 8-bit word and then shift it entirely out of the circuit?

.....

3. Calculate the digital delay time Δt provided by a 16-bit SISO register driven by a clock with a period of 50 nanoseconds.

.....

NOTES

11.4.2. Serial-In Parallel-Out (SIPO)

While the SISO register is an excellent delay mechanism, it is a massive bottleneck for data processing. A CPU cannot afford to wait cycle after cycle just to read a single byte of data. This bottleneck is resolved by the Serial-In Parallel-Out (SIPO) shift register, arguably the most vital logic circuit in modern computer communications.

1. Architectural Wiring

The internal input structure of a SIPO register is identical to a SISO register. The data is still fed sequentially, one bit at a time, into the **D** input of the first flip-flop (**FF₀**). The flip-flops are still cascaded, and they still share a universal synchronous clock.

The radical difference lies in the output wiring. Instead of burying the internal states of the flip-flops, engineers solder a physical external output wire to the **Q** pin of *every single flip-flop* in the chain. In a 4-bit SIPO register, there are four independent, parallel output lines (**\$Q_0**, **Q_1**, **Q_2**, and **\$Q_3**) exposed to the rest of the computer system.

2. The Deserialization Process

Because the internal states are physically exposed, the read mechanics of the SIPO register are entirely different from the SISO register. Let us trace the loading of the word 1011 again.

For the first three clock pulses, the bits stream in serially. The parallel output wires will show the incomplete shifting states (**1000, then 1100, then 0110**), which the CPU is programmed to ignore.

On the **fourth clock pulse**, the final bit is shifted in. The internal state becomes **1011**.

At this exact nanosecond, the entire 4-bit word 1011 is simultaneously available across the four parallel output wires. The CPU can read the entire word in a single operation. It requires **zero additional clock pulses** to retrieve the data.

3. Real-World Application: The UART and USB

This ability to take a single-file line of bits and instantly present them as a unified parallel block makes the SIPO register the ultimate "Deserializer."

Every time you plug a Universal Serial Bus (USB) device into your computer, data flows through the cable serially (one bit at a time) at immense speeds. However, the internal data bus of your motherboard requires data in parallel chunks (typically 64 bits wide).

A dedicated microchip called a UART (Universal Asynchronous Receiver-Transmitter) sits at the USB port. The core of this UART is a massive SIPO shift register. As the high-speed sequential bits stream out of the USB cable, they are clocked into the SIPO register. The moment the register is full (e.g., 64 bits have been

loaded), a control signal tells the CPU, "The parallel word is ready!" The CPU instantly reads all 64 output pins of the SIPO register simultaneously, clearing the register for the next incoming wave of serial data. Without the SIPO shift register, modern computer networking, internet routing, and peripheral connectivity would be physically impossible.

Check Your Progress 3.2.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. What is the fundamental difference in the physical hardware wiring between a SISO register and a SIPO register?

.....

2. Once an 8-bit word is fully loaded into an 8-bit SIPO register, how many clock pulses does the CPU need to read the data?

.....

3. Explain the specific role of a SIPO register in a Universal Serial Bus (USB) interface.

.....



Activity 11.1. Tracing the SIPO State Matrix

Objective : To manually map the internal states of a SIPO register during a data stream reception to understand parallel availability.

Description : Assume you are the CPU monitoring a 5-bit SIPO shift register. The serial data line is feeding the sequence 1-0-1-1-0 (entering left-to-right, so 0 enters first).

1. Draw a grid with 6 rows (Initial State + 5 Clock Pulses) and 5 columns (representing flip-flops Q0 through Q4).
2. Fill the "Initial State" row with all 0s.
3. For Clock Pulse 1, place the first input bit (0) in Q0. Shift the remaining 0s to the right.
4. For Clock Pulse 2, place the second input bit (1) in Q0. The previous 0 shifts to Q1.
5. Continue this manual shifting for all 5 clock pulses.

NOTES

Reflection :

Look at your final row. You have essentially acted as the clock signal, charting the physical movement of the bits. Notice how the parallel output is completely meaningless garbage data to the CPU until that exact 5th clock pulse strikes, at which point the full valid word instantly materializes across the columns.

11.5. SUMMARY

In this unit, we conducted a rigorous hardware-level examination of Shift Registers, the critical synchronized logic circuits responsible for data movement, arithmetic manipulation, and sequential-to-parallel translation. We established that the architecture relies on cascaded D (Data) flip-flops, specifically chosen for their transparent predictability and immunity to invalid logic states. We explored the strict physical timing constraints of edge-triggering, proving that setup time, hold time, and propagation delay must be meticulously balanced to prevent data "race-through" and metastability.

Transitioning from physics to architecture, we analyzed the Serial-In Serial-Out (SISO) register. We determined that because data both enters and exits sequentially through a single physical line, an N -bit SISO requires exactly $2N$ clock cycles to process a word, making it highly mathematically predictable and ideal for use as a precise digital delay line ($\Delta t = N * T_{clk}$). Finally, we deconstructed the Serial-In Parallel-Out (SIPO) register, the definitive hardware deserializer. By wiring independent external outputs to the internal states of every flip-flop in the cascade, the SIPO register instantly converts sequential data streams into multi-bit parallel blocks. This completely eliminates the secondary read cycles required by SISO architectures, rendering the SIPO register the fundamental bridging hardware for modern networking protocols, UART communication, and USB interfaces.

11.6. GLOSSARY

- **Cascade:** The physical daisy-chain wiring configuration where the Q output of one flip-flop acts as the direct D input for the immediate next flip-flop.
- **Digital Delay Line:** An application of a SISO register where a signal is perfectly paused for a specific duration, calculated by multiplying the number of cascaded flip-flops by the clock period.
- **Edge-Triggering:** A clocking mechanism where the flip-flop only reads its input during the microscopic nanosecond that the clock voltage transitions (e.g., from LOW to HIGH), preventing data race-through.
- **Metastability:** An unstable hardware state where a flip-flop fails to settle on a clean 0 or 1 voltage, usually caused by violating the setup or hold time constraints.

- **Propagation Delay(t_{pd}):** The physical time it takes for electrons to travel through the silicon gates of a flip-flop and change the logic level at the output pin after a clock edge.
- **UART (Universal Asynchronous Receiver-Transmitter):** A hardware microchip utilizing SIPO registers to translate serial data from external cables into parallel data for the CPU databus.

11.7. ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 11.1

1. The D flip-flop is chosen because its single-input design guarantees stable data capture and inherently prevents the invalid/forbidden hardware states that can crash S-R flip-flops during shifting.
2. If they were level-triggered, a single input bit would race entirely through the entire register chain in a fraction of a nanosecond while the clock was HIGH, completely destroying the shifting sequence.
3. Shifting a binary word one position to the right is the hardware equivalent of dividing that number's decimal value by two.

Check Your Progress 11.2

1. Because a SISO register only has one physical output wire connected to the very last flip-flop; the internal states of the other flip-flops are not exposed to the CPU.
2. It requires exactly 16 clock pulses: 8 pulses to completely shift the data in, and 8 more pulses to sequentially shift the data out.
3. The digital delay (Δt) is calculated as $N * T_{CLK}$. Therefore, $16 * 50 \text{ ns} = 800 \text{ nanoseconds}$.

Check Your Progress 11.3

1. In a SIPO register, individual output wires are soldered to the **Q** outputs of *every single flip-flop* in the cascade, whereas a SISO register only has one single output wire at the very end of the chain.
2. It requires zero additional clock pulses. The moment the 8th pulse loads the final bit, the entire word is instantly available across the 8 parallel output wires.
3. The SIPO register inside a UART catches the high-speed, single-file sequential bits from the USB cable and organizes them into a wide parallel block so the CPU can read the data instantly.

11.8. SUGGESTED READINGS – OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectonline.com/User/UnitUrls?unitId=3835	

NOTES

11.9. REFERENCE

1. Floyd, T. L. (2015). Digital Fundamentals. Pearson Education.
2. Mano, M. M., & Ciletti, M. D. (2013). *Digital Design: With an Introduction to the Verilog HDL*. Pearson.
3. Tocci, R. J., Widmer, N. S., & Moss, G. L. (2017). *Digital Systems: Principles and Applications*. Pearson.

11.10. EXERCISE

PART 1

A. Multiple choice question

1. What prevents a single input bit from instantly traveling through all cascaded flip-flops in a shift register simultaneously?
 - (a) Using SRAM instead of DRAM.
 - (b) The edge-triggering design of the master clock.
 - (c) The high voltage of the power supply.
 - (d) Using S-R flip-flops instead of D flip-flops.
2. A 32-bit Serial-In Serial-Out (SISO) register is driven by a 1 MHz clock (period = 1 μ s). What is the exact digital delay time from input to output?
 - (a) 1 μ s
 - (b) 16 μ s
 - (c) 32 μ s
 - (d) 64 μ s
3. Which register configuration fundamentally acts as a hardware "Deserializer" for incoming network data?
 - (a) SIPO
 - (b) SISO
 - (c) PISO
 - (d) PIPO
4. If a 4-bit shift register currently holds the binary value 0100 (decimal 4), and all bits are shifted exactly one position to the left (with a 0 entering the right), what is the new decimal value?
 - (a) 2
 - (b) 4
 - (c) 8
 - (d) 16
5. In order for a cascaded shift register to function correctly without data crashing, the propagation delay (t_{pd}) of the

first flip-flop must be strictly greater than what parameter of the second flip-flop?

- (a) The clock frequency
- (b) The hold time (t_{hold})
- (c) The power consumption
- (d) The setup time (t_{setup})

Answers with Explanations:

1. **Answer : (b)**

Explanation : Edge-triggering ensures the flip-flops are only "open" to receive data for a microscopic fraction of a nanosecond, allowing data to move exactly one position per clock cycle.

2. **Answer : (c)**

Explanation : The delay is calculated by multiplying the number of flip-flops ($N=32$) by the clock period ($1\mu\text{s}$), resulting in exactly $32\mu\text{s}$.

3. **Answer : (a)**

Explanation : A Serial-In Parallel-Out (SIPO) register takes a single sequential stream of data and instantly outputs it as a simultaneous parallel multi-bit word.

4. **Answer : (c)**

Explanation : Shifting the binary value 0100 (4) to the left results in 1000, which is the binary representation of decimal 8 (effectively multiplying by 2).

5. **Answer : (b)**

Explanation : If the first flip-flop changes its output faster than the second flip-flop's hold time requires the old data to remain stable, the second flip-flop will erroneously capture the new data bit instead of the old one.

B. Short answer type question

1. Explain the specific architectural wiring that defines a "cascaded" digital circuit.
2. Why is the D (Data) flip-flop heavily preferred over the S-R flip-flop when engineering shift registers?
3. Define "setup time" (t_{setup}) and why violating it causes metastability in a memory circuit.
4. What is the fundamental difference in reading data from a SISO register versus reading from a SIPO register?
5. How does a shift register perform high-speed hardware division without using a complex logic ALU?

NOTES

PART 2

C. Long answer type question (200 words)

1. Describe the phenomenon of "race-through" in sequential logic circuits. Explain exactly how the implementation of edge-triggered flip-flops, combined with a synchronized master clock, solves this catastrophic hardware failure.
2. Contrast the primary engineering use-cases for Serial-In Serial-Out (SISO) and Serial-In Parallel-Out (SIPO) shift registers. Provide a detailed explanation of why a SISO register is perfectly suited for audio signal delay, while a SIPO register is mandated for USB peripheral communication.

PART 3

D. Long answer type question (300 words)

1. Provide a comprehensive, clock-by-clock analysis of how a UART microchip utilizes a Serial-In Parallel-Out (SIPO) shift register to bridge the communication gap between a serial network cable and a parallel CPU databus. In your explanation, trace the insertion of an 8-bit sequential data stream and define the exact moment the data becomes valid for the CPU to retrieve without requiring additional clock pulses.
2. Analyze the complex physics of timing constraints within a high-frequency shift register. Deconstruct the delicate balance required between the propagation delay (t_{pd}) of the transmitting flip-flop and the setup (t_{setup}) and hold times (t_{hold}) of the receiving flip-flop. Evaluate what physical errors occur at the transistor level when these constraints are violated during gigahertz-level overclocking.

UNIT – 12

COUNTERS AND STATE DIAGRAMS

NOTES

UNIT STRUCTURE

- 11.1. Introduction
- 11.2. Objective
- 12.3. Counters
 - 12.3.1. Asynchronous Counters
 - 12.3.2. Synchronous Counters
- 12.4. State Diagrams and Applications
 - 12.4.1. State Transition Tables
 - 12.4.2. Applications in Digital Systems
- 12.5. Summary
- 12.6. Glossary
- 12.7. Answers to check your progress
- 12.8. Suggested readings - OER
- 12.9. Reference
- 12.10. Exercise

12.1. INTRODUCTION

While previous units introduced the physical hardware of flip-flops and basic storage registers, the true power of digital logic lies in its ability to follow complex, algorithmic sequences. A computer does not merely store data; it progresses through distinct steps to execute programs. This sequential progression is mathematically modeled using the concepts of **State Machines**.

In this unit, we revisit counters, but we elevate our perspective from basic hardware wiring to formal architectural design. We will learn how to abstract the behavior of counters and other sequential circuits into **State Diagrams**. A state diagram acts as the structural blueprint for a digital system, visually mapping out every possible condition (state) a circuit can hold and the specific inputs required to transition from one state to another. By mastering state diagrams and their corresponding transition tables, you will possess the engineering toolkit required to design automated systems ranging from simple digital stopwatches to the complex control units found within modern microprocessors.

NOTES

12.2. OBJECTIVES

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

- Define the operational mechanics and design principles of digital counters.
- Distinguish between the timing characteristics of asynchronous and synchronous counters.
- Construct state diagrams to visually represent the sequential behavior of memory circuits.
- Formulate state transition tables to map present states to next states based on inputs.
- Apply state machine concepts to design practical digital systems like sequence detectors.

12.3. COUNTERS

A counter is a specialized sequential logic circuit designed to progress through a predetermined sequence of binary states upon the application of clock pulses. Counters are the fundamental timekeepers and sequence generators in digital electronics.

12.3.1. Asynchronous Counters

As discussed in basic memory concepts, an asynchronous counter (often called a ripple counter) is characterized by its decentralized clocking scheme.

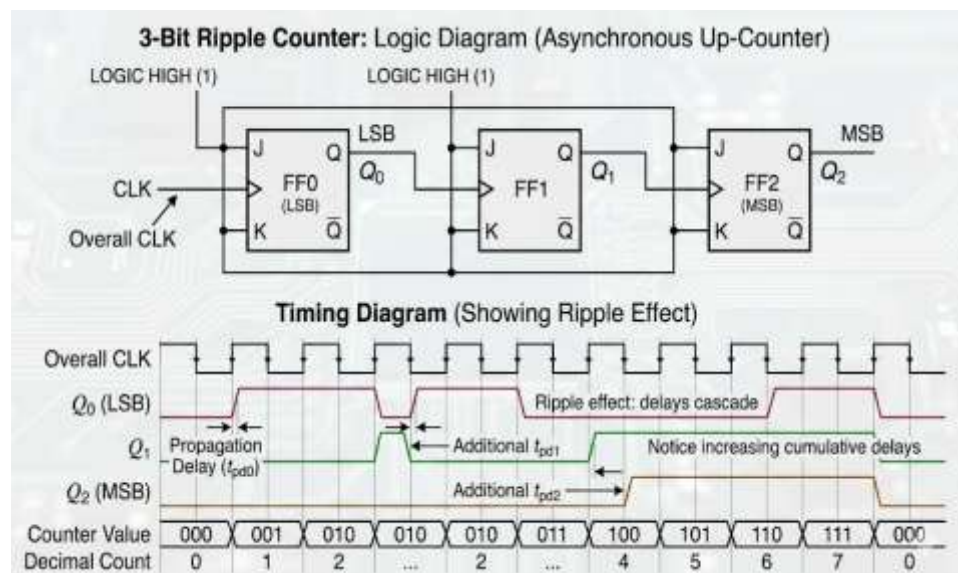


Figure 12.1 : 3-Bit Ripple Counter Timing Diagram.

- **The Architecture :** The master external clock is connected exclusively to the clock input of the very first flip-flop (the Least Significant Bit, or LSB). The output of this first flip-flop serves as the clock input for the second flip-flop, the output of the second clocks the third, and so on.

- **The Drawback:** Because the signal must propagate physically through each flip-flop one at a time, asynchronous counters suffer from cumulative **propagation delay**. If an engineer is designing a 16-bit asynchronous counter for a high-frequency system, the final flip-flop might not change state until long after the initial clock pulse has finished. This creates transient, invalid outputs—known as "glitches"—which make asynchronous counters highly unsuitable for precision, high-speed computing.
- **The Advantage:** Their primary advantage lies in their simplicity. They require less complex wiring and no additional combinational logic gates, making them highly efficient for low-speed frequency division tasks (like stepping down a crystal oscillator's frequency to keep time in a digital wristwatch).

12.3.2. Synchronous Counters

To resolve the timing limitations of the ripple counter, engineers employ the synchronous counter architecture.

- **The Architecture:** In a synchronous counter, the master clock signal is routed parallelly to the clock input of *every single flip-flop* in the circuit simultaneously.
- **The Logic:** Because all flip-flops trigger at the exact same microsecond, cumulative propagation delay is entirely eliminated. However, to ensure the counter follows the correct binary sequence (e.g., \$00 \rightarrow 01 \rightarrow 10 \rightarrow 11\$), external combinational logic (usually AND gates) must be placed before the \$J-K\$ or \$T\$ inputs of the flip-flops. This combinational logic acts as a "look-ahead" mechanism, constantly evaluating the current state of all flip-flops to determine exactly which ones need to toggle on the upcoming clock edge.

☒ **Check Your Progress 12.1.**

Notes : (a) Space is given below for your answer.
(b) Check your answer with the one given at the end of this unit.

1. What causes the "ripple effect" in an asynchronous counter?
.....
.....
.....

2. Why do synchronous counters completely eliminate cumulative propagation delay?
.....

NOTES

.....

.....

3. Which type of counter requires more complex external logic gating: synchronous or asynchronous?

.....

.....

.....

12.4. STATE DIAGRAMS AND APPLICATIONS

Designing a synchronous counter simply by guessing the logic gate connections becomes impossible as the sequence grows larger. We must use a formal design methodology based on **Finite State Machines (FSM)**.

A state machine is an abstract mathematical model of computation that can be in exactly one of a finite number of states at any given time. The visual representation of this model is the **State Diagram**.

- **Circles (Nodes):** Represent the unique physical states of the circuit. In a 2-bit counter, the states are 00, 01, 10, and 11.
- **Directed Arrows (Edges):** Represent the transition from the Present State to the Next State when a clock pulse occurs. The arrows are often labeled with the input variables that trigger the transition and the output variables that result.

12.4.1. State Transition Tables

Once a state diagram is drawn, the visual information must be translated into a mathematical format known as a **State Transition Table**. This table maps the "Present State" variables against all possible inputs to dictate the "Next State" variables.

If an engineer is designing a MOD-5 counter (which counts from 000 to 100 and then resets), the state transition table clearly defines that if the Present State is \$100\$, the Next State must be forcefully routed back to \$000\$, ignoring the natural binary progression to \$101\$.

Furthermore, to physically build the circuit, engineers expand this into an **Excitation Table**. By cross-referencing the desired Next State with the characteristic tables of the chosen flip-flops (like D or J-K), the excitation table reveals the exact Boolean logic expressions required to wire the input pins of the hardware.

12.4.2. Applications in Digital Systems

State diagrams are not limited to simple numeric counting; they govern almost all automated digital processes.

- **Sequence Detectors:** Imagine a digital lock that only opens when the specific binary sequence 1011 is entered. A state diagram is used to map the progression: one state remembers the first 1, the next state remembers 10, the third state remembers 101, and the final state unlocks the door. If a wrong bit is entered at any point, the state diagram dictates an immediate return to the default locked state.
- **Traffic Light Controllers:** A traffic intersection is a classic state machine. State A (North/South Green, East/West Red) transitions to State B (North/South Yellow) based on a counter timer input. The state diagram ensures there is absolutely no physical path to an invalid state (like all directions being Green simultaneously).
- **Vending Machines:** The machine exists in a state tracking the total coins inserted. Inserting a coin acts as an input that triggers an arrow to a new state of higher value. Reaching the target value state triggers an output (dispensing the item) and a return to the zero state.

NOTES

☒ Check Your Progress 12.2.

Notes : (a) Space is given below for your answer.
(b) Check your answer with the one given at the end of this unit.

1. In a state diagram, what do the circles and the directed arrows represent?
.....
.....
.....
2. What is the primary purpose of a state transition table?
.....
.....
.....

NOTES

3. Give an example of a real-world system that functions as a finite state machine.
-
-
-



Activity 12.1. Mapping a Sequence

Objective : To visually represent a custom counter sequence using a state diagram.

Description : On a piece of paper, draw a state diagram for a strange custom counter that counts only even numbers up to 6.

1. Draw four circles and label them with the binary states: 000, 010, 100, and 110.

2. Draw a directed arrow from 000 to 010.

3. Draw an arrow from 010 to 100.

4. Draw an arrow from 100 to 110.

5. Draw the final arrow looping from 110 all the way back to 000.

Reflection :

You have just designed the logic flow for a synchronous MOD-4 even-number counter. To physically build this, you would translate these arrows into an excitation table.

12.5. SUMMARY

In this unit, we advanced from the physical architecture of counters into the theoretical design framework of sequential logic. We reviewed the critical differences between asynchronous ripple counters, which suffer from cascading timing delays, and synchronous counters, which use parallel clocking and "look-ahead" logic to update all memory elements instantly. To design these complex synchronous systems, we introduced Finite State Machines and their visual blueprint: the State Diagram. We learned that by mapping Present States to Next States using transition and excitation tables, engineers can construct highly complex, error-free automated systems—such as traffic controllers and digital sequence detectors—ensuring that digital hardware executes algorithmic logic flawlessly.

12.6. GLOSSARY

- **Excitation Table:** A design table that shows the specific input signals required by a flip-flop to force a transition from a given Present State to a desired Next State.

- **Finite State Machine (FSM):** A mathematical model of computation representing a system that can be in exactly one of a finite number of states at any given time.
- **Next State:** The binary condition that a sequential circuit will transition into immediately following the next active clock edge.
- **Present State:** The current, stable binary condition held by the memory elements of a sequential circuit before a clock edge occurs.
- **State Diagram:** A graphical representation of a state machine, utilizing circles to denote states and directed arrows to indicate the logical transitions between them.
- **State Transition Table:** A tabular representation mapping all possible Present States and system inputs to their corresponding Next States and system outputs.

12.7. ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 12.1

1. It is caused by the master clock only triggering the first flip-flop; subsequent flip-flops are triggered by the output of the preceding one, causing the signal to "ripple" sequentially.
2. Because the master clock signal is wired directly and simultaneously to every flip-flop in the circuit, causing all state changes to happen in the exact same microsecond.
3. Synchronous counters require more complex external logic gating to pre-calculate which flip-flops should toggle on the upcoming universal clock pulse.

Check Your Progress 12.2

1. The circles represent the distinct binary states of the circuit, and the arrows represent the transitions between those states triggered by clock pulses and inputs.
2. Its purpose is to mathematically map the visual flow of a state diagram, defining exactly what the Next State will be based on the Present State and current inputs.
3. A traffic light controller, a digital combination lock, or a vending machine.

12.8. SUGGESTED READINGS – OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectionline.com/User/UnitUrls?unitId=3836	

NOTES

12.9. REFERENCE

1. Mano, M. M., & Ciletti, M. D. (2013). Digital Design: With an Introduction to the Verilog HDL. Pearson.
2. Tocci, R. J., Widmer, N. S., & Moss, G. L. (2017). *Digital Systems: Principles and Applications*. Pearson.

12.10. EXERCISE

PART 1

A. Multiple choice question

1. In a standard state diagram, what does a directed arrow pointing from one circle to another represent?
 - (a) A power surge in the circuit.
 - (b) A physical wire connecting two flip-flops.
 - (c) A logical transition from a Present State to a Next State.
 - (d) The reset pin of a sequential logic gate.
2. Which design tool is specifically used to determine the exact hardware input logic required to drive a flip-flop from a Present State to a desired Next State?
 - (a) State Diagram
 - (b) Excitation Table
 - (c) Boolean Algebra
 - (d) K-Map
3. Why are synchronous counters preferred over asynchronous counters in high-frequency computer processors?
 - (a) They consume less electricity.
 - (b) They require fewer logic gates to build.
 - (c) They completely eliminate cumulative propagation delay.
 - (d) They use SRAM instead of DRAM.
4. A digital combination lock that requires the exact sequence 1-0-1 to open is best modeled as a:
 - (a) Combinational Logic Circuit
 - (b) Ripple Counter
 - (c) Finite State Machine (FSM)
 - (d) Parallel-In/Serial-Out Register
5. In the context of a sequential circuit, what does the term "Present State" refer to?
 - (a) The physical location of the circuit board.

- (b) The binary data currently held in the circuit's memory elements.
- (c) The state of the circuit immediately after the next clock pulse.
- (d) The voltage level of the system clock.

Answers with Explanations:1. **Answer : (c)**

Explanation : Directed arrows in a state diagram represent the dynamic transition of the system as it moves from its current state to a new state based on clock pulses and inputs.

2. **Answer : (b)**

Explanation : An excitation table works backward from a state transition table; it tells the engineer exactly what logic inputs (e.g., \$J\$ and \$K\$ values) must be provided to force a specific state change.

3. **Answer : (c)**

Explanation : Synchronous counters route the clock to all memory elements simultaneously, ensuring all bits update at the exact same instant, which is crucial for high-speed synchronization.

4. **Answer : (c)**

Explanation : Because the lock must "remember" the history of previous inputs to know if the correct sequence has been entered, it operates as a Finite State Machine.

5. **Answer : (b)**

Explanation : The Present State represents the stable, real-time binary condition currently stored inside the flip-flops before any new clock edge triggers a change.

B. Short answer type question

1. Briefly define what a Finite State Machine (FSM) is in digital logic.
2. What specific timing problem in asynchronous counters makes them unsuitable for complex CPUs?
3. Contrast the meaning of "Present State" versus "Next State" in a state transition table.
4. How does combinational logic (like AND/OR gates) interact with flip-flops in a synchronous counter?
5. Give a brief example of how a state diagram prevents a traffic light controller from reaching an unsafe condition.

NOTES**PART 2****C. Long answer type question (200 words)**

1. Describe the step-by-step methodology an engineer uses to design a custom synchronous counter, starting from the creation of a state diagram and ending with the implementation of an excitation table.
2. Explain the fundamental differences between an asynchronous ripple counter and a synchronous counter, focusing specifically on clock distribution and propagation delay.

PART 3**D. Long answer type question (300 words)**

1. Analyze the application of Finite State Machines in sequence detection. Design a conceptual state diagram for a digital system that scans a stream of bits and only outputs a logical 1 when it detects the exact sequence 110. Map out the states and transitions required for success and failure.
2. Deconstruct the purpose and mechanical function of State Transition Tables and Excitation Tables. Explain why a State Diagram alone is insufficient for physically wiring a digital circuit and how these tables bridge the gap between abstract algorithmic logic and physical logic gate connections.

UNIT – 13

STATE MACHINES

NOTES

UNIT STRUCTURE

- 11.1. Introduction
- 11.2. Objective
- 13.3. Basics of State Machines
 - 13.3.1. Mealy and Moore Machines
- 13.4. State Transition Diagrams and Tables
- 13.5. Design and Analysis of Sequential Systems
 - 13.5.1. Sequence Generators
 - 13.5.2. Sequence Detectors
- 13.6. Summary
- 13.7. Glossary
- 13.8. Answers to check your progress
- 13.9. Suggested readings - OER
- 13.10. Reference
- 13.11. Exercise

13.1. INTRODUCTION

We have reached the culmination of Block 2. In Unit 4, you learned how a single flip-flop stores a bit of data. In Unit 5, we connected those flip-flops to create shift registers that move data and counters that track events. Now, we are going to combine everything you have learned to build the "brain" of digital systems: the State Machine.

Think about a vending machine. If a snack costs 15 rupees, the machine doesn't just need to know what coin you just inserted; it needs to remember the total amount you have inserted so far, what buttons you have pressed, and whether it should dispense the snack or give you change. This requires a system that can make complex decisions based on both its current inputs and its past history (its "state").

In digital electronics, we call this a Finite State Machine (FSM). An FSM is a mathematically defined behavioral model composed of a finite number of states, transitions between those states, and specific actions. Inside modern microprocessors, network routers, and even household appliances, you will find state machines acting as the central

NOTES

controllers, orchestrating when data moves, when memory is read, and when operations execute.

In this unit, we will explore the two primary types of state machines: the Mealy machine and the Moore machine. We will learn how to graphically design them using state transition diagrams, and we will look at practical applications, such as sequence detectors that scan data streams for specific binary patterns. Let’s dive into the logic of decision-making!

13.2. OBJECTIVES

After successfully completing this unit, you will be able to:

- Define the concept of a Finite State Machine (FSM) in digital logic.
- Differentiate between Mealy and Moore state machine architectures.
- Translate textual system requirements into State Transition Diagrams and Tables.
- Analyze the operational behavior of sequence generators and sequence detectors.
- Design basic sequential logic systems for real-world applications.

13.3. BASICS OF STATE MACHINES

A Finite State Machine (FSM) is a sequential logic circuit whose outputs and next state are determined by its current state and its current inputs. Because we are dealing with a finite amount of hardware (a specific number of flip-flops), the machine can only ever exist in a finite, limited number of states.

13.3.1. Mealy and Moore Machines

State machines are broadly classified into two categories based on how their outputs are generated.

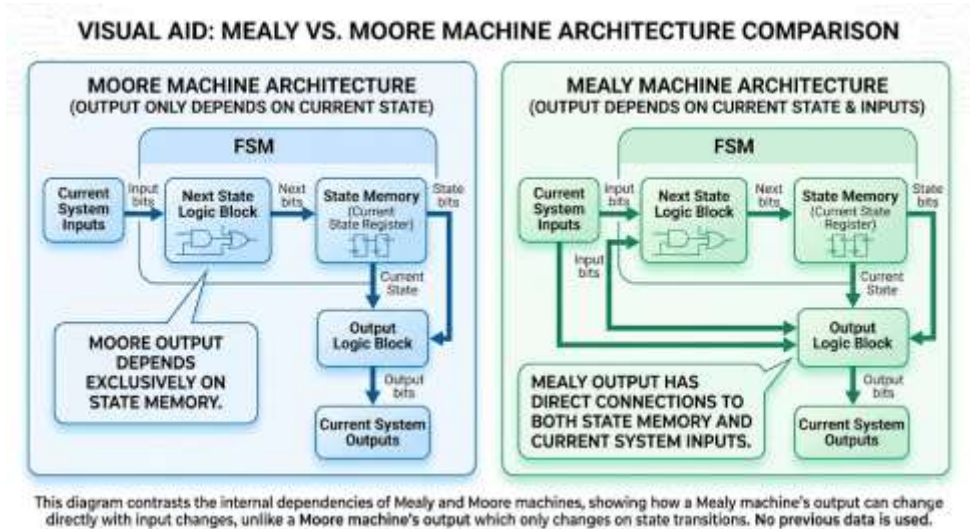


Figure 13.1: Mealy vs. Moore Machine Architecture.

1. The Moore Machine:

In a Moore machine, the output depends **only on the present state** of the flip-flops. The external inputs dictate how the machine transitions from one state to another, but they do not directly connect to the output logic. Because the output is tied strictly to the state, Moore machines are inherently safer in designs where you want to avoid momentary glitches caused by fluctuating input signals. However, because you have to wait for a clock cycle to change the state before the output changes, Moore machines can sometimes require more states and act slightly slower.

2. The Mealy Machine:

In a Mealy machine, the output depends on **both the present state AND the present input**. This means that if an external input changes, the output can change immediately, without waiting for the next clock edge to change the internal state. Mealy machines are typically more efficient and require fewer states (and fewer flip-flops) than Moore machines to perform the same task. The trade-off is that they are highly susceptible to "glitches"—if the input signal is noisy or unstable, the output will instantly reflect that noise.

Feature	Moore Machine	Mealy Machine
Output Dependence	Only on the Present State	On Present State AND Present Input
Number of States	Usually requires more states	Usually requires fewer states
Output Change	Synchronized with the clock	Can change asynchronously with input
Hardware Complexity	Simpler output logic	More complex output logic
Susceptibility to Noise	Low (glitch-free outputs)	High (input noise can glitch the output)

Table 13.1 : Comparison of Mealy and Moore Machines.

13.4. STATE TRANSITION DIAGRAMS AND TABLES

To design a state machine, we cannot simply start wiring logic gates together. We must first model its behavior visually and mathematically.

- **State Transition Diagrams:** As introduced in Unit 5, a state diagram is a graphical map of the system's behavior.
 - **States** are represented by circles.
 - **Transitions** between states are represented by directed arrows.
 - In a **Moore Machine**, the output is written *inside* the circle alongside the state name (e.g., State A / Output 1), because the output belongs to the state. The arrows are labeled only with the input value that causes the transition.

NOTES

➤ In a **Mealy Machine**, the output is written *on the transition arrow* alongside the input (e.g., Input 1 / Output 0), because the output is a result of the specific input being applied during that specific state.

- **State Transition Tables:** Once the diagram is drawn, we translate it into a State Transition Table. This table meticulously lists every possible Present State, every possible Input, the resulting Next State, and the resulting Output. This table serves as the definitive truth table required to calculate the Boolean algebra expressions for the flip-flop inputs (like J and K or D) and the final output logic gates. Without this mathematical translation step, bridging the gap between a conceptual diagram and a physical circuit would be nearly impossible.

13.5. DESIGN AND ANALYSIS OF SEQUENTIAL SYSTEMS

Once we have mastered state diagrams and tables, we can apply FSM theory to design specialized sequential systems. Two of the most common applications are sequence generators and sequence detectors.

13.5.1. Sequence Generators

A sequence generator is a digital circuit designed to produce a specific, predetermined sequence of bits or states indefinitely, synchronized with a clock signal. You can think of a standard binary counter (which counts \$00, 01, 10, 11\$) as a basic sequence generator. However, FSMs allow us to generate arbitrary, non-sequential patterns, such as \$1-4-2-7-5\$. These circuits are widely used in digital communications to generate encryption keys, test patterns, or control signals for stepping motors, where the specific pattern dictates the motor's physical movement.

13.5.2. Sequence Detectors

A sequence detector does the exact opposite: it monitors a continuous, incoming stream of serial data and triggers an output (usually turning a \$0\$ into a \$1\$) only when it recognizes a specific predetermined pattern. For example, a "101 Sequence Detector" will scan an incoming binary stream and flash a HIGH output the moment it reads a \$1\$, followed by a \$0\$, followed by a \$1\$.

Sequence detectors can be designed as **overlapping** or **non-overlapping**.

- In an *overlapping* detector looking for "101", the stream 10101 would trigger the output twice (the last '1' of the first sequence acts as the first '1' of the second sequence).
- In a *non-overlapping* detector, that same stream would only trigger the output once, as the machine resets completely after a successful detection.

Diagram Idea for SLM: A state diagram for a Moore-type "101" sequence detector, showing 4 states: Reset, Got '1', Got '10', and Got '101' (where the output inside the last circle is 1).

NOTES

**Activity 13.1. The Digital Turnstile**

Let's model a real-world Finite State Machine! Imagine a subway turnstile. It has two possible states: LOCKED and UNLOCKED. It accepts two inputs: inserting a COIN or PUSHING the arm.

Your Task : Grab a piece of paper and map out the State Diagram for this turnstile based on these logical rules:

1. Draw two circles: one labeled "LOCKED (State 0)" and one labeled "UNLOCKED (State 1)".
2. If the turnstile is LOCKED and someone PUSHES, it stays LOCKED. (Draw an arrow from LOCKED back to itself, labeled "Push").
3. If the turnstile is LOCKED and someone inserts a COIN, it transitions to UNLOCKED. (Draw an arrow from LOCKED to UNLOCKED, labeled "Coin").
4. If the turnstile is UNLOCKED and someone inserts another COIN, it stays UNLOCKED. (Draw an arrow from UNLOCKED back to itself, labeled "Coin").
5. If the turnstile is UNLOCKED and someone PUSHES through, it transitions back to LOCKED. (Draw an arrow from UNLOCKED to LOCKED, labeled "Push").

Self-Reflection : Is this an example of a Mealy or a Moore machine? (Hint: Does the physical state of the turnstile itself represent the output, or does a combination of the state and your immediate action trigger an output?)

**Check Your Progress 13.1.**

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

Fill in the blanks/Answer in one sentence:

1. A _____ State Machine is a sequential logic circuit with a limited number of possible states.

.....

NOTES

2. In a _____ machine, the output is determined strictly by the present state of the flip-flops.

.....
.
.....
.
.....
.

3. In a Mealy machine state diagram, the output is written on the _____, not inside the state circle.

.....
.
.....
.
.....
.

4. A circuit designed to output a continuous, predetermined pattern of binary numbers is called a sequence _____.

.....
.
.....
.
.....
.

5. 5. An overlapping sequence detector searching for the pattern "11" in the data stream 01110 will detect the pattern _____ times.

.....
.
.....
.
.....
.

13.6. SUMMARY

In this unit, we explored the concept of Finite State Machines (FSMs), the foundational architecture for decision-making in digital electronics. You learned that an FSM is a sequential circuit that uses memory elements (flip-flops) to store a "state," which represents the history of past inputs. We differentiated between the two primary architectures: the Moore machine, where the output is safely tied only to the current internal state, and the Mealy machine, where the output reacts instantly to both the current state and external inputs, requiring fewer states but increasing the risk of signal glitches.

We also discussed the critical design tools used by digital engineers. State Transition Diagrams allow us to visually map out complex behaviors using circles and arrows, which are then translated into State Transition Tables. These tables are the mathematical bridge necessary to calculate the logic gates that will drive the physical circuit.

Finally, we looked at practical implementations by analyzing sequence generators and sequence detectors. Sequence generators are used to output specific control patterns, while sequence detectors act as digital scanners, monitoring incoming serial data streams for overlapping or non-overlapping target patterns. By mastering state machines, you have now grasped the core principles behind how microprocessors, memory controllers, and automated systems are designed. This concludes Block 2, setting a robust foundation as we move toward Memory Systems in Block 3.

13.7. GLOSSARY

- **Finite State Machine (FSM)** : A mathematical model of computation and a sequential logic circuit that can exist in exactly one of a finite number of states at any given time.
- **Mealy Machine** : A type of state machine where the output is determined by both the present state of the system and the current external inputs.
- **Moore Machine** : A type of state machine where the output is determined solely by the present state of the system, independent of immediate input changes.
- **State Transition Diagram** : A visual representation of an FSM using circles (states) and directed arrows (transitions) to show the behavioral flow of the circuit.
- **State Transition Table** : A tabular representation of a state diagram that lists present states, inputs, next states, and outputs, used to design the physical logic circuit.

NOTES

NOTES

- **Sequence Generator** : A digital circuit designed to continuously produce a specific, predetermined sequence of bits or states.
- **Sequence Detector** : A digital FSM designed to monitor a serial input data stream and trigger an output only when a specific, predetermined binary pattern is recognized.

13.8. ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 13.1.

1. Finite
2. Moore
3. transition arrows
4. generator
5. two (The first two 1s make the first detection, and the second and third 1s make the second detection).

12.8. SUGGESTED READINGS – OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectonline.com/User/UnitUrls?unitId=3837	

13.10. REFERENCE

1. Mano, M. M., & Ciletti, M. D. (2013). *Digital Design: With an Introduction to the Verilog HDL* (5th ed.). Pearson Education.
2. Floyd, T. L. (2015). *Digital Fundamentals* (11th ed.). Pearson.
3. Tocci, R. J., Widmer, N. S., & Moss, G. L. (2017). *Digital Systems: Principles and Applications* (12th ed.). Pearson.
4. Kumar, A. A. (2016). *Fundamentals of Digital Circuits* (4th ed.). PHI Learning Pvt. Ltd.

13.11. EXERCISE

PART 1

A. Multiple Choice Questions

1. In a Finite State Machine, the "State" is physically stored using which digital components?
 - (a) Logic Gates
 - (b) Multiplexers
 - (c) Flip-Flops

- (d) Decoders
2. Which of the following is the defining characteristic of a Moore machine?
 - (a) The output depends on the present state and present inputs.
 - (b) The output depends only on the present state.
 - (c) It requires fewer logic states than a Mealy machine.
 - (d) Its output can change asynchronously without a clock pulse.
 3. In a state transition diagram for a Mealy machine, where is the output value typically written?
 - (a) Inside the state circle
 - (b) On the transition arrow alongside the input
 - (c) Outside the diagram as a separate equation
 - (d) In a separate truth table only
 4. A digital circuit designed to monitor a serial data stream and flash a HIGH signal when it reads the pattern "1011" is called a :
 - (a) Sequence Generator
 - (b) Ripple Counter
 - (c) Sequence Detector
 - (d) Multiplexer
 5. What is the main disadvantage of a Mealy machine compared to a Moore machine?
 - (a) It requires more flip-flops.
 - (b) It is slower to respond to inputs.
 - (c) Its outputs are susceptible to immediate glitches from noisy inputs.
 - (d) It cannot be mapped using a state diagram.

Answers with Explanations :

1. **Answer : (c)**

Explanation : Flip-flops are the foundational memory elements in sequential circuits. Their current stored values (e.g., $Q_1=0$, $Q_0=1$) physically represent the system's "state."

2. **Answer: (b)**

Explanation : By definition, a Moore machine's output logic is isolated from the external inputs and relies entirely on the current state of the memory elements.

3. **Answer : (b)**

SEQUENTIAL LOGIC AND STATE MACHINES

NOTES

Explanation : Because a Mealy machine's output depends on both the state and the input, the output is documented on the transition arrow (e.g., Input/Output) to show what happens during that specific change.

4. **Answer : (c)**

Explanation : Sequence detectors are specific FSMs designed to scan an incoming stream and trigger an output upon finding a predefined pattern.

5. **Answer: (c)**

Explanation : Because the external inputs bypass the flip-flops and connect directly to the output logic in a Mealy machine, any noise or bounce on the input line instantly affects the output.

B. Short Answer Type Questions (100 words)

1. Briefly explain the primary difference between a sequence generator and a sequence detector.
2. Why is it impossible to build a State Machine using only combinational logic gates (like AND/OR gates)?
3. What is the purpose of converting a State Transition Diagram into a State Transition Table?

PART 2

C. Long Answer Type Questions (200 words)

1. Compare and contrast the Mealy and Moore machine architectures. Discuss their definitions, their relative hardware complexity (number of states typically required), and how they handle input noise.
2. Explain the difference between an "overlapping" and a "non-overlapping" sequence detector using the binary input stream 1111 and the target pattern 11. How many times would each type of detector trigger a successful output?
3. Describe the general step-by-step process a digital design engineer must follow to design a custom Finite State Machine, starting from a word-problem description to the final logic circuit diagram.

PART 3

D. Detailed Answer Type Questions (300 words)

1. Design a Mealy machine for sequence 101 and compare it with Moore implementation.
2. How does overlapping detection affect state transitions?
3. Optimize the FSM to reduce the number of states (if possible).



DR. C.V. RAMAN UNIVERSITY

Chhattisgarh, Bilaspur AN AISECT GROUP UNIVERSITY

Approved by: PCI | AICTE | NCTE | BCI | Member of: AIU | Joint Committee: JUGC | SER | AICTE | Recognized by: UGC | A NAAC Accredited University



DR. C. V. RAMAN UNIVERSITY (C.G.)

SLM FEEDBACK FORM

PROGRAMME : _____

COURSE : _____

BLOCK : _____

UNIT : _____

Dear Student,

While studying the units of this block, you may have found certain portions of the text difficult to comprehend. We wish to know your difficulties and suggestions, in order to improve the course. Therefore, we request you to fill up and send us the following questionnaire, which pertains to this block. If you find the space provided insufficient, kindly use a separate sheet.

Please mail to aisectcontent@aisect.org / for online submission please scan QR code:

Questionnaire

1. Name : _____

2. Enrolment No. :

--	--	--	--	--	--	--	--	--	--	--

3. How many hours did you need for studying the units? :

--

4. Please give your reactions to the following items based on your reading of the block:

Items	Excellent	Very Good	Good	Poor	Give specific Examples (If poor)
Presentation Quality					
Language and Style					
Illustrations					
Used (diagrams, tables, etc.)					
Conceptual Clarity					
Check Your Progress Questions					
Suggested Readings - OER					

5. Any other comments:

Block IV	MEMORY ARCHITECTURE AND CONCEPTS	153 – 182
Unit 14	Auxiliary Memory	155
Unit 15	Semiconductor Memory	165
Unit 16	Advanced Memory Concepts	175



This section transitions our focus from basic logic components to massive storage *systems*. It addresses the critical hierarchy that governs how computers manage data across vastly different speed and capacity domains. We begin by examining the standard **Memory Hierarchy** pyramid, which dictates the strict balance between access time, cost per bit, and total storage capacity. Following this, we explore internal Semiconductor Memory technologies. You will analyze the physical operating mechanisms of Static RAM (SRAM, used in high-speed CPU caches), Dynamic RAM (DRAM, used in massive main system memory), and non-volatile standard ROM and modern Flash storage. Because main memory must be incredibly fast, it is always **volatile**, requiring constant electricity to retain its data, unlike non-volatile permanent storage. In contrast, the second half of this block explores Auxiliary Memory, the robust magnetic and optical technologies used for permanent, massive data storage, such as Hard Disk Drives (HDDs), Solid State Drives (SSDs), and Magnetic Tapes. Understanding how data is transferred between these layers via complex timing protocols is Prerequisites to designing efficient and scalable computing machines.

UNIT – 14

AUXILIARY MEMORY

NOTES

UNIT STRUCTURE

- 14.1. Introduction
- 14.2. Objective
- 14.3. Overview of Auxiliary Memory
 - 14.3.1. Magnetic Drum, Disk, and Tape
- 14.4. Characteristics of Auxiliary Memory
 - 14.4.1. Access Time
 - 14.4.2. Storage Capacity
- 14.5. Use Cases of Auxiliary Memory
- 14.6. Summary
- 14.7. Glossary
- 14.8. Answers to check your progress
- 14.9. Suggested readings - OER
- 14.10. Reference
- 14.11. Exercise

14.1. INTRODUCTION

Welcome to Block 3! In our previous blocks, we built the foundational logic of digital systems. You learned how flip-flops store individual bits and how state machines use those bits to make rapid decisions. However, the memory we have discussed so far—like the registers inside a CPU—is inherently temporary. When the power is turned off, all that carefully stored data vanishes instantly. This type of temporary, fast memory is known as "volatile" memory.

But what happens when you want to save a photograph, a document, or an entire operating system so that it is still there when you turn your computer on tomorrow? This is where Auxiliary Memory steps in. Also known as secondary storage or mass storage, auxiliary memory is the long-term vault of a computer system. It is specifically designed to store vast amounts of data permanently (non-volatile storage), safely preserving your information even when the power is completely disconnected.

In this unit, we are going to step away from the microscopic logic gates and look at the macroscopic hardware that handles mass storage. We will explore the classic magnetic storage technologies that revolutionized computing: magnetic tapes, drums, and disks. You will

NOTES

learn the physics and mechanics of how a physical medium can hold binary data using magnetism. We will also dive into the technical metrics that define a good storage device, breaking down what exactly makes one drive "faster" or "larger" than another by analyzing access times and storage capacities. Let us transition from the world of rapid calculation into the world of permanent data preservation!

14.2. OBJECTIVES

After successfully completing this unit, you will be able to:

- Define the role and necessity of auxiliary memory in a computer system.
- Differentiate between the operational mechanisms of magnetic drums, disks, and tapes.
- Analyze the key characteristics of storage devices, specifically access time and storage capacity.
- Calculate or conceptualize components of access time, such as seek time and rotational latency.
- Identify real-world use cases and appropriate applications for various auxiliary storage systems.

14.3. OVERVIEW OF AUXILIARY MEMORY

Auxiliary memory forms the bottom tier of the computer memory hierarchy. Unlike the main memory (RAM), which is directly accessible by the CPU, auxiliary memory is accessed indirectly via input/output controllers. Because its primary job is massive, permanent storage rather than immediate processing speed, it is significantly cheaper per megabyte than main memory.

14.3.1. Magnetic Drum, Disk, and Tape

Historically and practically, magnetism has been the most reliable way to store large volumes of binary data. By polarizing tiny metallic PARTICLES on a surface to point either "North" or "South," we can represent binary 1s and 0s.

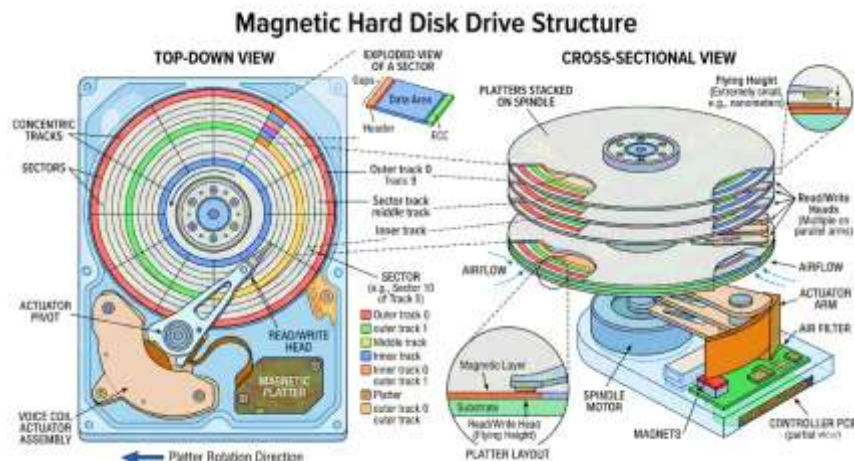


Figure 14.1 : Magnetic Hard Disk Drive Structure.

NOTES

- **Magnetic Tape:** Think of the classic cassette tape. Magnetic tape is a long strip of plastic coated with a magnetic material, wrapped around spools. Its defining feature is **sequential access**. To read a piece of data stored in the middle of the tape, the drive must physically wind past all the preceding data. While this makes it incredibly slow for everyday tasks, tapes can hold massive amounts of data very cheaply, making them perfect for long-term archiving.
- **Magnetic Drum:** One of the earliest forms of computer memory, the magnetic drum was a large metal cylinder coated with magnetic material. Stationary read/write heads were positioned along the length of the drum. As the drum spun continuously, the heads would read or write data on circular tracks. Though obsolete today, drums paved the way for modern disk technology.
- **Magnetic Disk:** This is the foundation of the Hard Disk Drive (HDD). A magnetic disk consists of one or more flat, circular platters made of glass or aluminum, coated with a magnetic film. The platters spin at high speeds, while a movable mechanical arm (the read/write head) floats fractions of a millimeter above the surface. Unlike tape, disks offer **direct (or random) access**. The arm can jump directly to the specific track where the data is stored, making it vastly faster than sequential tape.

Comparison of Data Storage Mediums: Access Methods

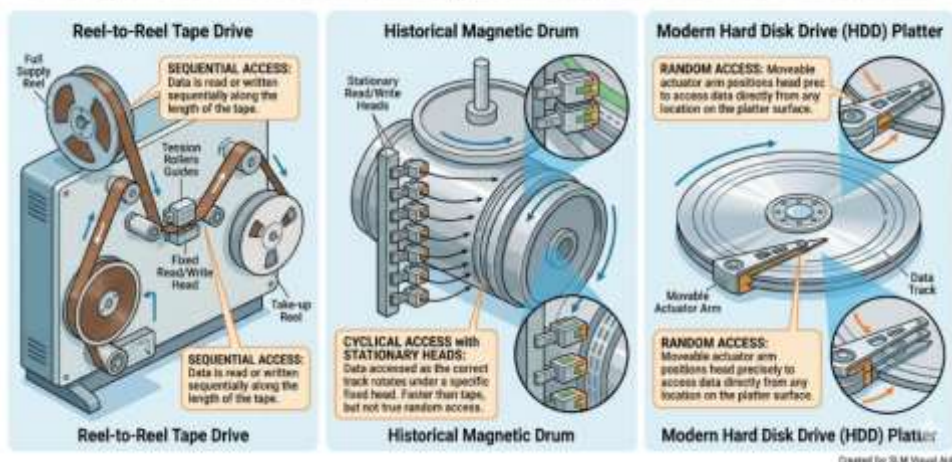


Figure 14.2 : A visual comparison of the three mediums.

14.4. CHARACTERISTICS OF AUXILIARY MEMORY

When evaluating or designing a computer system, engineers must judge auxiliary memory based on two primary performance metrics: how fast it can retrieve data, and how much data it can hold.

14.4.1. Access Time

Access time is the total time it takes for the computer to request data from the storage device and for the device to successfully deliver that data. For a magnetic disk, access time is not a single number; it is a combination of three distinct mechanical delays:

NOTES

- 1. **Seek Time:** The time it takes for the mechanical read/write arm to move to the correct circular track on the disk. This is usually the longest delay.
- 2. **Rotational Latency:** Once the arm is on the correct track, it must wait for the spinning platter to rotate the exact sector containing the data directly under the head.
- 3. **Data Transfer Time:** The time it takes to actually read the magnetic signals and push the binary data through the cables to the computer's main memory.

Total Access Time
= Seek Time + Rotational Latency + Data Transfer Time

14.4.2. Storage Capacity

Storage capacity defines the maximum amount of binary data a device can hold, usually measured in Gigabytes (GB) or Terabytes (TB). In magnetic disks, capacity is determined by a few physical factors :

- **Recording Density:** How many bits can be squeezed into a single inch of a track (Bits Per Inch - BPI).
- **Track Density:** How many concentric circular tracks can be squeezed onto the radius of the platter (Tracks Per Inch - TPI).
- **Number of Platters:** Large hard drives stack multiple platters on top of each other, using both the top and bottom surfaces of each platter to double or quadruple the total capacity.

Characteristic	Magnetic Tape	Magnetic Disk (HDD)
Access Method	Sequential Access	Direct / Random Access
Access Time	Very Slow (Seconds to Minutes)	Fast (Milliseconds)
Storage Capacity	Extremely High (Archival)	High (Everyday computing)
Cost per Gigabyte	Lowest	Low

Table 14.1: Key Characteristics Comparison

14.5. USE CASES OF AUXILIARY MEMORY

Because different auxiliary memory types excel at different tasks, they are used strategically within a computer's architecture.

- **Mass Storage for Operating Systems:** Magnetic disks (and their modern solid-state equivalents) are used to store the computer's Operating System, installed software, and user files. Their fast direct-access speeds allow you to boot up your computer and open applications without noticeable delays.
- **Virtual Memory:** Sometimes, a computer tries to run an application that is too large to fit entirely in the fast, expensive RAM. The operating system will use a portion of the auxiliary magnetic

disk as "Virtual Memory," temporarily swapping chunks of data between the RAM and the disk to keep the computer running smoothly.

Data Archiving and Disaster Recovery: For enterprise data centers, hospitals, and banks, securely storing decades of records is a legal requirement. Magnetic tape remains the industry standard for this use case. A tape cartridge can hold dozens of Terabytes of data, uses zero electricity while sitting on a shelf, and is highly resistant to ransomware attacks since it is physically disconnected from the network (air-gapped).



Activity 14.1. The IT Storage Consultant

Imagine you are an IT consultant hired to upgrade the storage infrastructure for a large film production studio. The studio has two distinct needs:

1. **The Editing Bay :** Video editors need to quickly jump back and forth across massive 4K video files while editing a movie. Delay is unacceptable.
2. **The Deep Archive :** The studio needs to permanently store all the raw, unedited footage from movies filmed 10 years ago. This data is rarely accessed but must never be lost.

Your Task : Write a brief, 3-4 sentence proposal recommending the specific type of auxiliary memory (Magnetic Tape vs. Magnetic Disk) for each need. Provide a technical justification for your choices based on the concepts of "Access Time" and "Sequential vs. Direct Access" learned in this unit.



Check Your Progress 14.1.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. **True or False :** Auxiliary memory is considered "volatile" because it loses all its data when the computer's power is turned off.

.....

2. Fill in the blank: The physical movement of the read/write arm to the correct circular track on a magnetic disk is called the _____ time.

.....

NOTES

3. Match the Following:

Column A (Technology)	Column B (Access Method)
A. Magnetic Tape	1. Direct / Random Access
B. Magnetic Disk	2. Sequential Access

4. Multiple Choice : Which of the following components represents the time a disk head must wait for the correct data sector to spin underneath it?

(a) Seek Time	(b) Rotational Latency
(c) Transfer Rate	(d) Propagation Delay

5. Fill in the blank :
When a computer's main RAM is full, the operating system can use a section of the auxiliary disk as _____ memory to keep applications running.

14.6. SUMMARY

In this unit, we explored the critical role of auxiliary memory in computer architecture. You learned that while main memory (RAM) is fast, it is volatile and expensive. Auxiliary memory solves this by providing massive, non-volatile, and cost-effective storage, acting as the permanent vault for operating systems, applications, and user data.

We examined the historical and practical technologies that make mass storage possible, focusing heavily on magnetic mediums. You discovered that magnetic tape relies on sequential access, making it highly suitable for cheap, long-term archiving, but incredibly slow for active computing. In contrast, magnetic disks (HDDs) utilize spinning platters and a movable read/write arm to provide direct access, allowing the system to retrieve data from any location on the disk in a matter of milliseconds.

Furthermore, we broke down the engineering metrics of storage performance. You now understand that a disk's access time is not a single variable, but rather the sum of mechanical seek time, rotational

latency, and electronic data transfer time. We also discussed how physical factors like track density and recording density dictate a drive's total storage capacity. By understanding these auxiliary storage systems, you are now prepared to explore faster, semiconductor-based memory solutions in the upcoming units.

14.7 GLOSSARY

- **Auxiliary Memory:** Non-volatile, secondary storage used for long-term data retention (e.g., hard drives, magnetic tapes).
- **Volatile Memory:** Memory that requires continuous power to maintain its stored data; data is lost when powered off.
- **Non-Volatile Memory:** Memory that retains its stored data even when the device is entirely powered down.
- **Sequential Access:** A data retrieval method where the storage medium must be read in a linear, physical order (like a cassette tape).
- **Direct Access:** A data retrieval method where the read/write mechanism can jump directly to the specific location of the data (like a hard disk).
- **Seek Time:** The mechanical time required for a disk drive's read/write head to position itself over the correct data track.
- **Rotational Latency:** The waiting time for the spinning disk platter to rotate the desired data sector underneath the read/write head.
- **Virtual Memory:** A memory management technique where the operating system uses a portion of auxiliary storage to simulate additional main memory (RAM).

14.8. ANSWERS TO CHECK YOUR PROGRESS

1. False
2. Seek
3. A - 2, B - 1
4. (b) Rotational Latency
5. Virtual

14.9. SUGGESTED READINGS – OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectonline.com/User/UnitUrls?unitId=3838	

14.10 REFERENCE

1. Mano, M. M., & Ciletti, M. D. (2013). *Digital Design: With an Introduction to the Verilog HDL* (5th ed.). Pearson Education.

MEMORY ARCHITECTURE AND CONCEPTS

NOTES

2. Patterson, D. A., & Hennessy, J. L. (2013). *Computer Organization and Design: The Hardware/Software Interface* (5th ed.). Morgan Kaufmann.
3. Stallings, W. (2015). *Computer Organization and Architecture: Designing for Performance* (10th ed.). Pearson.
4. Floyd, T. L. (2015). *Digital Fundamentals* (11th ed.). Pearson.

14.11. EXERCISE

PART 1

A. Multiple Choice Questions (MCQs)

1. The primary purpose of auxiliary memory in a computer system is to:
 - (a) Perform high-speed arithmetic calculations
 - (b) Provide volatile storage for active CPU instructions
 - (c) Provide permanent, non-volatile mass storage for data
 - (d) Connect the motherboard to peripheral display devices
2. Which of the following storage devices provides data purely through "Sequential Access"?
 - (a) Magnetic Disk
 - (b) Magnetic Tape
 - (c) Solid State Drive
 - (d) Magnetic Drum
3. In a magnetic disk drive, the time taken to move the read/write head to the desired circular track is known as:
 - (a) Data Transfer Time
 - (b) Rotational Latency
 - (c) Propagation Delay
 - (d) Seek Time
4. How is the data storage capacity of a magnetic disk primarily increased by manufacturers?
 - (a) By increasing the recording density and stacking multiple platters
 - (b) By removing the read/write head
 - (c) By switching from direct access to sequential access
 - (d) By increasing the size of the computer's RAM
5. When a computer runs out of primary RAM space, it can temporarily swap data to the auxiliary magnetic disk using a technique called:
 - (a) Cache Memory
 - (b) Virtual Memory

(c) Sequential Buffering

(d) Read-Only Memory

Answers with Explanations:

1. **Answer : (c)**

Explanations : Provide permanent, non-volatile mass storage for data
Explanation: Auxiliary memory (like hard drives) is designed for long-term, permanent storage of files and operating systems, retaining data even without power.

2. **Answer : (b)**

Explanations : Magnetic Tape
Explanation: Magnetic tape is a linear medium. To reach data at the end of the tape, the drive must physically spool past all the preceding data, which is the definition of sequential access.

3. **Answer : (d)**

Explanations : Seek Time
Explanation: Seek time refers specifically to the mechanical movement of the actuator arm placing the head over the correct track.

4. **Answer : (a)**

Explanations : By increasing the recording density and stacking multiple platters
Explanation: Capacity is maximized by packing more bits into every inch of the magnetic surface (density) and by using both sides of multiple stacked platters.

5. **Answer: (b)**

Explanations : Virtual Memory
Explanation: Virtual memory is a memory management feature where the OS utilizes a portion of the secondary storage to act as an extension of the primary RAM.

B. Short Answer Type Questions (100 words)

1. Explain the difference between volatile and non-volatile memory, providing one example of each.
2. Briefly describe the structural difference between a magnetic drum and a modern magnetic disk.
3. Why is magnetic tape still heavily utilized in large enterprise data centers today despite its slow speed?

PART 2

C. Long Answer Type Questions (200 words)

1. Explain the components of access time in a magnetic hard disk: seek time, rotational latency, and transfer time. Which one is the main bottleneck and why?
2. Compare **sequential access** and **direct (random) access** using examples of a **magnetic tape drive** and a **magnetic disk drive**.
3. Explain **virtual memory** and how RAM and disk work together to run large applications efficiently.

NOTES**D. Long Answer Type Questions (300 words)**

1. Explain the architecture of a Moore Machine.
2. Compare the output behavior of a Moore Machine and a Mealy Machine.
3. Design a finite state machine to detect the sequence 101 in a continuous input stream with overlapping allowed.

UNIT – 15

SEMICONDUCTOR MEMORY

NOTES

UNIT STRUCTURE

- 15.1. Introduction**
- 15.2. Objective**
- 15.3. Types of Semiconductor Memory**
 - 15.3.1. RAM and ROM**
 - 15.3.2. DRAM and SRAM**
- 15.4. Memory Hierarchy**
 - 15.4.1. Cache Memory**
 - 15.4.2. Virtual Memory**
- 15.5. Associative and Content-Addressable Memory**
- 15.6. Summary**
- 15.7. Glossary**
- 15.8. Answers to check your progress**
- 15.9. Suggested readings - OER**
- 15.10. Reference**
- 15.11. Exercise**

15.1. INTRODUCTION

Up to this point in our course, we have extensively discussed various combinational and sequential logic circuits. We have seen how flip-flops and registers can store individual bits or small words of data. However, when we talk about practical digital systems—like the smartphone in your hand or the laptop on your desk—we are dealing with millions or even billions of bytes of data. How do we store this massive amount of information efficiently? This is where Semiconductor Memory comes into the picture.

In the early days of computing, memory was magnetic and bulky. Today, semiconductor memory, built using integrated circuits (ICs) based on silicon technology, has completely revolutionized the way digital devices are designed. It is compact, incredibly fast, and consumes very little power. In this unit, we will explore the fascinating world of semiconductor memory. We will begin by classifying the two primary families of memory: Random Access Memory (RAM) and Read-Only Memory (ROM), understanding why a computer fundamentally needs both to operate.

NOTES

As we progress, we will delve into the subdivisions of RAM, specifically Static RAM (SRAM) and Dynamic RAM (DRAM), and see how hardware engineers balance speed and storage capacity. You will also learn about the "Memory Hierarchy," an architectural principle that organizes different types of memory to optimize a computer's overall performance. We will discuss Cache memory, which acts as a high-speed buffer, and Virtual memory, which cleverly tricks the system into thinking it has more physical RAM than it actually does. Finally, we will look at Content-Addressable Memory (CAM), a specialized storage mechanism used in high-speed networking. Let us dive in and uncover the storage secrets of digital systems!

15.2. OBJECTIVES

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

- Categorize the different types of semiconductor memories used in digital systems.
- Differentiate between RAM and ROM, as well as SRAM and DRAM, based on their architecture and applications.
- Understand the concept of memory hierarchy and evaluate the trade-offs between speed, cost, and capacity.
- Explain the functional roles of Cache Memory and Virtual Memory in a computer system.
- Describe the operation and application of Associative or Content-Addressable Memory (CAM).

15.3. TYPES OF SEMICONDUCTOR MEMORY

Semiconductor memory is a digital electronic device used for data storage, implemented on a semiconductor-based integrated circuit. There are many ways to classify memory, but the most fundamental distinction is based on how data is retained and accessed. The two broadest categories are RAM (Random Access Memory) and ROM (Read-Only Memory). Let us break these down.

15.3.1. RAM and ROM

RAM is a read/write memory. The term "Random Access" means that any memory location can be accessed in the same amount of time, regardless of its physical position. However, RAM is volatile; it loses all its stored data the moment the power supply is turned off. It acts as the primary working workspace for the computer's processor. ROM, on the other hand, is non-volatile. The data written into a ROM chip is permanent (or semi-permanent) and remains intact even when the power is removed. ROM is traditionally used to store crucial system instructions, such as the BIOS (Basic Input/Output System) that boots up your computer.

Feature	RAM (Random Access Memory)	ROM (Read-Only Memory)
Data Retention	Volatile (Data lost without power)	Non-Volatile (Data retained without power)
Operations	Read and Write operations	Primarily Read operations
Speed	Very fast	Slower compared to RAM
Usage	Working memory for CPU/OS	Storing firmware and boot instructions

Table 15.1 : Comparison of RAM and ROM

15.3.2. DRAM and SRAM

RAM is further divided into Dynamic RAM (DRAM) and Static RAM (SRAM), based on the internal circuit design used to store a bit of data.

SRAM uses a bistable latching circuitry (made of 4 to 6 transistors) to store each bit. As long as power is supplied, SRAM holds its data without needing to be refreshed. It is extremely fast but takes up more physical space on a chip and is more expensive.

DRAM stores each bit of data in a tiny capacitor within an integrated circuit. Because capacitors slowly leak electrical charge, DRAM must be periodically "refreshed" (read and re-written) thousands of times per second to prevent data loss. Although slower than SRAM, DRAM is much cheaper and allows for a much higher data density, making it the standard choice for main system memory.

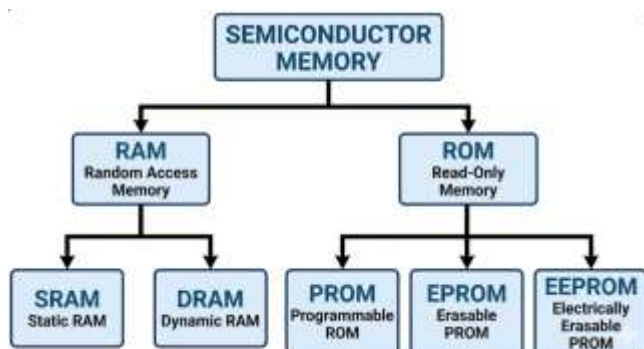


Figure 15.1 : block diagram showing the classification of Semiconductor Memory.

Feature	SRAM (Static RAM)	DRAM (Dynamic RAM)
Storage Element	Flip-flops (Transistors)	Capacitors and Transistors
Refresh Requirement	No refresh required	Requires constant refreshing
Speed	Faster	Slower
Density & Cost	Low density, High cost	High density, Low cost

NOTES

☒ **Check Your Progress 15.1.**

Notes : (a) Space is given below for your answer.
(b) Check your answer with the one given at the end of this unit.

- 1. Why is ROM considered non-volatile memory?
.....
.....
.....
- 2. What is the main reason DRAM requires periodic refreshing?
.....
.....
.....

15.4. MEMORY HIERARCHY

No single type of memory can fulfill all the requirements of a modern computer. A computer needs memory that is blazing fast to keep up with the CPU, but it also needs memory that is cheap and holds massive amounts of data. Since fast memory (like SRAM) is very expensive, and cheap memory (like magnetic disks or flash drives) is very slow, computer architects use a concept called the **Memory Hierarchy**. This hierarchy is structured like a pyramid based on speed, capacity, and cost per bit.

15.4.1. Cache Memory

Cache memory is a small, ultra-fast block of memory positioned directly between the CPU and the main memory (DRAM). It is typically built using expensive SRAM. The CPU operates much faster than main memory. Without cache, the CPU would waste a lot of time waiting for data to arrive. Cache stores copies of the data and instructions from frequently used main memory locations. When the CPU needs data, it first checks the Cache (L1, L2, or L3). If the data is there (a Cache Hit), it is retrieved almost instantly.

15.4.2. Virtual Memory

While Cache speeds up access, Virtual Memory expands capacity. Virtual memory is a memory management technique where the operating system uses a portion of the secondary storage (like a Hard Drive or SSD) to act as an extension of the main RAM. When the physical RAM gets full, the operating system moves inactive pages of data from RAM to the hard drive, freeing up space for active processes. When that inactive data is needed again, it is "swapped" back into RAM. This gives the user the illusion of having a nearly unlimited amount of main memory.

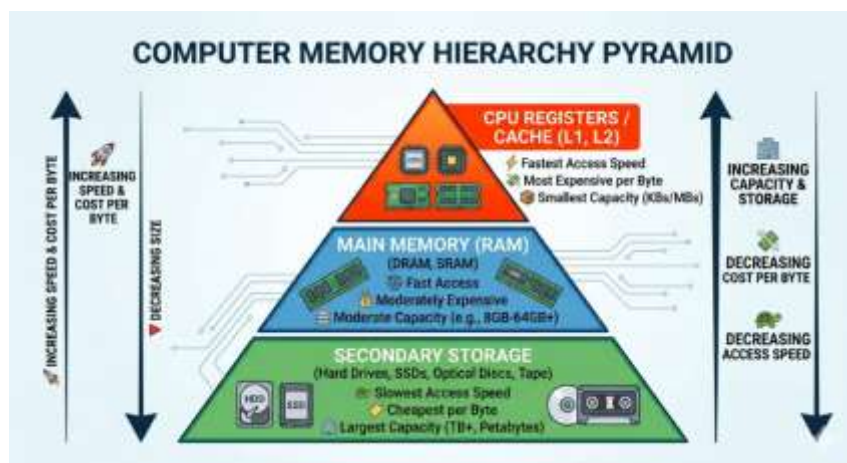


Figure 5.2 : A pyramid diagram illustrating the Memory Hierarchy.

Hierarchy Level	Memory Type	Speed	Capacity	Cost per Bit
Level 0	CPU Registers	Fastest	Smallest (Bytes)	Highest
Level 1	Cache (SRAM)	Very Fast	Small (MBs)	High
Level 2	Main Memory (DRAM)	Moderate	Large (GBs)	Moderate
Level 3	Secondary (SSD/HDD)	Slowest	Largest (TB+)	

Table 15.3 : Levels of Memory Hierarchy.

15.5. ASSOCIATIVE AND CONTENT-ADDRESSABLE MEMORY

In standard RAM architectures, the CPU provides a specific memory "address" (a location number), and the memory controller retrieves the data stored at that exact address. This is efficient for most computing tasks, but what if you need to search a massive database for a specific piece of data, and you don't know its address? In standard RAM, the system would have to search location by location until it finds the match, which consumes valuable time.

This brings us to **Content-Addressable Memory (CAM)**, also known as Associative Memory. Unlike RAM, which returns the data based on a provided address, CAM does the exact opposite: the user supplies the data word, and the CAM searches its entire memory simultaneously (in parallel) to find if that data word exists anywhere within it. If a match is found, the CAM returns the address where that data is located.

Because CAM searches all memory cells at the same time in hardware, it is incredibly fast. However, this parallel searching requires complex circuitry for every single memory cell, making CAM much more expensive, larger in physical size, and more power-hungry than standard RAM. As a result, it is not used for main memory. Instead, it

MEMORY ARCHITECTURE AND CONCEPTS

NOTES

is highly specialized. It is widely used in computer networking devices, such as network routers and switches, where it rapidly searches routing tables to find MAC addresses or IP addresses to forward data packets without delay.

Table 15.4 : Standard RAM vs. Content-Addressable Memory (CAM)

Feature	Standard RAM	CAM (Associative Memory)
Input Provided	Memory Address	Data Word (Content)
Output Returned	Data Word	Memory Address of the Match
Search Mechanism	Sequential (Location by Location)	Parallel (Simultaneous search)
Primary Use Case	Main system memory	High-speed network routers

Check Your Progress 15.2.

Notes : (a) Space is given below for your answer.
(b) Check your answer with the one given at the end of this unit.

1. How does Cache memory improve the overall processing speed of a computer?
.....
.....
.....
2. What input does a user provide to a Content-Addressable Memory (CAM)?
.....
.....
.....

Activity 15.1.

Turn on your personal computer or laptop. Navigate to the Task Manager (on Windows, press Ctrl+Shift+Esc) or Activity Monitor (on Mac). Go to the "Performance" tab and click on "Memory".
Observe the total amount of RAM your system has. Next, switch to the "CPU" tab. Look at the information listed under the graphs. Can you identify the sizes of the L1, L2, and L3 Cache memory on your processor? Write down these specifications and compare the size of your RAM (in GBs) versus your Cache (in MBs). This practical observation highlights the cost/capacity trade-off in the Memory Hierarchy!

15.6. SUMMARY

In this unit, we explored the critical role of semiconductor memory in digital electronics. We started by drawing a clear line between the two main memory families: Random Access Memory (RAM), which is volatile and serves as the computer's high-speed workspace, and Read-Only Memory (ROM), which is non-volatile and retains essential boot-up instructions. We then broke down RAM into Static RAM (SRAM) and Dynamic RAM (DRAM), learning that while SRAM uses flip-flops for faster, refresh-free access, DRAM relies on high-density capacitors, making it the cost-effective choice for main memory despite its need for constant refreshing.

Furthermore, we examined the Memory Hierarchy, a structural pyramid that balances the trade-offs between speed, capacity, and cost. We discovered how Cache memory (built from SRAM) acts as a high-speed bridge preventing the CPU from idling, while Virtual memory borrows space from the hard drive to seamlessly expand the apparent capacity of main RAM. Finally, we looked into Content-Addressable Memory (CAM), a specialized associative memory that flips the standard memory model on its head by searching via data content rather than an address, proving invaluable for high-speed network routing. Understanding these memory types and architectures is essential for any modern digital system design.

15.7. GLOSSARY

- **Volatile Memory** : Computer memory that requires power to maintain the stored information.
- **Non-Volatile Memory** : Computer memory that retains stored information even when unpowered.
- **Refresh** : The process of periodically reading and rewriting data in DRAM to prevent the capacitors from discharging and losing data.
- **Cache Hit** : A state where the data requested by the CPU is successfully found in the cache memory.
- **Page/Swapping** : The process of moving blocks of data between physical RAM and secondary storage in virtual memory management.

15.8. ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 15.1

1. ROM is considered non-volatile because the data programmed into it is retained permanently, even when the power supply to the device is turned off.
2. DRAM stores data in tiny capacitors that naturally leak electrical charge over time. It requires periodic refreshing to restore this charge and prevent the loss of data bits.

NOTES

NOTES

Check Your Progress 15.2

1. Cache memory stores frequently accessed data and instructions close to the CPU. Since it is much faster than main memory, it drastically reduces the time the CPU spends waiting for data to fetch.
2. In CAM, the user provides the actual "Data Word" (or content) as the input, and the memory returns the address location where that specific data is stored.

15.9. SUGGESTED READINGS – OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectonline.com/User/UnitUrls?unitId=3839	

15.10. REFERENCE

1. Mano, M. M. (2017). *Digital Logic and Computer Design*. Pearson Education.
2. Floyd, T. L. (2015). *Digital Fundamentals*. Pearson.
3. Stallings, W. (2016). *Computer Organization and Architecture*. Pearson.

15.11. EXERCISE

PART – 1

A. Multiple Choice Questions

1. Which of the following memories must be refreshed repeatedly to retain its data?
 - (a) ROM
 - (b) SRAM
 - (c) DRAM
 - (d) PROM
2. In the memory hierarchy, which memory type is the fastest but has the smallest capacity?
 - (a) Cache Memory
 - (b) CPU Registers
 - (c) Main Memory
 - (d) Magnetic Disk
3. What does CAM stand for in the context of digital memory?
 - (a) Computer-Aided Memory
 - (b) Content-Addressable Memory

- (c) Cache Allocated Memory
- (d) Core Access Memory
- 4. The technique of utilizing hard disk space as an extension of main memory is called:
 - (a) Cache Memory
 - (b) Flash Memory
 - (c) Virtual Memory
 - (d) Associative Memory
- 5. Which storage element is primarily used to construct Static RAM (SRAM)?
 - (a) Capacitors
 - (b) Magnetic cores
 - (c) Flip-flops (Transistors)
 - (d) Diodes

Answers with explanations:**1. Answer: (c)**

Explanation: DRAM (Dynamic Random Access Memory) stores data in capacitors. Because capacitors lose their electrical charge over time, the memory must be refreshed (recharged) periodically to maintain the integrity of the stored data.

2. Answer: (b)

Explanation: CPU Registers are located inside the processor itself. While they provide the fastest data access speeds in the entire memory hierarchy, they are very expensive and take up significant physical space on the chip, resulting in the smallest storage capacity.

3. Answer: (b)

Explanation: CAM stands for Content-Addressable Memory. Unlike standard RAM, which finds data based on an address, CAM compares input data against a table of stored data and returns the address where the matching data is found.

4. Answer: (c)

Explanation: Virtual Memory is a memory management technique that uses secondary storage (like a hard drive) to act as an overflow for the physical RAM. This allows the computer to run larger applications by "swapping" data between the disk and the RAM.

5. Answer: (c)

Explanation: Static RAM (SRAM) is constructed using flip-flops (circuits typically made of 4 to 6 transistors). Unlike DRAM, which uses capacitors, flip-flops hold their state indefinitely as long as power is supplied, eliminating the need for a refresh cycle.

NOTES

B. Short Answer Questions

1. Explain the fundamental difference between volatile and non-volatile memory with one example of each.
2. Briefly describe how a "Cache Hit" differs from a "Cache Miss."

PART – 2

C. Long Answer Questions: (200 words)

1. Compare and contrast SRAM and DRAM in terms of their internal circuitry, speed, physical density, and primary applications in modern computers.
2. Explain the concept of Virtual Memory. How does the operating system manage the transfer of data when physical RAM is full, and what is the benefit to the end-user?

PART – 3

D. Long Answer Questions: (300 words)

1. Explain the concept of Memory Hierarchy with a neat diagram (or pyramid structure).
2. Describe the working principle of Content-Addressable Memory.
3. Discuss the advantages of Content-Addressable Memory in network routers.
4. Also, explain its limitations and why it is not suitable as standard main memory.

UNIT – 16

ADVANCED MEMORY CONCEPTS

NOTES

UNIT STRUCTURE

- 16.1. Introduction**
- 16.2. Objective**
- 16.3. Virtual Memory**
 - 16.3.1. Address Space and Memory Space**
- 16.4. Address Mapping Techniques**
 - 16.4.1. Page Table and Page Replacement**
- 16.5. Cache Memory**
 - 16.5.1. Hit Ratio and Mapping Techniques**
- 16.6. Writing into Cache Memory**
 - 16.6.1. Methods and Algorithms**
- 16.7. Summary**
- 16.8. Glossary**
- 16.9. Answers to check your progress**
- 16.10. Suggested readings - OER**
- 16.11. Reference**
- 16.12. Exercise**

16.1. INTRODUCTION

Have you ever wondered how your computer manages to run a massive high-definition video editing software or a complex modern video game even when the actual physical RAM (Random Access Memory) is much smaller than the software's requirements? Or why some applications seem to "load" instantly while others take time? The secret lies in the advanced memory management concepts we are about to explore in this unit. In our previous units, we looked at the physical components of memory, such as SRAM and DRAM. Now, we are going to look at the "intelligence" behind how these components are used.

In this unit, we dive deep into the concepts of Virtual Memory and Cache Memory. Virtual Memory is a brilliant architectural trick that allows a system to behave as if it has a much larger memory than it physically possesses. It acts as a bridge between the CPU's logical requirements and the physical reality of the hardware. We will discuss how "Address

NOTES

Space" differs from "Memory Space" and how the system keeps track of data using Page Tables.

Furthermore, we will expand our understanding of Cache Memory. While we know it is fast, we haven't yet discussed the logic used to decide which data stays in the cache and how the CPU finds it. We will examine mapping techniques—Direct, Associative, and Set-Associative—and look at the "Hit Ratio," which is the ultimate report card for a cache's efficiency. Finally, we will address the critical issue of data consistency: when the CPU updates a value in the fast cache, how and when does that change make it back to the main memory? By the end of this unit, you will appreciate the sophisticated coordination required to keep a modern digital system running smoothly and efficiently.

16.2. OBJECTIVES

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

- Distinguish between Logical Address Space and Physical Memory Space.
- Analyze the mechanism of Virtual Memory and its role in running large applications.
- Evaluate different Page Replacement algorithms and their impact on system performance.
- Calculate Hit Ratios and understand various Cache mapping techniques.
- Compare Write-through and Write-back methods for maintaining Cache consistency.

16.3. VIRTUAL MEMORY

Virtual Memory is a concept that decouples the addresses used by a program from the addresses available in the physical hardware. It allows the execution of processes that may not be completely in the main memory. This is PARTICULARLY useful in multi-programming environments where many programs share the same physical RAM.

16.3.1. Address Space and Memory Space

In a virtual memory system, we distinguish between two types of addresses. The **Address Space** (or Logical Address Space) is the set of addresses generated by the CPU for a program. The **Memory Space** (or Physical Address Space) is the set of actual locations available in the physical RAM.

If a computer has a 32-bit CPU, it can theoretically address 2^{32} bytes (4 GB) of data. This is its "Address Space." However, the actual computer might only have 1 GB of physical RAM installed. Virtual memory allows the 4 GB address space to be mapped onto the 1 GB physical memory plus the secondary storage (hard disk). When the CPU needs data that isn't in the RAM, the system automatically fetches it from the disk, a process that is invisible to the programmer.

Feature	Address Space (Logical)	Memory Space (Physical)
Generated By	The CPU / Programmer	The Hardware (RAM)
Size	Determined by CPU bit-width	Determined by installed RAM
Visibility	Visible to the software	Hidden from the software
Components	Divided into "Pages"	Divided into "Frames"

NOTES

Table 16.1: Logical Address Space vs. Physical Memory Space

16.4. ADDRESS MAPPING TECHNIQUES

To make Virtual Memory work, the system needs a way to translate a logical address into a physical one instantly. This is handled by a hardware unit called the Memory Management Unit (MMU).

16.4.1. Page Table and Page Replacement

The mapping is done using a **Page Table**. The logical address is divided into two PARTs: a "Page Number" and an "Offset." The Page Table acts as a directory that tells the MMU which physical "Frame" corresponds to a specific "Page." If the CPU looks for a page that isn't currently in a RAM frame, a **Page Fault** occurs, and the operating system must fetch that page from the disk.

But what if the RAM is already full? The system must choose an existing page to kick out to make room for the new one. This is called **Page Replacement**. Common algorithms include:

- **FIFO (First-In-First-Out):** The oldest page in memory is replaced first.
- **LRU (Least Recently Used):** The page that hasn't been accessed for the longest time is replaced. This is based on the principle of "Temporal Locality."
- **Optimal:** Replaces the page that will not be used for the longest period in the future (difficult to implement in practice).

Algorithm	Logic	Complexity	Efficiency
FIFO	Oldest page out	Low	Moderate (Belady's Anomaly)
LRU	Least recently used out	High	Very Good
Optimal	Future-based	Theoretical	Best (Reference only)

Table 16.2 : Comparison of Page Replacement Algorithms

16.5. CACHE MEMORY

While Virtual Memory helps us handle *size*, Cache Memory helps us handle *speed*. Cache memory is a small, high-speed buffer that stores the most frequently used data from the main memory.

NOTES

16.5.1. Hit Ratio and Mapping Techniques

The effectiveness of a cache is measured by the **Hit Ratio**. A "Hit" occurs when the CPU finds the requested data in the cache. A "Miss" occurs when it doesn't.

$$\text{Hit Ratio} = \frac{\text{Total Memory Accesses}}{\text{Number of Hits}} \times 100$$

To find data in the cache, the system uses **Mapping Techniques**:

1. **Direct Mapping:** Each block of main memory maps to exactly one specific line in the cache. It is simple but can lead to "thrashing" if two frequently used blocks map to the same line.
2. **Associative Mapping:** Any block of main memory can be stored in any line of the cache. It is very flexible but requires complex hardware to search the entire cache simultaneously.
3. **Set-Associative Mapping:** A compromise where the cache is divided into sets, and a memory block maps to a specific set but can be placed in any line within that set.

Technique	Hardware Cost	Flexibility	Search Speed
Direct	Lowest	Low	Fastest
Associative	Highest	High	Slower (Complex)
Set-Associative	Moderate	Moderate	Fast

Table 16.3 : Cache Mapping Comparison

16.6. WRITING INTO CACHE MEMORY

When the CPU performs a "Write" operation, it usually updates the value in the Cache. However, this creates a problem: the value in the Cache is now different from the value in the Main Memory.

16.6.1. Methods and Algorithms

There are two primary policies to ensure the Main Memory is eventually updated :

1. **Write-Through:** Every time the CPU writes to the Cache, the same data is simultaneously written to the Main Memory. This ensures high consistency but slows down the system because it must wait for the slower Main Memory write to complete.
2. **Write-Back:** The CPU only updates the Cache. The Main Memory is updated only when that specific block is being replaced in the cache. A "dirty bit" is used to mark if a block has been modified. This is much faster but carries a risk of data loss if a power failure occurs before the update.

Feature	Write-Through	Write-Back
Speed	Slower (Waiting for RAM)	Faster (Cache speed)
Consistency	Immediate / High	Delayed
Complexity	Simple	Complex (Requires Dirty Bit)
Bus Traffic	High	Low

Table 16.4 : Write-Through vs. Write-Back

NOTES

✓ Check Your Progress 16.1.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. Define a "Page Fault."

.....

2. What is the formula for calculating Hit Ratio?

.....

3. Which cache mapping technique is a compromise between Direct and Associative?

.....

**Activity 16.1.**

Imagine you are organizing a small desk (Cache) while working from a large bookshelf (Main Memory).
 Activity title: TITLE OF ACTIVITY: Desk Cache Simulation: LRU vs. FIFO

OBJECTIVE : Understand LRU and FIFO memory replacement strategies using a simple desk analogy.

MATERIAL NEEDED : Pen and paper.

ACTIVITY : 1. Imagine your desk (Cache) holds a maximum of 3 books. The main bookshelf is your Main Memory. 2.

LRU : Your desk is full. You need a 4th book. If you return the book you haven't opened in hours, you are using the Least Recently Used (LRU) algorithm. 3.

NOTES

FIFO : Write down a sequence of 5 books to read (e.g., Book A, B, C, D, B). 4. Simulate bringing them to your desk one by one. 5. When the desk hits its 3-book limit, apply First-In, First-Out (FIFO): you must remove the book that arrived first, regardless of when you last read it. Record the desk's contents at each step.

EXPECTATION : A 5-step written record showing exactly which 3 books were on the desk at any time, clearly identifying which books were evicted to the shelf.

REFLECTION : How do FIFO and LRU differ practically? What is the flaw of FIFO if the "oldest" book on your desk is a frequently used dictionary?

kshelf (Main Memory). If you can only keep 3 books on your desk, and you need a 4th, which one would you put back on the shelf? If you pick the one you haven't touched in hours, you are performing the **LRU algorithm**. Try to write down a sequence of 5 book titles and simulate a "FIFO" replacement strategy on paper.

16.7. SUMMARY

In this unit, we explored the sophisticated mechanisms that allow modern computers to exceed their physical hardware limitations. We began with Virtual Memory, understanding that the CPU operates in a vast Logical Address Space that is seamlessly mapped to a smaller Physical Memory Space through Page Tables. We learned that the Operating System acts as a silent manager, handling Page Faults and using algorithms like LRU and FIFO to decide which data stays in the limited RAM "frames."

Furthermore, we examined the performance-boosting role of Cache Memory. We saw that the goal of any cache is to achieve a high Hit Ratio, ensuring the CPU spends less time waiting for the slower Main Memory. We compared different mapping strategies—Direct, Associative, and Set-Associative—and evaluated how they balance hardware cost with searching flexibility. Finally, we looked at the critical Write policies (Write-through and Write-back) that maintain data integrity across the memory hierarchy. Together, these concepts form the backbone of memory management, ensuring that even as software grows more demanding, hardware can keep pace through intelligent architectural design.

16.8. GLOSSARY

- **Logical Address** : An address generated by the CPU during program execution.
- **Physical Address** : The actual address of a location in the RAM hardware.
- **Page Fault** : An interrupt that occurs when the CPU attempts to access a page not currently in RAM.

- **Dirty Bit** : A status bit used in Write-Back caches to indicate that the cache block has been modified and must be written to main memory before replacement.
- **Thrashing** : A condition where the system spends more time swapping pages in and out than executing instructions.

16.9. ANSWERS TO CHECK YOUR PROGRESS

1. A Page Fault is a trap to the operating system that occurs when a program tries to access a data page that is mapped in virtual address space but not currently loaded in physical RAM.
2. $\text{Hit Ratio} = \frac{\text{Total Hits}}{\text{Total Hits} + \text{Total Misses}}$.
3. Set-Associative Mapping.

16.10. SUGGESTED READINGS – OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectionline.com/User/UnitUrls?unitId=7135	

16.11. REFERENCE

1. Hamacher, V. C. (2011). *Computer Organization*. McGraw-Hill.
2. Stallings, W. (2016). *Computer Organization and Architecture*. Pearson.
3. Tanenbaum, A. S. (2014). *Structured Computer Organization*. Prentice Hall.

16.12. EXERCISE

PART 1

A. Multiple Choice Questions

1. The set of all logical addresses generated by a program is called:
 - (a) Memory Space
 - (b) Address Space
 - (c) Frame Space
 - (d) Cache Space
2. Which Page Replacement algorithm suffers from Belady's Anomaly?
 - (a) LRU
 - (b) Optimal
 - (c) FIFO
 - (d) MRU
3. In a 2-way Set-Associative cache, each set contains:
 - (a) 1 line
 - (b) 2 lines
 - (c) 4 lines
 - (d) The entire cache
4. Which cache write policy is the fastest for the CPU?
 - (a) Write-Through
 - (b) Write-Once
 - (c) Write-Back
 - (d) Read-Only

NOTES

5. The hardware unit responsible for translating virtual addresses to physical addresses is:

- (a) ALU (b) Control Unit (c) MMU (d) Register File

Answers with explanations:

1. **Answer : (b)**

Explanation : The Address Space represents the total range of logical addresses that the CPU can generate for a program, regardless of the physical memory available.

2. **Answer : (c)**

Explanation : Belady's Anomaly is a phenomenon where increasing the number of page frames results in an increase in page faults; this specifically affects the FIFO algorithm.

3. **Answer : (b)**

Explanation : In N-way set-associative mapping, each "Set" contains exactly "N" lines. Therefore, a 2-way set has 2 lines.

4. **Answer : (c)**

Explanation : Write-back is faster because the CPU only has to update the high-speed cache and can proceed immediately, without waiting for the slower main memory to finish writing.

5. **Answer : (c)**

Explanation : The MMU (Memory Management Unit) is a dedicated hardware component that handles the mapping and translation of logical addresses into physical ones in real-time.

B. Short answer type questions

1. Distinguish between a Page and a Frame.
2. Explain the concept of "Locality of Reference."
3. What is the primary disadvantage of Direct Mapping in cache memory?
4. Define "Hit Time" and "Miss Penalty."

PART – 2

C. Long answer type question (200 words)

1. Describe the steps taken by the Operating System when a "Page Fault" occurs. How does the system recover the data and resume execution?
2. Explain the working of the Least Recently Used (LRU) algorithm. Why is it generally considered more efficient than the FIFO algorithm?
3. Compare Write-Through and Write-Back policies. Discuss a scenario where Write-Through would be preferable despite being slower.

PART – 3

D. Long answer type question (300 words)

1. Analyze the three Cache Mapping techniques: Direct, Associative, and Set-Associative. Provide a technical

comparison of their hardware complexity, search logic, and impact on the Hit Ratio. Why is Set-Associative mapping used in most modern high-performance processors?

2. Discuss the mathematical relationship between the Address Space and Memory Space in a 64-bit architecture with only 16 GB of RAM. Explain how Page Tables and the MMU facilitate this massive difference in size, and describe the role of the "Offset" in locating a specific byte within a page.
3. Evaluate the concept of the "Memory Wall"—the increasing gap between CPU speed and Main Memory access time. How do Advanced Memory Concepts like Multi-level Caching (L1, L2, L3) and Virtual Memory attempt to solve this bottleneck? Use the concept of Hit Ratios and Average Memory Access Time (AMAT) in your discussion.

NOTES



DR. C.V. RAMAN UNIVERSITY

Chhattisgarh, Bilaspur AN AISCT GROUP UNIVERSITY

Approved by: PCI | AICTE | NCTE | BCI | Member of: AU | Joint Committee: UGC | SER | AICTE | Recognized by: UGC | A NAAC Accredited University



DR. C. V. RAMAN UNIVERSITY (C.G.)

SLM FEEDBACK FORM

PROGRAMME : _____

COURSE : _____

BLOCK : _____

UNIT : _____

Dear Student,

While studying the units of this block, you may have found certain portions of the text difficult to comprehend. We wish to know your difficulties and suggestions, in order to improve the course. Therefore, we request you to fill up and send us the following questionnaire, which pertains to this block. If you find the space provided insufficient, kindly use a separate sheet.

Please mail to aisectcontent@aisect.org/ for online submission please scan QR code:

Questionnaire

1. Name : _____

2. Enrolment No. :

--	--	--	--	--	--	--	--	--	--	--

3. How many hours did you need for studying the units? :

--

4. Please give your reactions to the following items based on your reading of the block:

Items	Excellent	Very Good	Good	Poor	Give specific Examples (If poor)
Presentation Quality					
Language and Style					
Illustrations					
Used (diagrams, tables, etc.)					
Conceptual Clarity					
Check Your Progress Questions					
Suggested Readings - OER					

5. Any other comments:

Block V	ADVANCED SYSTEM DESIGN AND MICROPROCESSORS	187 - 254
Unit 17	Arithmetic and Logic Circuits Architecture	189
Unit 18	Digital Design with PLDs	197
Unit 19	Timing and Control Circuits	209
Unit 20	Microprocessor Basics	219
Unit 21	Digital System Design	229
Unit 22	Future Trends in Digital Electronics	241



This final section marks the culmination of your digital electronics training, where the atomized concepts of logic, arithmetic, and memory converge into a complete computing architecture. We will analyze the core design of the **Microprocessor**, starting with its key functional components: the **Arithmetic and Logic Unit (ALU)**, the Control Unit, and the Register File. You will understand how these blocks are coordinated by complex timing protocols to execute software instructions. This block also introduces Programmable Logic Devices (PLDs), a paradigm shift from rigid hardware design using physical logic gates to soft-driven methodology. Engineers use PLDs (like FPGAs) to rapidly prototype entire digital systems using only software, effectively creating a programmable sandbox for hardware engineering. Finally, we look toward the engineering process itself and future trends. You will synthesize your knowledge to architect a complete digital system, integrating optimal speed, cost, and power constraints, before concluding with an exploration of emerging technologies, such as Quantum Computing and AI Acceleration, which are redefining the boundaries of digital systems prerequisites.

UNIT – 17

ARITHMETIC AND LOGIC CIRCUITS

NOTES

UNIT STRUCTURE

- 17.1. Introduction
- 17.2. Objective
- 17.3. Arithmetic Circuits
 - 17.3.1. Binary Adders and Subtractors
- 17.4. Logic Circuits
 - 17.4.1. Comparators
- 17.5. Design of ALUs
- 17.6. Summary
- 17.7. Glossary
- 17.8. Answers to check you progress – possible one
- 17.9. Suggested readings - OER
- 17.10. Reference
- 17.11. Exercise

17.1. INTRODUCTION

We have explored the basic "alphabet" of digital systems—binary numbers, logic gates, and Boolean algebra. We also looked at how memory elements store data. But storage alone doesn't make a computer smart; the ability to *process* that data does. Have you ever wondered how your computer actually performs a calculation when you open the calculator app, or how it decides which number is larger when sorting a list of files? The magic lies in Arithmetic and Logic Circuits.

In this unit, we are going to assemble the basic logic gates into the functional "muscles" of a computational system. We will begin by exploring Arithmetic Circuits, specifically focusing on binary adders and subtractors. You will see how a simple combination of XOR, AND, and OR gates can perform mathematical addition and subtraction, keeping track of "carries" and "borrows" just like you do when doing math on paper.

NOTES

Next, we will transition into Logic Circuits, with a special focus on Comparators. These are the decision-making circuits that evaluate two binary numbers to determine if they are equal, or if one is greater or less than the other. Finally, we will bring all these concepts together to understand the design of the Arithmetic Logic Unit (ALU). The ALU is the very heart of any Central Processing Unit (CPU). It acts as a versatile processing block where arithmetic and logic meet, controlled by specific command codes. By the end of this unit, you will have a clear picture of how raw binary data is manipulated to solve complex computational problems. Let's get started!

17.2. OBJECTIVE

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

- Understand the fundamental principles and logic behind arithmetic circuits.
- Design and analyze Binary Adders and Subtractors (Half and Full configurations).
- Explain the functioning of Logic Circuits, specifically Magnitude Comparators.
- Evaluate the internal architecture and operational control of an Arithmetic Logic Unit (ALU).
- Apply the concepts of selection lines to switch between arithmetic and logical operations.

17.3. ARITHMETIC CIRCUITS

Arithmetic circuits are combinational logic circuits designed to perform mathematical operations on binary numbers. The most fundamental operations in any digital processor are addition and subtraction. Even complex operations like multiplication and division are essentially repeated addition and subtraction at the hardware level.

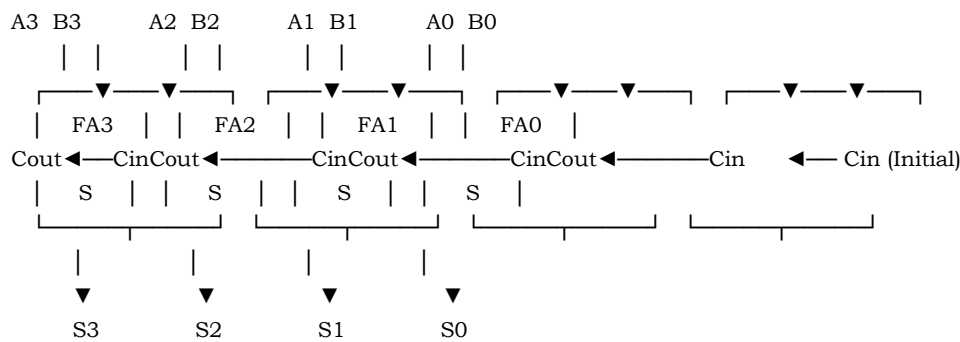
17.3.1. Binary Adders and Subtractors

To add binary numbers, we start with the **Half-Adder**. It takes two single-bit inputs (A and B) and produces a Sum and a Carry. However, when adding multi-bit numbers, we need a circuit that can also accept a carry from a previous lower-order addition. This requires a **Full-Adder**, which takes three inputs (A, B, and Carry-in) and produces a Sum and a Carry-out. By chaining multiple Full-Adders together, we can add 4-bit, 8-bit, or even 64-bit numbers.

Subtraction follows a similar logic. A **Half-Subtractor** takes two bits and computes the Difference and a Borrow. A **Full-Subtractor** accounts for a Borrow-in from a previous stage. In modern systems, instead of building separate subtractor circuits, we often use Full-Adders combined with 2's complement logic to perform subtraction, which saves hardware space and reduces costs.

Circuit Type	Number of Inputs	Number of Outputs	Primary Limitation / Use
Half-Adder	2 (A, B)	2 (Sum, Carry)	Cannot accept a carry from a previous stage.
Full-Adder	3 (A, B, Cin)	2 (Sum, Cout)	Building block for multi-bit ripple-carry adders.
Half-Subtractor	2 (A, B)	2 (Diff, Borrow)	Cannot accept a borrow from a previous stage.
Full-Subtractor	3 (A, B, Bin)	2 (Diff, Bout)	Handles multi-bit binary subtraction.

NOTES

Table 17.1 : Comparison of Basic Arithmetic Circuits.**Diagram Component Breakdown**

- **Full Adder (FA) Blocks:** Each box represents a 1-bit Full Adder logic circuit.
- **Inputs (A_n, B_n):** The individual bits of the two 4-bit numbers being added.
- **Sum Outputs (S_n):** The resulting bits of the addition.
- **The Carry Chain (The "Ripple"):**
 - **C_{in} (Initial):** Usually tied to Ground (0) for simple addition.
 - **$C_{out} \rightarrow C_{in}$:** The carry-out of the lower bit "ripples" into the carry-in of the next higher bit.
 - **C_{out} (Final):** The carry-out of the MSB ($F A_3$), which acts as the 5th bit of the result (overflow).

✓ Check Your Progress 17.1.**Notes :** (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. What logic gate is primarily responsible for producing the "Sum" in a Half-Adder?

NOTES

-
-
-
2. Why is a Full-Adder required for adding binary numbers larger than 1 bit?
-
-
-

17.4. LOGIC CIRCUITS

While arithmetic circuits crunch numbers, Logic Circuits make decisions. In the context of processing, logic circuits refer to combinational blocks that evaluate the state of binary data without necessarily performing mathematical math on them.

17.4.1. Comparators

A Magnitude Comparator is a specialized logic circuit that compares two binary numbers (let's call them A and B) and determines their relative magnitudes. The circuit provides three distinct outputs:

- 1. $A > B$ (A is greater than B)
- 2. $A = B$ (A is equal to B)
- 3. $A < B$ (A is less than B)

At the foundational level, a **1-bit Comparator** compares two single bits. It uses an XNOR gate to determine equality (since XNOR outputs 1 only when both inputs are the same). To find if $A > B$, it uses the logic $A \cdot \overline{B}$ (A is 1 and B is 0). To find if $A < B$, it uses $\overline{A} \cdot B$ (A is 0 and B is 1). For multi-bit numbers (like comparing two 4-bit numbers), the circuit compares the Most Significant Bits (MSB) first, and if they are equal, it cascades down to the lower-order bits. Comparators are essential in sorting algorithms, password verification, and conditional branching in microprocessors.

Input A	Input B	A > B Output	A = B Output	A < B Output
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Table 10.2 : Truth Table for a 1-Bit Magnitude Comparator.



Activity 17.1. THE SMART THERMOSTAT LOGIC

NOTES

Instructional Dialogue: Now that you have explored magnitude comparators, let's see how this logic applies to a real-world digital appliance. Imagine you are programming the logic for a smart thermostat.

Measurable Objectives:

- **Analyze** the inputs of a temperature control system.
- **Apply** comparator logic to trigger specific hardware actions.

Segmented Tasks :

1. **Identify the Data :** Assume the "Target Temperature" is Binary Number A, and the "Current Room Temperature" is Binary Number B.
2. **Assign the Logic :** Map the three outputs of a comparator ($A > B$, $A = B$, $A < B$) to three physical actions: "Turn on Heater", "Turn on AC", and "Standby".
3. **Write the Rule :** Write a brief 50-word logic rule explaining what the system should do for each comparator output.

Contextualization : Every automated climate control system in cars, homes, and industrial plants uses digital comparators at its core to decide whether to heat, cool, or do nothing based on sensor readings.

Self-Evaluation Key:

- **Check your logic :** If $A > B$ (Target is greater than Current), your system must "Turn on Heater." If $A < B$ (Target is less than Current), it must "Turn on AC." If $A = B$, it must go to "Standby." If your mapping matches this, you have successfully designed a functional comparator application!

17.5. DESIGN OF ALUS

The **Arithmetic Logic Unit (ALU)** is the crowning achievement of combinational logic design. It is a multi-purpose, unified block of hardware that combines both Arithmetic circuits (Adders/Subtractors) and Logic circuits (Comparators, AND, OR, XOR operations) into a single module.

Instead of having a separate chip for every operation, an ALU uses **Selection Lines** (also known as Op-codes) to determine what operation to perform at any given moment. For example, if an ALU has two selection lines (S_1, S_0), it can perform four different operations. The ALU takes two multi-bit data inputs (Operands), receives an operation code from the Control Unit, processes the data through the selected internal circuit, and sends the result to the output bus. It also outputs "Status Flags" (like Zero, Carry, Negative, or Overflow) which tell the system

NOTES

about the nature of the result. The design of an ALU is what dictates the processing power and instruction set of a CPU.

Selection Line S1	Selection Line S0	Operation Performed	Type of Operation
0	0	Addition (A + B)	Arithmetic
0	1	Subtraction (A - B)	Arithmetic
1	0	Bitwise AND (A $\cdot$ B)	Logical
1	1	Bitwise XOR (A $\oplus$ B)	Logical

Table 10.3 : Example of ALU Operation Codes (2-bit Selection).

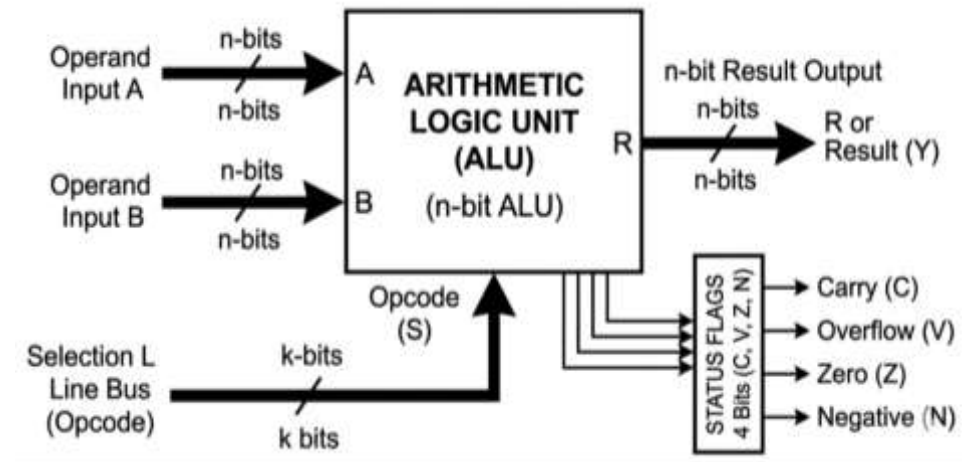


Figure 17.1 : Block schematic of an ALU showing two n-bit operand inputs.

17.6. SUMMARY

In this concluding unit, we unraveled the mechanisms that enable digital computers to process data mathematically and logically. We started with Arithmetic Circuits, breaking down the construction of Half-Adders, Full-Adders, and Subtractors, establishing that the ability to track carries and borrows is what allows processors to handle large binary numbers. We then shifted to Logic Circuits, focusing on Magnitude Comparators. By analyzing the 1-bit comparator truth table, we saw how XNOR, AND, and NOT gates are orchestrated to make relative value judgments (>, <, =).

Finally, we brought these discrete circuits together in the study of the Arithmetic Logic Unit (ALU). We learned that the ALU is a highly integrated module that relies on multiplexers and selection lines to dynamically switch between arithmetic addition and logical operations based on software instructions. Understanding the ALU bridges the gap between basic hardware gates and complex software programming, providing a complete picture of how digital systems "think" and calculate.

17.7. GLOSSARY

- **Operand:** The data or binary number upon which an operation is to be performed.
- **Carry-in (Cin):** A bit carried over from a less significant mathematical operation into the current operation.
- **Magnitude Comparator:** A digital circuit that compares two numbers and determines if they are equal or if one is greater than the other.
- **Op-code (Operation Code):** A binary sequence sent via selection lines to instruct the ALU on which specific operation to perform.
- **Status Flags:** Extra output bits from an ALU that indicate specific conditions of the result, such as whether it is zero or negative.

17.8. ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 10.1

1. The **XOR (Exclusive-OR) gate** is responsible for producing the Sum in a Half-Adder.
2. A Full-Adder is required for multi-bit addition because it features a **Carry-in input**, allowing it to accept and process the carry bit generated by the addition of the previous, lower-order bits.

17.9. SUGGESTED READINGS - OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectionline.com/User/UnitUrls?unitId=7136	

17.10. REFERENCE

1. Mano, M. M., & Ciletti, M. D. (2013). *Digital Design: With an Introduction to the Verilog HDL*. Pearson.
2. Floyd, T. L. (2015). *Digital Fundamentals*. Pearson Education.
3. Tocci, R. J., Widmer, N. S., & Moss, G. L. (2017). *Digital Systems: Principles and Applications*. Pearson.

17.11. EXERCISE

PART 1

A. Multiple Choice Questions

1. A Half-Adder circuit is constructed using which two logic gates?

NOTES

NOTES

- (a) AND and OR
 - (b) XOR and AND
 - (c) XNOR and OR
 - (d) NAND and NOR
2. In a 1-bit Magnitude Comparator, the logic expression for $A = B$ is implemented using which gate?
- (a) XOR
 - (b) XNOR
 - (c) NAND
 - (d) OR
3. To construct an 8-bit parallel ripple-carry adder, how many Full-Adders are typically required?
- (a) 4
 - (b) 7
 - (c) 8
 - (d) 16
4. Which of the following is an Arithmetic operation inside an ALU?
- (a) Bitwise AND
 - (b) Subtraction
 - (c) Logical Shift
 - (d) Bitwise XOR
5. The input signals that tell the ALU which specific operation to perform are called:
- (a) Operands
 - (b) Status Flags
 - (c) Selection Lines
 - (d) Data Buses

Answers with explanations:

1. **Answer : (b)**

Explanation : A Half-Adder uses an XOR gate to calculate the Sum (outputting 1 when inputs are different) and an AND gate to calculate the Carry (outputting 1 only when both inputs are 1).

2. **Answer : (b)**

Explanation : The XNOR gate is the equality gate. It outputs a logical High (1) only when both inputs are exactly the same (either 0,0 or 1,1), making it perfect for the $A=B$ condition.

3. **Answer : (c)**

Explanation : A parallel ripple-carry adder requires one Full-Adder for each bit position. Therefore, to add two 8-bit numbers together, exactly 8 Full-Adders are needed.

4. **Answer : (b)**

Explanation : Subtraction involves numerical calculation involving borrows, making it an arithmetic operation. AND, Shift, and XOR are strictly logical or bit-manipulation operations.

5. **Answer : (c)**

Explanation : Selection lines (or control lines/op-codes) are the inputs fed into the ALU by the Control Unit that determine the multiplexer paths, selecting whether the ALU will add, subtract, AND, or OR the operands.

B. Short answer type questions

1. State the difference between a Half-Subtractor and a Full-Subtractor.
2. What are the three possible outputs of a digital Magnitude Comparator?
3. Briefly define the purpose of "Status Flags" in an ALU.
4. Write the boolean expression for the "Borrow" output of a Half-Subtractor.

PART 2

C. Long answer type question (200 words)

1. Describe the concept of a "Ripple Carry Adder." Explain why the propagation delay increases as you add more bits, and briefly mention how this limitation is overcome in modern processors.
2. Explain the logic and gate implementation required to determine the $A > B$ condition in a 1-bit magnitude comparator. Provide the truth table segment that corresponds to this specific outcome.
3. How is a Full-Adder used to perform binary subtraction? Discuss the concept of 2's complement arithmetic and how adding an inverter and a carry-in of 1 transforms an adder into a subtractor.
4. Discuss the role of the Arithmetic Logic Unit (ALU) within the broader architecture of a Central Processing Unit (CPU). How does the ALU interact with the CPU's registers during an instruction cycle?

NOTES**D. Long answer type question (300 words)**

1. Provide a detailed functional and structural analysis of a Full-Adder circuit. Derive the Boolean expressions for both the 'Sum' and the 'Carry-out' outputs using a 3-input truth table. Furthermore, explain how you can construct a Full-Adder using exactly two Half-Adders and one OR gate, detailing the flow of logic through this combined circuit.
2. Analyze the design and operation of a 4-bit Magnitude Comparator. Explain the sequential logic process the circuit uses to evaluate two 4-bit numbers (e.g., A₃A₂ A₁ A₀ and B₃ B₂ B₁ B₀). Why does the circuit always begin its comparison at the Most Significant Bit (MSB), and under what specific condition does it move on to evaluate the lower-order bits?
3. Design a theoretical, simple 1-bit ALU that can perform Addition, Subtraction, Bitwise AND, and Bitwise OR. Outline the block diagram components required (including multiplexers and basic logic gates). Discuss the role of a 2-bit selection code in routing the operands through the correct logic block and ensuring only the requested result is sent to the final output line.

UNIT – 18

DIGITAL DESIGN WITH PLDS

NOTES

UNIT STRUCTURE

- 18.1. Introduction
- 18.2. Objective
- 18.3. Programmable Logic Devices
 - 18.3.1. PAL, PLA, and FPGA
- 18.4. Applications of PLDs in Digital Design
- 18.5. Summary
- 18.6. Glossary
- 18.7. Answers to check you progress – possible one
- 18.8. Suggested readings - OER
- 18.9. Reference
- 18.10. Exercise

18.1. INTRODUCTION

Historically, engineering a digital circuit meant physically wiring together discrete Logic Gates (like those found in the 7400-series TTL chips). If an engineer wanted to build a custom counter or a specialized arithmetic unit, they had to design a custom Printed Circuit Board (PCB) and solder dozens of individual chips together. This method was slow, consumed massive amounts of physical space, required high power, and was completely rigid—if a single logic error was discovered after manufacturing, the entire board had to be discarded and redesigned.

The invention of **Programmable Logic Devices (PLDs)** revolutionized digital hardware engineering. A PLD is essentially a "blank canvas" integrated circuit. Instead of containing a fixed, unchangeable logic function, a PLD contains a massive internal grid of generic logic gates (AND and OR gates) with programmable connections. By using specialized software and hardware programmers, an engineer can electrically blow or intact these internal connections to configure the chip into almost any digital circuit imaginable.

In this unit, we transition from rigid hardware design to "soft" hardware design. We will explore the evolutionary architectures of PLDs—from early Programmable Array Logic (PAL) to modern, immensely powerful Field-Programmable Gate Arrays (FPGAs)—which form the backbone of contemporary prototyping, telecommunications, and custom processor design.

NOTES

18.2. OBJECTIVES:

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

- Define the concept and physical advantages of Programmable Logic Devices (PLDs).
- Analyze the internal grid architecture of programmable AND and OR arrays.
- Differentiate between the hardware structures of PLA, PAL, and PROM.
- Describe the advanced architecture of Field-Programmable Gate Arrays (FPGAs), including Configurable Logic Blocks (CLBs).
- Evaluate the workflow and real-world applications of PLDs in modern digital system design.

18.3. PROGRAMMABLE LOGIC DEVICES

A Programmable Logic Device (PLD) relies on the fundamental theorem of Boolean algebra: any combinational logic function can be expressed in Sum-of-Products (SOP) form. Because of this mathematical law, the internal architecture of almost all basic PLDs consists of two distinct stages :

1. **An AND Array:** Generates the product terms (the "minterms").
2. **An OR Array:** Sums the product terms together.

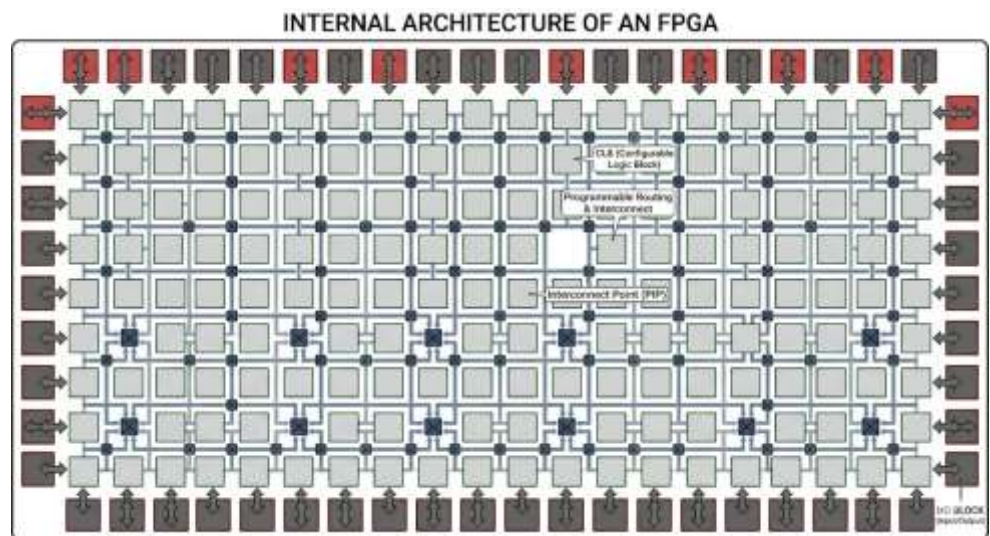


Figure 18.1: Internal Architecture of an FPGA

In a PLD, inputs enter the chip and are immediately split into their true and complemented forms (e.g., A and $\bar{A}$). These signals run vertically across the chip. The inputs to the AND gates run horizontally. At every intersection, there is a programmable "fuse." If the fuse is left intact, the connection is made; if the fuse is electrically "blown" by the programmer, the connection is broken. This grid intersection logic allows engineers to physically burn Boolean equations directly into the silicon.

✓ Check Your Progress 18.1.**NOTES**

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. What fundamental mathematical form allows PLDs to represent any combinational logic circuit?

.....

2. What are the two primary logic gate arrays found inside a standard PLD?

.....

3. What is the physical mechanism used to program the connections inside early PLDs?

.....

18.3.1. PAL, PLA, and FPGA

The classification of PLDs depends entirely on which of the internal arrays (the AND array or the OR array) is programmable and which is permanently fixed by the manufacturer.

1. Programmable Logic Array (PLA)

The PLA is the most flexible of the traditional PLDs. In a PLA, **both the AND array and the OR array are programmable.**

- An engineer can program exactly which inputs feed into which AND gates to create custom product terms.
- The engineer can then program exactly which product terms feed into which OR gates.

While highly flexible, the PLA has a major drawback: having two fully programmable grid matrices introduces significant electrical resistance and capacitance, leading to high propagation delays. PLAs are relatively slow.

2. Programmable Array Logic (PAL)

To solve the speed issues of the PLA, engineers developed the PAL. In a PAL, **the AND array is programmable, but the OR array is permanently fixed** at the factory.

- The engineer can still create custom product terms by programming the AND grid.
- However, those AND gates are hardwired to specific OR gates.

NOTES

Because there is only one programmable matrix, the PAL is significantly faster, cheaper to manufacture, and easier to program than the PLA. It became the industry standard for implementing "glue logic" (small custom circuits used to interface larger chips on a motherboard).

3. Field-Programmable Gate Array (FPGA)

While PALs and PLAs are excellent for basic logic, they are too small to contain an entire microprocessor or a complex telecommunications router. For massive digital systems, the industry uses the **FPGA**.

Unlike PLAs or PALs, an FPGA does not use a simple AND/OR grid. Instead, it utilizes an advanced, sea-of-gates architecture. An FPGA consists of three main components:

1. **Configurable Logic Blocks (CLBs):** Thousands of individual logic "islands" spread across the chip. Each CLB contains Look-Up Tables (LUTs) to process Boolean logic, and flip-flops to store sequential data.
2. **Programmable Interconnects:** A massive web of programmable wiring (a switch matrix) that allows the engineer to route signals from any one CLB to any other CLB across the chip.
3. **I/O Blocks:** Specialized programmable pins that interface the internal logic with the external physical circuit board.

Because FPGAs contain integrated flip-flops, they are not limited to combinational logic; they can easily implement complex sequential state machines, memory architectures, and complete multi-core CPUs.

✓ Check Your Progress 18.2.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. In a Programmable Array Logic (PAL) device, which array is fixed and which is programmable?

.....

2. Why is a PAL generally faster than a PLA?

.....

3. What is the function of a Configurable Logic Block (CLB) inside an FPGA?

.....

NOTES**18.4. APPLICATIONS OF PLDS IN DIGITAL DESIGN**

The implementation of PLDs, specifically FPGAs, has fundamentally changed the hardware design workflow. Today, engineers rarely draw logic gate diagrams by hand. Instead, they write code.

The Hardware Description Language (HDL) Workflow

1. **Coding:** An engineer describes the behavior of the circuit using a text-based Hardware Description Language (HDL), such as Verilog or VHDL.
2. **Synthesis:** Specialized compiler software analyzes the code and translates it into a "Netlist"—a massive mathematical list of required logic gates and flip-flops.
3. **Implementation & Bitstream:** The software maps the Netlist onto the specific physical architecture of the target FPGA. It then generates a "Bitstream" file. When this file is downloaded onto the FPGA, the chip electrically reconfigures its internal switches to become the custom circuit.

Real-World Applications:

- **ASIC Prototyping:** Application-Specific Integrated Circuits (ASICs) cost millions of dollars to manufacture. Engineers use FPGAs to build and physically test the logic of a new computer chip before committing to permanent ASIC silicon manufacturing.

NOTES

- **Digital Signal Processing (DSP):** FPGAs excel at parallel processing, making them ideal for handling massive streams of data in real-time, such as radar systems, medical MRI imaging, and 5G cellular base stations.
- **Aerospace and Defense:** Because hardware can be updated remotely by flashing a new Bitstream, FPGAs are used heavily in satellites and military hardware where physical chip replacement is impossible.

 Check Your Progress 18.3.

Notes : (a) Space is given below for your answer.
(b) Check your answer with the one given at the end of this unit.

1. What does the acronym HDL stand for in digital design?
.....
.....
.
2. What is a "Bitstream" in the context of FPGA programming?
.....
.....
.....
3. Why are FPGAs used for prototyping ASICs?
.....
.....
.....

 Activity 18.1 : Paper-Based PLA Programming Simulation

Objective : To manually map a set of Boolean Sum-of-Products (SOP) expressions onto a Programmable Logic Array (PLA) grid by determining which electrical "fuses" to keep intact and which to blow.

Description : In this exercise, you will act as the programming hardware, mapping a simple Half Adder circuit onto a blank PLA matrix. A Half Adder has two inputs (A and B) and produces two outputs: Sum S and Carry C.

The Boolean equations for a Half Adder are:

Sum S: $A \cdot B + \bar{A} \cdot \bar{B}$

Carry C: $A \cdot B$

Step 1: Set Up the Blank Matrix

1. Draw four vertical lines representing your input variables and their complements: A, $\bar{A}$, B, $\bar{B}$.

2. Draw three horizontal lines crossing over the four vertical lines. These represent the inputs to three separate AND gates (which will generate your product terms).
3. Draw two new vertical lines to the right of your AND gates. These represent the inputs to two separate OR gates (which will generate your final outputs, S and C). Extend the outputs of your three AND gates so they cross these two new vertical OR lines.

Step 2 : Program the AND Array (Generating Product Terms)

In a PLA, an intact fuse is represented by an "X" at an intersection.

1. **Product Term 1 ($A \cdot \bar{B}$)** : On the first horizontal AND line, place an "X" where it crosses the A line and the $\bar{B}$ line.
2. **Product Term 2 ($\bar{A} \cdot B$)** : On the second horizontal AND line, place an "X" where it crosses the $\bar{A}$ line and the B line.
3. **Product Term 3 ($A \cdot B$)** : On the third horizontal AND line, place an "X" where it crosses the A line and the B line.

Step 3: Program the OR Array (Generating Sum Terms)

Now you must connect your generated product terms to the correct output lines.

1. **Output S** : On the vertical line representing the OR gate for S, place an "X" where it intersects with the horizontal line for Product Term 1 and Product Term 2.
2. **Output C** : On the vertical line representing the OR gate for C, place an "X" where it intersects with the horizontal line for Product Term 3.

Reflection:

You have successfully programmed a PLA on paper! Look closely at your matrix. If this were a **PAL** (Programmable Array Logic) instead of a PLA, you would only have been allowed to draw "X" marks on the first step (the AND array). The connections for the OR array in Step 3 would have been permanently drawn in at the factory, and you would have had to carefully select which fixed OR gates to route your custom AND gates into.

18.5 SUMMARY

In this unit, we explored the transformative shift from discrete logic hardware to Programmable Logic Devices (PLDs). We examined how the fundamental Sum-of-Products Boolean form allows chips with programmable AND/OR arrays to physically manifest any combinational logic circuit. We differentiated the architectures of the highly flexible but slower PLA (programmable AND, programmable OR) from the faster, industry-standard PAL (programmable AND, fixed OR). We then advanced to the Field-Programmable Gate Array (FPGA), detailing its complex architecture of Configurable Logic Blocks (CLBs)

NOTES

and programmable interconnects, which allow for the design of massive, highly integrated sequential systems. Finally, we reviewed the modern digital design workflow, highlighting how Hardware Description Languages (HDLs) are synthesized into bitstreams to instantly reconfigure FPGA silicon for high-performance applications like signal processing and ASIC prototyping.

18.6. GLOSSARY

- **Bitstream:** A binary configuration file generated by synthesis software that dictates the physical state of the internal programmable switches within an FPGA.
- **CLB (Configurable Logic Block):** The fundamental logic building block inside an FPGA, typically containing Look-Up Tables (LUTs), multiplexers, and flip-flops.
- **FPGA (Field-Programmable Gate Array):** A high-density programmable logic device featuring thousands of CLBs connected by a massive programmable routing network.
- **HDL (Hardware Description Language):** A specialized programming language (like Verilog or VHDL) used to describe the structure and behavior of digital logic circuits.
- **PAL (Programmable Array Logic):** A PLD architecture featuring a programmable AND array and a permanently fixed OR array.
- **PLA (Programmable Logic Array):** A PLD architecture featuring both a programmable AND array and a programmable OR array.

18.7 ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 18.1

1. The Sum-of-Products (SOP) form.
2. The AND array and the OR array.
3. Programmable electrical "fuses" at the intersections of the internal grid matrix.

Check Your Progress 18.2

1. In a PAL, the AND array is programmable, but the OR array is permanently fixed.
2. Because a PAL only has one programmable matrix, it has less internal electrical resistance and capacitance, significantly reducing propagation delay.
3. A CLB acts as an independent logic "island" that performs custom Boolean logic (via Look-Up Tables) and provides sequential memory storage (via internal flip-flops).

Check Your Progress 18.3

1. Hardware Description Language.

2. A binary file that is downloaded into the FPGA to electrically set all of its internal programmable switches, turning the blank chip into a functional custom circuit.
3. Because ASICs are incredibly expensive and unchangeable once manufactured. FPGAs allow engineers to test the hardware logic physically in the real world and fix errors instantly before spending money on ASIC fabrication.

18.8. SUGGESTED READINGS – OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectonline.com/User/UnitUrls?unitId=7137	

18.9 REFERENCE

1. Floyd, T. L. (2015). *Digital Fundamentals*. Pearson Education.
2. Mano, M. M., & Ciletti, M. D. (2013). *Digital Design*. Pearson.
3. Tocci, R. J., Widmer, N. S., & Moss, G. L. (2017). *Digital Systems: Principles and Applications*. Pearson.

18.10. EXERCISE

PART 1

A. Multiple choice question

1. In a Programmable Logic Array (PLA), which internal structures are configured by the user?
 - (a) Only the AND array.
 - (b) Only the OR array.
 - (c) Both the AND array and the OR array.
 - (d) Neither; they are fixed at the factory.
2. Why did the PAL (Programmable Array Logic) architecture largely replace the PLA in early digital design?
 - (a) PALs use less power and operate at much higher speeds.
 - (b) PALs have programmable OR arrays which are more flexible.
 - (c) PALs contain complex microprocessors inside them.
 - (d) PALs do not require software to program.
3. The basic programmable logic unit inside an FPGA is known as a:
 - (a) Logic Gate Row (LGR)
 - (b) Configurable Logic Block (CLB)
 - (c) Hardwired OR Matrix (HOM)

NOTES

- (d) Bitstream Execution Unit (BEU)
4. What file format is ultimately loaded onto an FPGA to physically configure its internal interconnects and CLBs?
 - (a) An HTML file
 - (b) A C++ script
 - (c) A Bitstream
 - (d) A Netlist
 5. Which of the following is NOT a typical advantage of using an FPGA over designing a custom ASIC?
 - (a) Faster time-to-market.
 - (b) The ability to reprogram the hardware remotely after deployment.
 - (c) Lower power consumption at massive manufacturing scales.
 - (d) Lower upfront development costs.

Answers with Explanations:

1. **Answer : (c)**

Explanation : A PLA is defined by having maximum flexibility, meaning the user can program the connections for both the AND matrix (product terms) and the OR matrix (sum terms).

2. **Answer : (a)**

Explanation : By permanently fixing the OR array, the PAL eliminates the propagation delay caused by a second programmable grid matrix, allowing for much faster logic execution.

3. **Answer : (b)**

Explanation: FPGAs are built using thousands of Configurable Logic Blocks (CLBs) spread across the chip, which process the Boolean logic and store sequential states.

4. **Answer : (c)**

Explanation: The synthesis software generates a binary Bitstream, which is the exact sequence of 1s and 0s required to open and close the millions of tiny programmable switches inside the FPGA silicon.

5. **Answer : (c)**

Explanation : While FPGAs are great for prototyping and low-volume production, custom ASICs are fundamentally more power-efficient and cost-effective when manufactured in millions of units.

B. Short answer type question

1. Define the fundamental structural difference between a PAL and a PLA.
2. What role does a Hardware Description Language (HDL) play in modern PLD design?
3. Briefly explain the function of the "programmable interconnects" inside an FPGA.
4. Why does the fundamental architecture of basic PLDs rely heavily on the Sum-of-Products (SOP) Boolean form?

5. State one major real-world application where an FPGA is highly preferred over a standard microprocessor.

PART 2

C. Long answer type question (200 words)

1. Describe the evolutionary path from standard discrete logic gates (like the 7400 series) to Programmable Array Logic (PAL) devices. Detail the specific engineering limitations of discrete logic that necessitated the invention of programmable internal gate arrays.
2. Explain the standard FPGA design workflow, detailing the transition from writing Verilog/VHDL code, through the software synthesis process, to the final generation and deployment of the configuration bitstream.

PART 3

D. Long answer type question (Answer in 300 words)

1. Analyze the internal architecture of a Field-Programmable Gate Array (FPGA). Deconstruct the roles of the Configurable Logic Blocks (CLBs), the programmable routing switch matrix, and the Input/Output (I/O) blocks. Explain how the integration of Look-Up Tables (LUTs) and internal flip-flops allows an FPGA to execute complex sequential logic that older PLDs could not handle.
2. Evaluate the engineering trade-offs between utilizing an FPGA versus manufacturing an Application-Specific Integrated Circuit (ASIC). In your evaluation, compare the two technologies regarding initial development costs (NRE), unit costs at high production volumes, hardware flexibility post-deployment, and overall power efficiency, providing specific use-case scenarios for each.

NOTES

NOTES

UNIT – 19

TIMING AND CONTROL CIRCUITS

NOTES

UNIT STRUCTURE

- 19.1. Introduction
- 19.2. Objective
- 19.3. Design of Clocks and Timing Circuits
 - 19.3.1. Applications in Sequencers
- 19.4. Overview of Control Circuits
 - 19.4.1. Design Examples
- 19.5. Summary
- 19.6. Glossary
- 19.7. Answers to check you progress – possible one
- 19.8. Suggested readings - OER
- 19.9. Reference
- 19.10. Exercise

19.1. INTRODUCTION

In previous units, we explored the physical components of a digital system: combinational logic (which processes data) and sequential logic (which stores data). However, possessing the physical components is not enough to create a functional computer. If you have an Arithmetic Logic Unit (ALU), a memory bank, and a set of registers, how does the system know *when* to move data from memory to the ALU, and *when* to store the result back into a register?

A digital system requires a strict conductor to prevent total data chaos. This conductor is composed of two interdependent PARTs: the **Timing Circuit** and the **Control Circuit**.

The timing circuit acts as the system's metronome or heartbeat, generating the continuous, precise clock pulses that synchronize every flip-flop on the motherboard. The control circuit acts as the system's brain. It listens to the clock, decodes the current software instructions, and generates the exact sequence of electrical signals required to open and close pathways, direct multiplexers, and trigger memory writes. In this unit, we will dissect the hardware that generates this vital

NOTES

synchronization and control, exploring how simple square waves dictate the complex flow of digital information.

19.2. UNIT OBJECTIVES:

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

- Define the fundamental role of clocks and timing circuits in synchronous digital systems.
- Differentiate between various oscillator designs, including RC timers and quartz crystal oscillators.
- Explain the function and architecture of digital sequencers.
- Compare and contrast hardwired control units with microprogrammed control units.
- Analyze real-world design examples of control circuits, such as traffic light controllers and CPU instruction cycles.

19.3. DESIGN OF CLOCKS AND TIMING CIRCUITS

In a synchronous digital system, every state change is driven by a master clock. A clock signal is a continuous square wave that oscillates between a logical LOW (0V) and a logical HIGH (e.g., 5V).

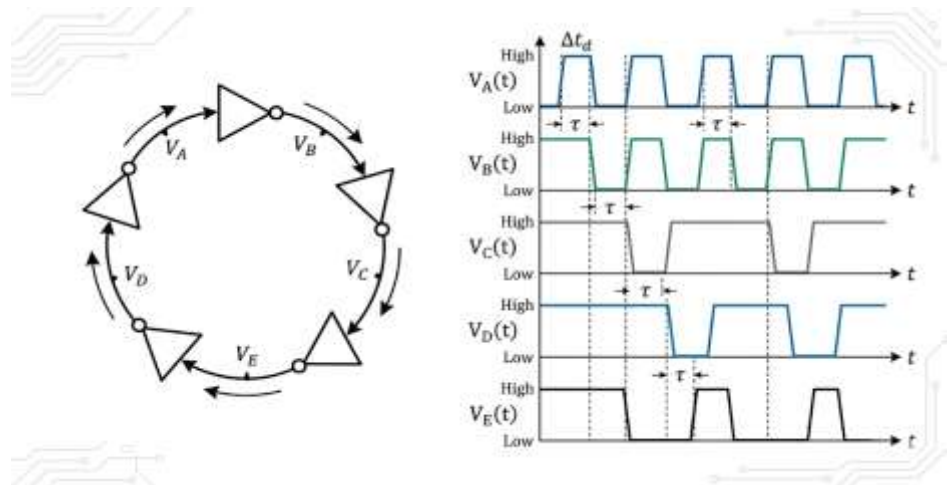


Figure 19.1 : Ring Oscillator Circuit and Waveform.

Key Timing Metrics:

- **Frequency f :** The number of clock cycles that occur in one second, measured in Hertz (Hz). A 3 GHz CPU executes three billion cycles per second.
- **Period T :** The physical time duration of one complete clock cycle. It is the mathematical inverse of frequency ($T = 1/f$).
- **Duty Cycle:** The percentage of one period in which the signal is HIGH. Most digital clocks require a clean 50% duty cycle.

Oscillator Technologies

To generate these continuous pulses, engineers design oscillator circuits. The two most common types in digital design are:

1. **RC Oscillators (e.g., the 555 Timer):** These circuits use Resistor-Capacitor (RC) networks to dictate the timing. The time it takes for a capacitor to charge and discharge sets the clock frequency. While highly adjustable and inexpensive, RC oscillators are vulnerable to temperature fluctuations and are generally not precise enough for high-speed CPUs.
2. **Quartz Crystal Oscillators:** For precise digital systems, piezoelectric quartz crystals are used. When a voltage is applied to a quartz crystal, it vibrates at a highly specific, mechanically determined frequency. These oscillations are converted into a perfectly stable square wave, providing the ultra-precise gigahertz frequencies required by modern processors.

Clock Skew and Distribution

As digital systems grow larger, the physical length of the wires becomes a problem. If the clock signal takes slightly longer to reach a flip-flop on the far left of the board compared to one on the right, the flip-flops will trigger out of sync. This dangerous delay is called **Clock Skew**. Engineers solve this by designing "clock trees"—perfectly balanced, symmetrical wiring networks equipped with signal buffers to ensure the clock edge hits every component simultaneously.

☒ Check Your Progress 19.1.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. What is the mathematical relationship between a clock's frequency and its period?

.....

2. Why are quartz crystal oscillators preferred over RC oscillators in modern computer processors?

.....

3. Define the term "Clock Skew" in a digital system.

.....

19.3.1 Applications in Sequencers

While the master clock provides a steady stream of identical pulses, a system often needs a specific sequence of pulses to perform a complex action. This is where a **Sequencer** is used.

NOTES

A sequencer is a specialized sequential circuit (often constructed using a modified shift register or a counter attached to a decoder) that takes the main clock signal and breaks it down into a multi-phase timing sequence.

For example, a machine might require a strict three-step process: (1) Open a valve, (2) Inject fluid, (3) Close the valve. The sequencer takes the raw, repetitive clock pulses and generates three distinct, overlapping control signals (\$T_1\$, \$T_2\$, and \$T_3\$) that turn on and off in a guaranteed, repeatable loop. In microprocessors, sequencers are vital for executing the distinct phases of an instruction cycle: Fetch, Decode, and Execute.

19.4. OVERVIEW OF CONTROL CIRCUITS

If the timing circuit provides the pulse, the **Control Unit (CU)** makes the decisions. The control circuit is the master orchestrator of the datapath. Its primary job is to read an input (like a machine code instruction), check the current system flags (like checking if the previous math operation resulted in a zero), and then output a massive array of parallel control signals.

These control signals physically enable specific buffers, configure multiplexers, tell the ALU which mathematical operation to perform, and trigger the write-enable pins on memory banks. There are two primary architectural philosophies for designing control circuits:

1. Hardwired Control

In a hardwired control unit, the logic is implemented entirely using fixed physical hardware: discrete AND, OR, and NOT gates combined with flip-flops.

- **Advantages :** It is blazingly fast because the control signals are generated directly by hardware propagation.
- **Disadvantages :** It is incredibly complex to design. Furthermore, it is completely inflexible; if there is a bug in the instruction logic or an update is needed, the entire silicon chip must be physically redesigned and remanufactured. Modern RISC (Reduced Instruction Set Computer) processors often utilize hardwired control for speed.

2. Microprogrammed Control

Instead of using complex webs of logic gates, a microprogrammed control unit uses a specialized internal ROM (Read-Only Memory). The control signals required for each instruction are stored as "words" (microinstructions) inside this memory. When the CPU receives an instruction, it essentially looks up the address in the ROM and reads out the pre-recorded control signals.

- **Advantages:** It is highly structured, easy to debug, and incredibly flexible. If you want to change how the processor executes a command, you simply update the ROM code without changing any physical wiring.

- **Disadvantages:** It is slower than hardwired control because reading from memory takes more clock cycles than passing a signal through basic logic gates. Older CISC (Complex Instruction Set Computer) architectures heavily utilized microprogrammed control.

NOTES**✓ Check Your Progress 19.2.**

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. What is the primary function of a Control Unit in a digital system?
.....
.....
.....
2. What are the two primary architectures for designing control circuits?
.....
.....
.....
3. What is the main advantage of a microprogrammed control unit over a hardwired one?
.....
.....
.....

19.4.1. Design Examples

To understand how timing and control merge, let us look at two practical design examples.

Example 1: Traffic Light Controller (Hardwired FSM)

A basic intersection traffic light is controlled by a hardwired sequencer and state machine. The timing circuit (a simple RC oscillator) provides a 1 Hz pulse. The control circuit utilizes a counter to track the seconds. When the counter reaches 30 seconds in the "Green" state, the control logic triggers a state change to "Yellow." The control circuit physically outputs 5V on the yellow lamp wire and 0V on the green lamp wire, then resets the counter to wait for 5 seconds before switching to "Red."

Example 2: CPU Memory Read Cycle

When a CPU needs to read data from RAM, the control circuit must orchestrate a precise sequence of events.

1. **Clock Cycle 1 T_1 :** The control unit asserts the Address Latch Enable signal, telling the system that the address on the bus is valid.

NOTES

2. **Clock Cycle 2 T_2 :** The control unit asserts the Memory Read signal, instructing the RAM chip to fetch the requested data and place it on the data bus.
3. **Clock Cycle 3 T_3 :** The control unit triggers the Write Enable pin of an internal CPU register, latching the incoming data off the bus.

Without the timing circuit enforcing the T_1 , T_2 , T_3 progression, and the control circuit firing the specific pins, the CPU would crash instantly.



Activity 19.2. Quick Sequencer Sketch

Objective : To visually demonstrate how a sequencer divides a continuous master clock into separate, safe control steps.

Description :

Imagine your computer needs to do a 2-step task: T1 Read Memory, then T2 Write Data. You will sketch the signals to ensure these steps never overlap.

1. **The Master Clock (CLK) :** Draw a simple, repeating square wave (up, flat, down, flat) across your paper to represent 3 complete clock cycles.
2. **Phase 1 T1 :** Directly underneath the clock, draw a second line. Make this line jump HIGH (up) at the start of the first clock cycle, and drop LOW (down) the moment the first cycle ends. Keep it flat/LOW for the rest of the drawing.
3. **Phase 2 T2 :** Underneath T1, draw a third line. Keep it flat/LOW during the first cycle. When the second clock cycle begins, jump it HIGH. Drop it LOW when the second cycle ends.

Reflection :

Look at the T1 and T2 lines. Even though the master clock pulses continuously, you have created a clean "staircase" pattern. You just visually mapped how a Control Unit ensures Step 1 completes entirely before Step 2 begins, guaranteeing that data never collides!

19.5. SUMMARY

In this unit, we examined the critical synchronization and management hardware of digital systems: Timing and Control Circuits. We established that the clock acts as the foundational heartbeat, generated by highly stable quartz crystal oscillators to prevent timing errors like clock skew. We explored how sequencers divide these raw pulses into multiphase operational steps. Transitioning to the system's "brain," we defined the Control Unit, contrasting the blazing speed of rigid Hardwired Control logic against the high flexibility of ROM-based Microprogrammed Control. Finally, we analyzed real-world design applications, demonstrating how precise sequential timing and targeted control signals are strictly required to orchestrate tasks ranging from traffic light coordination to executing complex microprocessor memory cycles.

19.6. GLOSSARY

- **Clock Skew:** A timing phenomenon where a single clock signal arrives at different components at slightly different times due to the physical length of the wiring, potentially causing data errors.
- **Duty Cycle:** The ratio of the time a clock signal is in the active HIGH state compared to the total period of the clock cycle.
- **Hardwired Control:** A control unit architecture built completely out of fixed physical logic gates, prioritizing speed over flexibility.
- **Microprogrammed Control:** A control unit architecture that stores sequences of control signals as microinstructions within an internal ROM, prioritizing flexibility over raw speed.
- **Oscillator:** An electronic circuit that produces a periodic, oscillating electronic signal, typically a square wave used for digital clocking.
- **Sequencer:** A sequential logic circuit that steps through a predetermined sequence of distinct timing states.

19.7 ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 19.1

1. They are inversely proportional. The period is the mathematical inverse of the frequency ($T = 1/f$).
2. Quartz crystals vibrate at highly specific mechanical frequencies, providing the extreme stability and precision required for gigahertz computer processors, whereas RC oscillators are prone to drift due to temperature and component variations.
3. Clock skew is the timing delay that occurs when a clock pulse reaches different flip-flops at slightly different times due to unequal wire lengths across the circuit board.

Check Your Progress 19.2

1. The primary function is to direct the operation of the datapath by decoding instructions and generating the exact sequence of electrical signals necessary to route data and trigger memory/ALU operations.
2. Hardwired control and Microprogrammed control.
3. Flexibility and ease of modification. If a design flaw is found or an instruction needs updating, the microcode in the ROM can be rewritten without physically altering the logic gates on the silicon chip.

NOTES

NOTES

19.8. SUGGESTED READINGS - OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectonline.com/User/UnitUrls?unitId=7138	

19.9. REFERENCE

1. Floyd, T. L. (2015). *Digital Fundamentals*. Pearson Education.
2. Mano, M. M., & Ciletti, M. D. (2013). *Digital Design: With an Introduction to the Verilog HDL*. Pearson.
3. Tocci, R. J., Widmer, N. S., & Moss, G. L. (2017). *Digital Systems: Principles and Applications*. Pearson.

19.10. EXERCISE

PART 1

A. Multiple choice question

1. Which oscillator type is most appropriate for generating the ultra-precise, high-frequency master clock of a modern microprocessor?
 - (a) 555 Timer in astable mode
 - (b) RC (Resistor-Capacitor) network
 - (c) Quartz Crystal Oscillator
 - (d) SR Latch feedback loop
2. A clock signal has a frequency of 2 GHz. What is the period (T) of one complete clock cycle?
 - (a) 0.5 nanoseconds
 - (b) 2 nanoseconds
 - (c) 5 nanoseconds
 - (d) 500 picoseconds
3. The phenomenon where a clock signal reaches different PARTs of a motherboard at slightly different times, potentially causing timing violations, is known as:
 - (a) Clock Jitter
 - (b) Clock Skew
 - (c) Metastability
 - (d) Race Condition
4. Which of the following is a primary characteristic of a Hardwired Control Unit?
 - (a) It uses a programmable ROM to store control words.

- (b) It is exceptionally easy to modify after the chip is manufacture(d)
 - (c) It is constructed using fixed discrete logic gates and is designed for maximum speed
 - (d) It executes microinstructions.
5. In microprogrammed control, the control signals required to execute a specific CPU instruction are stored where?
- (a) Inside the main system Hard Drive.
 - (b) Inside a dedicated read-only memory (ROM) within the CPU.
 - (c) Inside the Arithmetic Logic Unit (ALU).
 - (d) Inside the Quartz Crystal.

Answers with Explanations :

1. **Answer : (c)**

Explanation : Piezoelectric quartz crystals vibrate at exact, mechanically determined frequencies, providing the supreme stability required by modern CPUs, unlike RC networks which suffer from thermal drift.

2. **Answer : (a)**

Explanation : Period is the inverse of frequency.

$$\frac{1}{(2 \times 10^9 \text{ Hz})} = 0.5 \times 10^{-9} \text{ seconds}$$

which is 0.5 nanoseconds (or 500 picoseconds). Options A and D are mathematically identical in reality, but A represents the standard decimal conversion in nanoseconds. (Note: Both 0.5 ns and 500 ps represent the same value; standard engineering convention widely accepts 0.5 ns for gigahertz period calculations).

3. **Answer : (b)**

Explanation : Clock Skew refers strictly to the spatial variation in arrival times of a clock edge at different spatial locations on a circuit board due to trace length inequalities.

4. **Answer : (c)**

Explanation : Hardwired controllers are built using physical networks of logic gates (AND, OR, NOT). Because there is no memory lookup delay, they are incredibly fast, but completely rigid

5. **Answer : (b)**

Explanation : Microprogrammed architectures abstract control logic into software (microcode), storing the necessary arrays of control 1s and 0s directly inside a specialized internal CPU ROM.

NOTES

B. Short answer type question

1. Define the "Duty Cycle" of a clock signal.
2. What is the specific engineering purpose of designing a "Clock Tree" on a motherboard?
3. Briefly explain the function of a sequencer circuit.
4. Why is a Microprogrammed Control Unit generally slower than a Hardwired Control Unit?
5. Identify two specific control signals that a Control Unit might send to the datapath during a memory operation.

PART 2

C. Long answer type question (200 words)

1. Describe the structural and operational differences between RC (Resistor-Capacitor) oscillators and Quartz Crystal oscillators. Evaluate why one is preferred for simple hobbyist circuits (like blinking an LED) while the other is strictly required for computer processing.
2. Analyze the role of a digital sequencer in a microprocessor's instruction cycle. Explain how a continuous, identical stream of master clock pulses is translated into the discrete, multi-phase steps required to fetch and decode an instruction.

PART 3

D. Long answer type question (300 words)

1. Provide a comprehensive comparative analysis of Hardwired Control vs. Microprogrammed Control architectures. In your evaluation, discuss the physical hardware differences, the speed of execution, the ease of debugging and modification, and the types of computer architectures (e.g., RISC vs. CISC) that typically employ each design philosophy.
2. Deconstruct the synchronization challenges inherent in large-scale printed circuit boards (PCBs). Detail the causes and catastrophic effects of "Clock Skew" in a high-speed synchronous system. Furthermore, explain the specific hardware design techniques and routing geometries engineers employ to mitigate these timing anomalies to ensure all flip-flops sample data simultaneously.

UNIT – 20

MICROPROCESSOR BASICS

NOTES

UNIT STRUCTURE

- 20.1. Introduction
- 20.2. Objective
- 20.3. Introduction to Microprocessors
 - 20.3.1. Basic Architecture and Functions
- 20.4. Role of Microprocessors in Digital Systems
- 20.5. Summary
- 20.6. Glossary
- 20.7. Answers to check you progress – possible one
- 20.8. Suggested readings - OER
- 20.9. Reference
- 20.10. Exercise

20.1. INTRODUCTION

As we progress through this course on digital electronics, we have spent a considerable amount of time investigating discrete components. In our earlier units, we meticulously constructed simple combinational logic circuits like adders, multiplexers, and decoders. We then advanced to sequential circuits, exploring how memory is created using flip-flops, and how timing is orchestrated using counters and shift registers. Most recently, we discussed the critical importance of control units and timing circuits to synchronize the flow of data.

Now, I want you to pause and imagine trying to build a modern computer using those discrete components. Decades ago, early computers like the ENIAC or even early minicomputers required massive cabinets filled with thousands of individual vacuum tubes or discrete transistor logic chips. If you wanted the computer to perform a different task, engineers literally had to unplug cables and physically rewire the machine. This hardwired logic was incredibly fast but phenomenally rigid, expensive, and space-consuming. The digital world needed a revolution. That revolution arrived in the 1970s with the invention of the integrated circuit and, specifically, the microprocessor.

A microprocessor takes all the essential components we have studied—the arithmetic logic unit (ALU), the sequential registers, and the complex

NOTES

control unit—and shrinks them down onto a single, microscopic piece of silicon. Instead of rewiring the hardware to change the machine's function, we can now simply change a set of binary instructions stored in memory. The hardware remains completely fixed, while the software dictates the machine's behavior. This shift from hardwired physical logic to programmable software logic is the single most important leap in the history of electronics. In this unit, we will explore what makes a microprocessor tick. We will look under the hood at its internal architecture, understand how it communicates with the outside world using electrical buses, and examine why it is the undisputed "brain" of virtually every modern digital system.

20.2. OBJECTIVES:

After successfully completing this unit, you will be able to:

- **Define** the fundamental concept of a microprocessor and its evolution from discrete logic circuits.
- **Analyze** the internal architecture of a standard microprocessor, detailing the roles of the ALU, Control Unit, and Registers.
- **Explain** the function of the system bus system, differentiating between the address, data, and control buses.
- **Differentiate** between a general-purpose microprocessor and a dedicated microcontroller.
- **Evaluate** the role of microprocessors in modern digital systems and embedded applications.

20.3. INTRODUCTION TO MICROPROCESSORS

A microprocessor is an electronic component that serves as the central processing unit (CPU) of a computer or digital system, housed on a single integrated circuit (IC). It is a multipurpose, clock-driven, register-based, programmable device. What does this mean for you and me? It means the device accepts digital data as input, processes it according to a sequence of instructions stored in its memory, and provides results as output.

The first commercial microprocessor was the Intel 4004, introduced in 1971. It was a simple 4-bit processor designed initially for a calculator. However, engineers quickly realized its potential. Because it was programmable, the same chip used in a calculator could be used to control a traffic light or a medical device, simply by writing a different program. Today, microprocessors process 64 bits of data simultaneously and operate at gigahertz frequencies, but the fundamental underlying logic remains exactly the same as that of the Intel 4004.

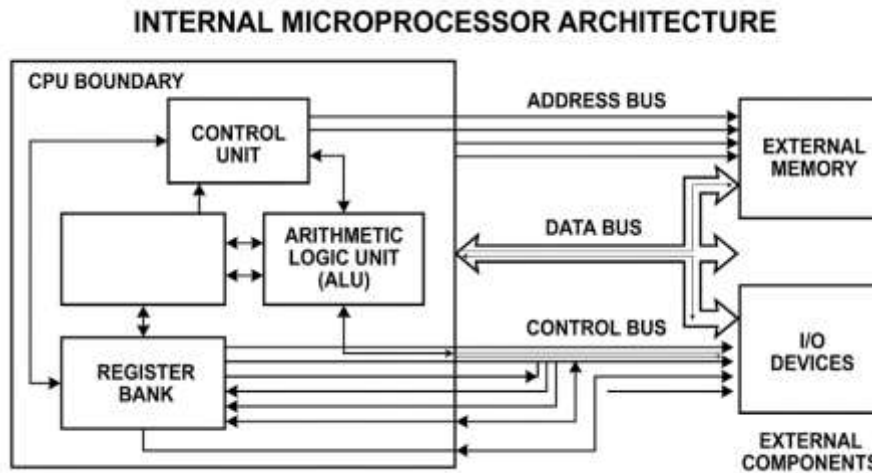


Figure 20.1 : Internal Microprocessor Architecture.

☒ Check Your Progress 20.1.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. What does the acronym CPU stand for, and how does it relate to a microprocessor?

.....

2. What was the name and manufacturer of the first commercial microprocessor?

.....

3. State the primary advantage of a microprocessor over hardwired digital logic.

.....

20.3.1. Basic Architecture and Functions

To understand how a microprocessor executes a program, we must examine its internal structure. A standard microprocessor consists of three primary functional blocks that work in perfect harmony.

1. **The Arithmetic Logic Unit (ALU)** The ALU is the digital system's mathematical engine. It contains the combinational logic circuits necessary to perform arithmetic operations (addition, subtraction) and logical operations (AND, OR, NOT, XOR). Whenever the

NOTES

software asks the processor to compare two numbers or perform a calculation, the data is routed through the ALU.

2. The Register Array The ALU cannot fetch data directly from a slow external hard drive; doing so would bottleneck the entire system. Instead, the microprocessor contains a small bank of ultra-fast internal memory cells called registers.

- **The Accumulator:** This is the most critical register. It holds the initial operand before an ALU operation and captures the final result immediately after the operation.
- **Program Counter (PC):** A dedicated sequential counter that holds the exact memory address of the *next* instruction to be fetched.
- **Instruction Register (IR):** Once an instruction is fetched from memory, it temporarily sits in the IR so the processor can decode it.
- **General Purpose Registers:** Additional registers (often labeled B, C, D, E, etc.) used to temporarily hold data during calculations.

3. The Control Unit (CU) The Control Unit is the brain within the brain. It does not process data; rather, it directs the flow of data. It reads the binary instruction from the Instruction Register, decodes its meaning, and generates the precise timing and control signals needed to execute the command.

- **The System Bus** For the microprocessor to be useful, it must communicate with external memory (RAM/ROM) and Input/Output (I/O) devices. It does this via the System Bus—a physical set of parallel electrical wires divided into three distinct channels:
- **Address Bus:** A unidirectional bus that carries the memory location the CPU wants to access.
- **Data Bus:** A bidirectional bus that physically carries the binary data back and forth between the CPU and memory.
- **Control Bus:** A set of individual wires carrying command signals, such as "Memory Read" (telling the RAM to output data) or "Memory Write" (telling the RAM to accept data).



Activity 20.2. Mapping the System Bus

Objective :

To visually understand how the Address, Data, and Control buses facilitate a memory read operation.

Description :

Draw a box labeled "Microprocessor" on the left and a box labeled "RAM" on the right. Draw a single arrow pointing from the Microprocessor to the RAM and label it "Address Bus: 0x00A1". Draw a second line from the Microprocessor to the RAM, but place a switch on it labeled "Control Bus: Read Enable". Finally, draw an arrow pointing from the RAM back to the Microprocessor labeled "Data Bus: 10110011".

Reflection :

This drawing illustrates the precise sequence: The CPU points to a location, sends the control command to read, and only then does the RAM send the data back along the bidirectional data path.

NOTES

20.4. ROLE OF MICROPROCESSORS IN DIGITAL SYSTEMS

Microprocessors have completely bifurcated the field of digital system design into two main categories: General-Purpose Computing and Embedded Systems.

1. **General-Purpose Digital Systems** : When we use a desktop computer, a laptop, or a smartphone, we are using a general-purpose system. In these architectures, the microprocessor is designed to be highly versatile. It interacts with massive amounts of external RAM and storage drives. It is designed to run complex operating systems (like Windows, Linux, or Android) and can execute completely unpredictable user applications—ranging from rendering high-definition video games to calculating massive spreadsheets. These microprocessors emphasize maximum processing power and speed.
2. **Embedded Digital Systems** : An embedded system is a computer hidden inside a larger mechanical or electrical device. It is dedicated to performing exactly one job flawlessly. Examples include the processor inside an automatic washing machine, a digital thermostat, an automobile's anti-lock braking system (ABS), or an industrial robotic arm.

In these applications, you do not need the raw power of a desktop CPU. Instead, engineers use a specific variant of the microprocessor called a **Microcontroller**. A microcontroller is essentially a small computer on a single chip. While a standard microprocessor requires separate, external chips for memory and I/O ports, a microcontroller integrates

NOTES

the CPU, a small amount of RAM, a small amount of ROM (to hold the dedicated program), and the I/O peripherals all onto one single piece of silicon. This makes embedded systems incredibly cost-effective, space-efficient, and reliable.

☒ **Check Your Progress 20.2.**

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. What are the three components of the System Bus?

.....

.....

.....

2. Briefly explain the difference between a Microprocessor and a Microcontroller.

.....

.....

.....

3. Give two real-world examples of an embedded system.

.....

.....

.....

20.5. SUMMARY

A **Microprocessor** is a programmable integrated circuit that combines the functions of a central processing unit (CPU) onto a single silicon chip, revolutionizing digital design by shifting focus from hardwired logic to software-driven functionality.

The internal architecture is comprised of three core components: The **Arithmetic Logic Unit (ALU)** which performs mathematical and logical operations; the **Register Array** (including the Accumulator and Program Counter) which provides high-speed, temporary internal data storage; and the **Control Unit**, which decodes instructions and orchestrates system timing.

Microprocessors communicate with external memory and peripherals via the **System Bus**, which is logically PARTitioned into a unidirectional **Address Bus**, a bidirectional **Data Bus**, and a **Control Bus**.

The microprocessor fundamentally operates on a continuous, clock-driven **Fetch-Decode-Execute** cycle, retrieving instructions from memory, translating them into hardware actions, and executing the logic.

In modern digital systems, microprocessors are utilized in powerful **General-Purpose Computers**, whereas highly integrated **Microcontrollers**—which combine a CPU, memory, and I/O onto one chip—are the standard for single-purpose **Embedded Systems**.

20.6 GLOSSARY

- **Accumulator:** The primary internal register in a microprocessor used to hold operands and capture the results of ALU operations.
- **Address Bus:** A unidirectional set of parallel wires used by the microprocessor to specify the physical memory location it wishes to access.
- **Arithmetic Logic Unit (ALU):** The combinational digital circuit within the microprocessor responsible for executing mathematical calculations and boolean logic.
- **Control Unit (CU):** The internal component that directs the operation of the processor by decoding instructions and generating specific electrical timing signals.
- **Data Bus:** A bidirectional communication pathway used to transfer the actual binary data between the CPU, memory, and I/O devices.
- **Embedded System:** A dedicated computer system hidden inside a larger device, designed to perform a specific, dedicated function (e.g., a microwave timer).
- **Microcontroller:** A highly integrated circuit that contains a processing core, memory, and programmable input/output peripherals on a single chip.
- **Microprocessor:** An integrated circuit containing all the functions of a CPU, serving as the programmable "brain" of a digital system.
- **Program Counter (PC):** A specialized internal register that always holds the memory address of the next instruction to be fetched and executed.

20.7. ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 20.1

1. CPU stands for Central Processing Unit. A microprocessor is a CPU built onto a single integrated circuit chip.
2. The Intel 4004.
3. Flexibility. A microprocessor allows the system's function to be completely changed simply by rewriting the software program, without needing to physically rewire any logic gates.

Check Your Progress 20.2:

1. The Address Bus, the Data Bus, and the Control Bus.
2. A microprocessor only contains the CPU core and requires external chips for RAM, ROM, and I/O. A microcontroller contains the CPU, RAM, ROM, and I/O all integrated together onto a single integrated circuit.

NOTES

3. A digital thermostat and an automobile's anti-lock braking system (ABS).

20.8. SUGGESTED READINGS - OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectonline.com/User/UnitUrls?unitId=7139	

20.9. REFERENCE

1. Gaonkar, R. S. (2013). Microprocessor Architecture, Programming, and Applications with the 8085 (6th ed.). Penram International Publishing.
2. Godse, A. P., & Godse, D. A. (2020). *Microprocessors & Microcontrollers*. Technical Publications.
3. Jain, R. P. (2003). *Modern Digital Electronics* (4th ed.). Tata McGraw-Hill Education.
4. Mano, M. M., & Ciletti, M. D. (2013). *Digital Design: With an Introduction to the Verilog HDL* (5th ed.). Pearson.
5. Tocci, R. J., Widmer, N. S., & Moss, G. L. (2017). *Digital Systems: Principles and Applications* (12th ed.). Pearson.
6. Floyd, T. L. (2015). *Digital Fundamentals* (11th ed.). Pearson Education.

20.10. EXERCISE

PART 1

A. Multiple choice question

1. Which internal component is responsible for holding the memory address of the next instruction to be executed?
 - (a) Accumulator
 - (b) Program Counter (PC)
 - (c) Instruction Register (IR)
 - (d) Arithmetic Logic Unit (ALU)
2. Which of the following best describes the Data Bus in a microprocessor system?
 - (a) It is unidirectional, flowing only into the CPU.
 - (b) It is unidirectional, flowing only out of the CPU.
 - (c) It is bidirectional, allowing data to be both read from and written to memory.
 - (d) It only carries clock signals.

3. The transition from hardwired digital logic to microprocessor-based design primarily provided engineers with an increase in:
 - (a) Hardware flexibility and programmability
 - (b) Physical wire size
 - (c) Power consumption
 - (d) System rigidity
4. If you were designing an embedded system for a digital microwave oven, which component would be the most efficient choice?
 - (a) A 64-bit general-purpose Microprocessor
 - (b) A complex Application Specific Integrated Circuit (ASIC)
 - (c) A Microcontroller
 - (d) Discrete 7400-series logic gates
5. What happens during the "Decode" phase of the microprocessor's execution cycle?
 - (a) The ALU performs mathematical addition.
 - (b) The result is written back to the hard drive.
 - (c) The instruction is fetched from external RAM.
 - (d) The Control Unit translates the binary instruction into hardware action signals.

Answers with explanation :

1. Answer : (b)

Explanation: The Program Counter is a sequential counter whose sole job is to point to the next instruction in memory, ensuring the program executes in the correct order.

2. Answer : (c)

Explanation: The Data Bus must be bidirectional because the processor needs to fetch (read) variables from RAM and also save (write) the calculated results back to RAM.

3. Answer : (a)

Explanation : By utilizing software instructions stored in memory, engineers gained the flexibility to change a system's function without physically redesigning the electronic circuits.

4. Answer : (c)

Explanation : A microcontroller contains the CPU, limited memory, and I/O ports all on one chip, making it highly cost-effective and perfectly suited for simple, dedicated embedded tasks.

5. Answer : (d)

Explanation : After the instruction is fetched, it sits in the Instruction Register where the Control Unit interprets its meaning and prepares the necessary electrical signals to execute it.

NOTES

B. Short answer type question

1. Define the term "Microprocessor" in the context of digital electronics.
2. List the three fundamental components of a system bus.
3. Briefly explain the function of the Accumulator register.
4. Why is the Address Bus unidirectional?
5. State one major difference between general-purpose computing systems and embedded systems.

PART 2

C. Long answer type question (200 words)

1. Describe the three core internal components of a microprocessor (ALU, Register Array, and Control Unit). How do these three blocks interact to process a single mathematical command?
2. Compare and contrast a Microprocessor and a Microcontroller. In your response, address the integration of memory, input/output peripherals, and specific use cases for each technology.
3. Trace the steps of a typical CPU "Memory Read" operation. Detail exactly how the Address, Data, and Control buses are utilized during this specific cycle.

PART 3

D. Long answer type question (300 words)

1. Evaluate the historical shift from hardwired discrete logic circuits to programmable microprocessor architectures. Discuss the engineering limitations of early hardwired systems, and explain how the introduction of the microprocessor drastically reduced manufacturing costs, minimized physical space requirements, and increased the functional flexibility of digital devices.
2. Provide a comprehensive analysis of the microprocessor's internal Register Array. Deconstruct the specific functions of the Accumulator, the Program Counter (PC), the Instruction Register (IR), and General-Purpose registers. Why are these ultra-fast internal memory cells strictly necessary, and how would relying solely on external RAM bottleneck the system's processing speed?
3. Imagine you are tasked with designing the control system for an automated industrial robotic arm. Argue why an embedded microcontroller approach is superior to using a high-end, general-purpose desktop microprocessor. Frame your answer around hardware costs, physical integration, real-time operating stability, and the concept of dedicated system functionality.

UNIT – 21

DIGITAL SYSTEM DESIGN

NOTES

UNIT STRUCTURE

- 21.1. Introduction
- 21.2. Objective
- 21.3. Basics of Digital System Design
 - 21.3.1. System-Level Approaches
- 21.4. Integration of Digital Circuits into Systems
- 21.5. Summary
- 21.6. Glossary
- 21.7. Answers to check you progress – possible one
- 21.8. Suggested readings - OER
- 21.9. Reference
- 21.10. Exercise

21.1. INTRODUCTION

We have traversed a remarkable path through the landscape of digital electronics. In our earliest sessions, we focused our microscopes on the fundamental building blocks: the simple AND, OR, and NOT logic gates. We learned how to weave these gates into combinational circuits that make instantaneous decisions, and later, how to trap data in time using flip-flops and sequential memory circuits. Most recently, we explored microprocessors, revealing how these components can be miniaturized onto a single silicon chip to execute programmable software.

However, understanding how a single transistor or a single microprocessor works is vastly different from engineering a complete, functioning technological product. Consider the smartphone in your pocket or the navigation system in a modern aircraft. These are not just collections of logic gates; they are highly complex, synchronized architectures where processors, memory banks, signal converters, and hardware accelerators must communicate flawlessly at billions of cycles per second. Up to this point, we have been studying the individual instruments of an orchestra. Now, it is time to learn how to conduct the entire symphony.

In this unit, we shift our perspective from the microscopic level of logic design to the macroscopic realm of **Digital System Design**. We will explore the rigorous methodologies engineers use to tame overwhelming

NOTES

hardware complexity. You will learn how we abstract massive circuits into system-level blocks, allowing us to describe hardware behavior mathematically before we ever solder a wire. We will delve deeply into the integration process—the physical and logical challenges of binding disparate digital circuits into one harmonious system. By the end of this module, you will transition from thinking like a circuit builder to thinking like a true systems architect, equipped to conceptualize and integrate large-scale digital machines.

21.2. OBJECTIVES

After successfully completing this unit, you will be able to:

- Analyze the foundational principles of digital system design, distinguishing between isolated logic circuits and integrated algorithmic systems.
- Evaluate different system-level approaches, specifically comparing the top-down and bottom-up design methodologies.
- Design abstract representations of digital hardware using Register Transfer Level (RTL) notation and Algorithmic State Machine (ASM) charts.
- Integrate distinct digital components—such as datapaths, control units, memory blocks, and I/O interfaces—into a cohesive, synchronized digital system.
- Formulate strategies to mitigate physical integration challenges, including clock skew, power dissipation, and signal integrity in high-speed systems.

21.3. BASICS OF DIGITAL SYSTEM DESIGN

When we discuss a "Digital System," we are referring to an interconnected arrangement of digital modules that work together to execute a specific algorithm or solve a complex problem. A digital system is defined by its ability to take input data, process that data through a series of discrete logical and mathematical operations, and produce a predictable, stable output.

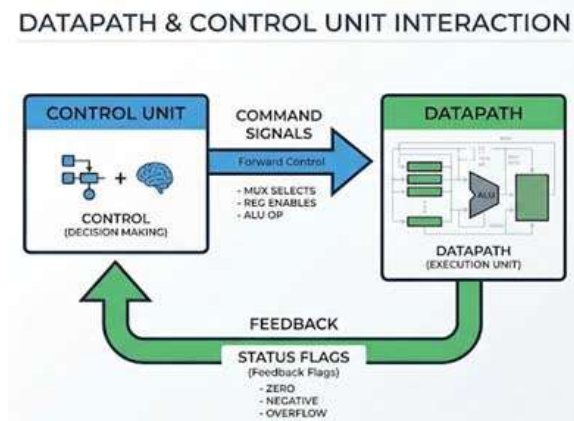


Figure 21.1 : Datapath and Control Unit Interaction

The Anatomy of a Digital System : To design such a system, we must fundamentally divide the architecture into two primary domains that interact continuously: the **Datapath** and the **Control Unit**.

1. **The Datapath (The Muscle):** The datapath is the physical highway where actual data processing occurs. It consists of interconnected processing elements like Arithmetic Logic Units (ALUs) for math, shift registers for data manipulation, multiplexers for routing, and memory registers for temporary storage. The datapath is considered "dumb"—it contains all the hardware necessary to perform incredible calculations, but it has no idea *when* to execute them or *where* to route the answers.
2. **The Control Unit (The Brain):** The control unit is a highly complex Finite State Machine (FSM). It does not perform math on the data. Instead, it reads the current status of the system and generates precise electrical signals (Control Words) that dictate the actions of the datapath. For example, the control unit will send a logical HIGH to a multiplexer's select pin, wait exactly one clock cycle, and then send a logical HIGH to a register's write-enable pin.

Register Transfer Level (RTL) Abstraction When designing the basics of a digital system, engineers cannot afford to draw millions of individual logic gates. It would take lifetimes. Instead, we use an abstraction language called **Register Transfer Level (RTL)**. RTL allows us to describe the behavior of a digital system by defining the flow of data between hardware registers and the logical operations performed during that transfer.

In RTL, we use mathematical notation to dictate physical hardware. For example, consider the expression: IF (Start == 1) THEN $R3 \leftarrow R1 + R2$

To a software engineer, this looks like a line of code. To a digital system designer, this explicitly defines physical hardware:

- It requires three physical hardware registers (R1, R2, R3).
- It requires a combinational binary adder circuit capable of adding the widths of R1 and R2.
- It requires physical copper buses connecting the outputs of R1 and R2 to the adder, and the adder to R3.
- It requires a control unit that checks the Start pin and, when HIGH, triggers the clock enable pin on R3 to latch the sum perfectly on the next rising clock edge.

By designing at the RTL level, we focus on the flow of data and the algorithmic requirements of the system, leaving the tedious gate-level synthesis to automated computer software.

NOTES

✓ Check Your Progress 21.1.

Notes : (a) Space is given below for your answer.
(b) Check your answer with the one given at the end of this unit.

1. What are the two fundamental, interacting domains that make up a digital system architecture?
.....
.....
.....
2. Which component of a digital system is responsible for generating timing signals and deciding when operations occur?
.....
.....
.....
3. What does the acronym RTL stand for in the context of digital hardware design?
.....
.....
.....

21.3.1. System-Level Approaches

To move from a vague product idea to a physical silicon chip, engineers employ rigorous system-level design approaches. Managing the complexity of a system with billions of transistors requires strict methodologies.

Top-Down vs. Bottom-Up Design The foundation of modern digital system design rests on the **Top-Down Methodology**. In a top-down approach, we start by defining the absolute highest level of the system—the "black box" specifications. We ask: What are the primary inputs? What are the desired outputs? What is the maximum power consumption allowed? What is the required clock speed?

Once the black box is defined, we break it down into major functional subsystems. For a digital camera system, this might mean splitting the design into an Image Capture Block, a Compression Block, and a Memory Interface Block. We then take each block and break it down further into RTL datapath and control units, and finally down to the gate and transistor level. This decomposition ensures that every piece of the system perfectly aligns with the original macroscopic goal.

Conversely, the **Bottom-Up Methodology** starts with the lowest-level components (transistors and gates) and connects them to build flip-flops, which are connected to build registers, which are connected to build processors. While useful for building standardized component libraries (like standard cells), attempting to build a modern, complex

system purely from the bottom up is disastrous. It inevitably leads to "integration hell," where subsystems built in isolation fail to communicate properly when connected because the overall system architecture was never planned cohesively.

Hardware Description Languages (HDLs) To execute a system-level approach today, engineers use Hardware Description Languages, primarily **Verilog** and **VHDL**. These are not programming languages like Python or C++; they are hardware modeling languages. When you write Verilog, you are writing text that a computer synthesizer will physically translate into logic gates and wire layouts on a silicon wafer. HDLs allow teams of engineers to simulate a massive digital system, testing its logic and timing in a virtual environment for months before spending millions of dollars to physically manufacture the chip.



Activity 21.1. Conceptualizing Top-Down Decomposition

Objective :

To practice the first step of the top-down system design methodology by functionally decomposing a common digital device.

Description :

Select a digital device you interact with daily, such as a digital microwave oven. On a piece of paper, draw a large box representing the whole system. Now, break this down. Draw arrows pointing into the box for "Inputs" (Keypad, Door Sensor, Temperature Probe). Draw arrows pointing out for "Outputs" (Magnetron Power, LCD Display, Beeper). Finally, inside the large box, draw three smaller functional subsystems that would be required to handle these I/O operations and connect them logically.

Reflection :

You have just performed high-level architectural decomposition. Before designing a single logic gate or timer circuit, you have established the system's structural boundaries, which is the cornerstone of professional digital engineering.

21.4. INTEGRATION OF DIGITAL CIRCUITS INTO SYSTEMS

Designing perfect, isolated subsystems in an HDL simulator is only half the battle. The true test of an engineer's skill comes during **System Integration**—the phase where all these distinct digital circuits are physically and electrically tied together. Integrating high-speed digital

NOTES

circuits introduces severe physical phenomena that can cause catastrophic system failures if not managed properly.

1. **Clock Distribution and Skew** In a synchronous digital system, every flip-flop and register must be triggered by the master clock simultaneously. However, electrical signals take time to travel down a copper wire. If the clock signal wire routing to a component on the left side of the circuit board is shorter than the wire routing to a component on the right side, the clock edge will arrive at different times. This dangerous delay is called **Clock Skew**.

If clock skew is severe enough, the right-side register might still be holding old data when the left-side register tries to read from it, destroying the algorithmic logic. To integrate systems safely, engineers design intricate "Clock Trees"—often taking the shape of an H-tree layout—to ensure that the physical wire length from the master clock generator to *every single component* in the system is exactly the same, down to the micrometer.

2. **Bus Interfacing and Protocols** When integrating different digital blocks (e.g., connecting a microprocessor to a graphics accelerator), they must share data. You cannot run dedicated wires between every single component; the board would be an unroutable mess. Instead, systems are integrated using **Buses**—shared communication pathways.

To prevent two components from trying to talk on the bus at the same time (which causes a short circuit and data corruption known as bus contention), engineers implement strict **Bus Protocols**. A protocol is a set of hardware rules. For example, a system might use an arbiter circuit. Before the graphics unit can send data to memory, it must send a "Request" signal to the arbiter. It must wait until the arbiter sends back a "Grant" signal before it is electrically allowed to drive the bus.

3. **Mixed-Signal Integration (ADC/DAC)** Digital systems rarely exist in a purely digital world. They must interact with analog environments—reading human voices, sensing temperatures, or controlling physical motors. Integrating digital systems into the real world requires **Analog-to-Digital Converters (ADCs)** and **Digital-to-Analog Converters (DACs)**. Integrating these mixed-signal components requires extreme care regarding power and grounding. Digital circuits are incredibly noisy; switching millions of transistors on and off creates voltage spikes. If this digital noise bleeds into the sensitive analog inputs of an ADC, it will corrupt the sensor readings. Therefore, engineers must physically isolate the analog ground planes from the digital ground planes on the Printed Circuit Board (PCB), ensuring clean integration.

✓ Check Your Progress 3.2.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. Which hardware design language is commonly used by engineers to simulate and synthesize digital systems?

- (a) Python
- (b) C++
- (c) Verilog
- (d) HTML

.....

2. The delay difference in the arrival time of a clock signal at different components in a system is known as what?

- (a) Propagation Delay
- (b) Clock Skew
- (c) Duty Cycle
- (d) Bus Contention

.....

3. True or False:

In a Top-Down design methodology, an engineer starts by building the logic gates and flip-flops first, before deciding what the overall system will do.

.....

NOTES**21.5. SUMMARY**

Digital System Design represents the highest level of hardware engineering, shifting focus from isolated logic gates to massive, integrated architectures that execute complex algorithms.

A digital system is architecturally PARTitioned into two main domains: the **Datapath**, which performs mathematical operations and routes data, and the **Control Unit**, which acts as a state machine to generate the precise timing and command signals required to drive the datapath.

NOTES

To manage extreme complexity, the industry relies on the **Top-Down Design Methodology**. This involves defining the macroscopic system specifications first, and systematically decomposing the architecture into functional subsystems, RTL modules, and finally gate-level schematics.

Register Transfer Level (RTL) is a crucial abstraction that allows engineers to describe hardware functionality mathematically, focusing on how data moves between registers on specific clock cycles. Hardware Description Languages (HDLs) like Verilog and VHDL are used to script this RTL for computer simulation and synthesis.

System Integration poses severe physical challenges. Engineers must meticulously design clock distribution networks (like H-trees) to eliminate **Clock Skew**, implement robust bus protocols to manage communication, and physically isolate digital noise when integrating mixed-signal (ADC/DAC) interfaces.

21.6. GLOSSARY

- **Analog-to-Digital Converter (ADC):** A crucial interface circuit that translates continuous real-world analog signals (like voltage or sound waves) into discrete binary numbers that a digital system can process.
- **Bottom-Up Design:** A design methodology that begins at the lowest level of abstraction (transistors or gates) and progressively combines them to build larger systems; rarely used alone in modern complex design due to integration difficulties.
- **Bus Contention:** A hazardous electrical state where two or more digital components attempt to drive an output signal onto a shared communication bus simultaneously, leading to data corruption and potential hardware damage.
- **Clock Skew:** A spatial variation in the arrival time of a clock signal at different components within a synchronous system, primarily caused by differences in physical wire routing lengths.
- **Control Unit:** The central state machine of a digital system that decodes instructions, monitors status flags, and issues the specific control signals required to sequence the datapath.
- **Datapath:** The physical collection of functional hardware units, such as ALUs, multiplexers, and registers, interconnected by buses, designed to perform specific data processing operations.
- **Hardware Description Language (HDL):** A specialized computer language, such as Verilog or VHDL, used to model, simulate, and synthesize the structural and behavioral characteristics of electronic circuits.
- **Register Transfer Level (RTL):** A level of design abstraction that models a synchronous digital circuit in terms of the logical

operations performed on data as it transfers between hardware registers.

- **Top-Down Design:** A systematic engineering methodology that begins with defining the highest-level overall system specifications and progressively decomposing the system into smaller, detailed hardware modules.

21.7. ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 21.1:

1. The Datapath and the Control Unit.
2. The Control Unit.
3. Register Transfer Level.

Check Your Progress 21.2:

1. (c) Verilog
2. (b) Clock Skew
3. False. (This describes Bottom-Up design. Top-Down starts with the overall system behavior before any gates are considered).

21.8. SUGGESTED READINGS – OER

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectionline.com/User/UnitUrls?unitId=7140	

21.9. REFERENCE

1. Kumar, A. (2014). *Fundamentals of Digital Circuits* (4th ed.). PHI Learning Pvt. Ltd.
2. Jain, R. P. (2003). *Modern Digital Electronics* (4th ed.). Tata McGraw-Hill Education.
3. Godse, A. P., & Godse, D. A. (2020). *Digital Electronics and Logic Design*. Technical Publications.
4. Mano, M. M., & Ciletti, M. D. (2013). *Digital Design: With an Introduction to the Verilog HDL* (5th ed.). Pearson.
5. Wakerly, J. F. (2006). *Digital Design: Principles and Practices* (4th ed.). Pearson Prentice Hall.
6. Patterson, D. A., & Hennessy, J. L. (2013). *Computer Organization and Design: The Hardware/Software Interface* (5th ed.). Morgan Kaufmann.
7. Tocci, R. J., Widmer, N. S., & Moss, G. L. (2017). *Digital Systems: Principles and Applications* (12th ed.). Pearson.

21.10. EXERCISE

PART 1**A. Multiple choice question**

1. Which of the following best defines the role of the Datapath in a digital system architecture?
 - (a) It manages the power supply distribution.
 - (b) It generates the high-frequency master clock signals.
 - (c) It performs the actual arithmetic, logical, and routing operations on the data.
 - (d) It acts as a finite state machine to sequence hardware commands.
2. A hardware engineer is writing code in VHDL to describe the flow of data between registers. At which level of abstraction is this engineer operating?
 - (a) Transistor Level
 - (b) Register Transfer Level (RTL)
 - (c) Application Software Level
 - (d) Analog Level
3. Why is the Bottom-Up design methodology generally considered inadequate when used as the sole approach for designing modern, complex digital processors?
 - (a) It requires too much high-level software.
 - (b) Transistors are no longer used in modern electronics.
 - (c) It often leads to severe integration failures because the overall architectural system boundaries were not planned in advance.
 - (d) It cannot be simulated.
4. To solve the problem of Clock Skew across a large printed circuit board, engineers often implement which of the following physical routing geometries?
 - (a) A serial daisy-chain layout
 - (b) An H-Tree clock distribution network
 - (c) A random-walk trace
 - (d) An open-circuit ground plane
5. When integrating digital processing components with real-world sensors (like a temperature probe), what specific interface component is strictly necessary?
 - (a) A microprocessor accumulator
 - (b) An Analog-to-Digital Converter (ADC)
 - (c) A Digital-to-Analog Converter (DAC)

(d) A synchronous counter

NOTES

Answers with explanation :1. **Answer : (c)**

Explanation : The Datapath is the physical "muscle" of the system, comprising the ALUs, registers, and multiplexers required to execute data manipulation. Timing and state sequencing are handled by the Control Unit.

2. **Answer : (b)**

Explanation: Hardware Description Languages like VHDL and Verilog are primarily utilized to model digital circuits at the Register Transfer Level (RTL), dictating how data moves synchronously between memory elements.

3. **Answer : (c)**

Explanation : Building from the gates upward without a macroscopic top-down plan results in disconnected subsystems that frequently suffer from bus incompatibilities and timing failures when finally connected together.

4. **Answer : (b)**

Explanation : An H-Tree network relies on symmetrical branching. This ensures that the physical copper trace length from the central clock generator to every end-point flip-flop is mathematically identical, thereby equalizing signal propagation delay.

5. **Answer : (b)**

Explanation : Real-world sensors produce continuous analog voltages. A digital system only understands binary 1s and 0s. Therefore, an ADC is required to sample the analog wave and convert it into a discrete digital data format.

B. Short answer type question

1. Identify the fundamental difference between a Hardware Description Language (like Verilog) and a standard software programming language (like Python).
2. Briefly explain the concept of "Bus Contention" during system integration.
3. List three functional components that are typically found within a system's Datapath.
4. Describe the interaction between the Datapath and the Control Unit using the concepts of "Status Flags" and "Control Words."
5. Define "Clock Skew" and state why it is dangerous in synchronous digital systems.

NOTES

PART 2**C. Long answer type question (200 words)**

1. Compare and contrast the Top-Down and Bottom-Up methodologies in digital system design. Discuss why the industry standard relies heavily on Top-Down decomposition for system architecture, while still utilizing Bottom-Up principles for standard cell libraries.
2. Analyze the significance of Register Transfer Level (RTL) abstraction. Explain how describing hardware via the movement of data between registers and clock cycles simplifies the design of systems containing millions of logic gates.
3. Discuss the specific challenges engineers face when integrating mixed-signal (analog and digital) components onto a single circuit board. Focus your explanation on the issues of digital switching noise and grounding techniques.

PART 3**D. Long answer type question (300 words)**

1. Provide a comprehensive architectural breakdown of a Digital System. Explain the symbiotic relationship and continuous feedback loop between the Datapath and the Control Unit.
2. Evaluate the physical challenges of System Integration in high-speed digital environments. Detail the catastrophic effects of clock skew and the precise routing strategies used to prevent it.
3. Outline the Top-Down design methodology for a digital vending machine controller. Describe your step-by-step process from high-level block definitions down to the required RTL logic.

UNIT – 22

FUTURE TRENDS IN DIGITAL ELECTRONICS

NOTES

UNIT STRUCTURE

- 22.1. Introduction**
- 22.2 Objective**
- 22.3. Emerging Technologies in Digital Electronics**
 - 22.3.1. Quantum Computing**
 - 22.3.2. Neuromorphic Computing**
- 22.4. Applications in AI and IoT**
- 22.5. Summary**
- 22.6. Glossary**
- 22.7. Answers to check you progress – possible one**
- 22.8. Suggested readings - OER**
- 22.9. Reference**
- 22.10. Exercise**

22.1 INTRODUCTION

Traveled together from the simplest conceptualization of a binary “1” and “0” all the way up to architecting massive, synchronized digital systems containing millions of logic gates. We have studied standard TTL and CMOS logic families, built intricate datapaths, and explored how programmable silicon like FPGAs allows us to manipulate hardware with software.

Throughout this entire course, our underlying assumption has been based on classical physics and the traditional von Neumann architecture: data is stored in memory, fetched by a processor, manipulated using boolean logic, and sent back. For over fifty years, the digital electronics industry has thrived on a simple golden rule known as Moore's Law—the observation that the number of transistors on a microchip doubles roughly every two years. To achieve this, engineers simply made the transistors smaller and smaller.

However, we are now approaching a hard physical wall. As we shrink transistors down to the size of just a few atoms (sub-3 nanometer nodes), classical physics breaks down. Electrons begin to "teleport"

NOTES

across logic gates due to a phenomenon called quantum tunneling, making it incredibly difficult to maintain a reliable "0" or "1". Furthermore, shuttling data back and forth between memory and the CPU is creating a massive energy and heat bottleneck.

If we want to continue advancing human computation, we can no longer just "shrink" things. We must fundamentally rethink what a digital system is. In this concluding unit, we will step into the future. We will explore Quantum Computing, where bits can be both 0 and 1 simultaneously. We will look at Neuromorphic hardware, where engineers are literally building silicon that mimics the biological structure of the human brain. Finally, we will examine how these radical new hardware paradigms are enabling the explosive growth of Artificial Intelligence (AI) and the Internet of Things (IoT). Let us look at what awaits us on the horizon of digital electronics.

22.2. OBJECTIVES

After successfully completing this unit, you will be able to:

- Analyze the physical and thermal limitations of classical silicon-based CMOS scaling as defined by the twilight of Moore's Law.
- Explain the fundamental principles of quantum mechanics, specifically superposition and entanglement, as they apply to quantum computing qubits.
- Evaluate the structural differences between traditional von Neumann architectures and brain-inspired Neuromorphic computing paradigms.
- Synthesize how Spiking Neural Networks (SNNs) and novel components like memristors are revolutionizing energy-efficient hardware.
- Assess the transformative impact of ultra-low-power digital logic and edge-computing hardware on the Internet of Things (IoT) and Artificial Intelligence (AI).

22.3. EMERGING TECHNOLOGIES IN DIGITAL ELECTRONICS

To understand why emerging technologies are necessary, you and I must first understand the "Von Neumann Bottleneck." In traditional computers, the CPU and the Memory (RAM) are physically separated. To perform an addition, the CPU must fetch the first number from memory, fetch the second number, add them, and send the result back. At gigahertz speeds, this constant back-and-forth travel consumes an enormous amount of time and electrical power. Today, a modern processor spends more energy moving data than it does actually computing it.

Furthermore, we are hitting the limits of Dennard Scaling—the principle that as transistors get smaller, their power density stays constant. Modern chips are now so densely packed that cooling them

has become one of the greatest engineering challenges of our time. To overcome the bottleneck and the heat wall, researchers are exploring entirely new physical substrates and architectures.

22.3.1. Quantum Computing

Classical digital electronics rely on deterministic Boolean logic. A voltage level is strictly interpreted as either a logical 0 or a logical 1. Quantum computing entirely discards this binary constraint by leveraging the bizarre laws of quantum mechanics.

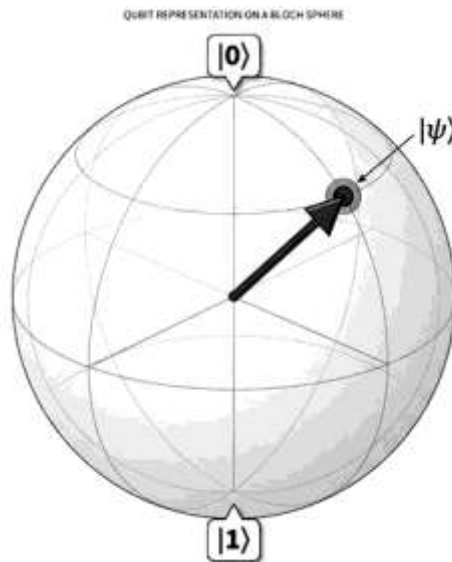


Figure 22. 1 : Qubit Representation on a Bloch Sphere.

The Qubit

The fundamental unit of a quantum computer is the quantum bit, or **Qubit**. While a classical bit is like a coin sitting flat on a table (it is either heads up or tails up), a qubit is like a coin spinning perfectly in the air. Until you force it to stop and fall (measurement), it exists in a state of both heads and tails simultaneously. This property is called **Superposition**.

Because a single qubit can represent a combination of both 0 and 1, two qubits can represent four states simultaneously (00, 01, 10, 11). Three qubits represent eight states. This grows exponentially. A quantum processor with just 300 perfectly entangled qubits could theoretically represent more states simultaneously than there are atoms in the observable universe.

Entanglement

The second pillar of quantum computing is **Entanglement**. If two qubits become entangled, their physical states become mathematically linked, regardless of the physical distance between them. If you measure one qubit and it collapses to a '1', its entangled PARTner will instantly collapse to a corresponding state. In digital logic terms, this acts like an instantaneous, invisible, zero-latency wire connecting logic gates.

What does this mean for Digital Design?

NOTES

Quantum computers will not replace your desktop PC. You will not use a quantum computer to browse the internet or write a Word document. Instead, they act as ultra-powerful co-processors for highly specific mathematical problems. For example, classical computers take thousands of years to find the prime factors of massively large numbers (the basis of RSA internet encryption). A quantum computer, utilizing Shor's Algorithm, can theoretically find these factors in minutes because it evaluates all possible answers simultaneously rather than checking them one by one sequentially.

Currently, building qubits requires extreme environments, such as cooling superconducting circuits to near absolute zero (-273°C) to prevent thermal noise from destroying the fragile quantum state (a phenomenon called decoherence).

22.3.2. Neuromorphic Computing

While quantum computing looks to physics for the future, Neuromorphic computing looks to biology. The human brain is arguably the most efficient, powerful computer in existence. It operates on barely 20 watts of power (the equivalent of a dim lightbulb), yet it can recognize faces, process complex languages, and drive a car—tasks that require massive, power-hungry supercomputers to replicate classically.

Breaking the Von Neumann Architecture

The brain does not have a separate "CPU" and "Hard Drive." In our biological neural networks, the processing (neurons) and the memory (synapses) are completely intertwined. Neuromorphic computing attempts to build silicon hardware that mimics this exact biological structure.

Spiking Neural Networks (SNNs)

Classical digital circuits run on a continuous, high-frequency master clock. Even if a logic gate isn't doing anything useful, it is still consuming power switching on and off with the clock. Biological brains are asynchronous and event-driven. A biological neuron only "fires" (spikes) when it receives enough electrical stimulus from its neighbors.

Neuromorphic chips, such as Intel's Loihi or IBM's TrueNorth, utilize **Spiking Neural Networks (SNNs)**. The digital logic only activates when an event occurs. If a neuromorphic camera is staring at a blank wall, it consumes nearly zero power because no pixels are changing, and therefore no "spikes" are generated.

The Memristor

To achieve the colocation of memory and processing, engineers are developing a fourth fundamental circuit element (alongside the resistor, capacitor, and inductor) called the **Memristor** (Memory Resistor). A memristor's electrical resistance changes based on the history of the current that has flowed through it. Even when the power is turned off, it "remembers" its last resistance state.

By organizing memristors into crossbar arrays, engineers can create artificial synapses. As current (data) passes through the array, the memristors physically change their resistance, "learning" the pattern in real-time right at the site of computation, entirely bypassing the Von Neumann bottleneck.

☑ Check Your Progress 22.1.

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. What classical computing constraint describes the performance limit caused by the physical separation of the CPU and Memory?
.....
.....
.....
2. Name the two fundamental quantum mechanical properties that allow qubits to perform massively parallel calculations.
.....
.....
.....
3. How does a Spiking Neural Network (SNN) differ from a traditional, continuously clocked digital circuit regarding power consumption?
.....
.....
.....

22.4. APPLICATIONS IN AI AND IOT

The radical advancements in digital hardware are directly fueling the two biggest technological shifts of our century: Artificial Intelligence (AI) and the Internet of Things (IoT).

Hardware Acceleration for AI

Until recently, AI software (like deep learning and large language models) was executed on general-purpose CPUs and Graphics Processing Units (GPUs). However, AI algorithms rely heavily on massive, parallel matrix multiplications. Standard CPUs are very inefficient at this.

This has led to the rise of **Application-Specific Integrated Circuits (ASICs) for AI**, often called Neural Processing Units (NPUs) or Tensor Processing Units (TPUs). These are digital circuits custom-designed purely for AI math. By stripping away all the general-purpose hardware (like branch predictors and complex instruction decoders) and replacing them with massive arrays of simple Multiply-Accumulate (MAC) logic

NOTES

units, these chips can process AI tasks hundreds of times faster than a standard CPU, using a fraction of the power.

The Internet of Things (IoT) and Edge Computing

The Internet of Things envisions a world where billions of everyday objects—from streetlights and agricultural soil sensors to refrigerators and medical implants—are connected to the internet, continuously sharing digital data.

To make this possible, digital electronics must become incredibly small and consume almost zero power.

- **Energy Harvesting:** Future IoT digital circuits are being designed to operate without batteries. They will harvest ambient energy from their environment, such as capturing micro-currents from indoor lighting (solar), thermal gradients (thermoelectric), or even stray radio frequency (RF) waves in the air.
- **Edge AI:** Currently, an IoT device like a smart speaker records your voice, sends the audio file to a massive cloud server for processing, and waits for a response. This causes latency and massive network congestion. The future trend is **Edge Computing**. By using ultra-low-power neuromorphic chips, the AI processing happens "on the edge"—right there inside the smart speaker itself. This reduces the need for constant internet transmission, drastically improves privacy, and allows for real-time, instantaneous decision-making in critical systems like autonomous vehicles.



Activity 22.1. Designing the Edge

Objective : To evaluate the benefits of Edge Computing versus Cloud Computing in a practical scenario.

Description : Imagine you are designing the digital control system for a pacemaker (a medical device implanted in the heart). You have two choices:

1. A system that sends heartbeat data via Wi-Fi to a powerful cloud server, which then sends back an instruction to shock the heart if an anomaly is detected.
2. A system that uses a tiny, low-power Neuromorphic AI chip implanted directly in the device to analyze the heartbeat locally and shock it immediately.

Reflection : Consider the consequences of a Wi-Fi outage or a 2-second transmission delay in scenario 1. Edge computing (scenario 2) is not just a convenience in future digital design; for mission-critical and life-saving applications, localized hardware processing is an absolute necessity.

✓ Check Your Progress 22.2.**NOTES**

Notes : (a) Space is given below for your answer.

(b) Check your answer with the one given at the end of this unit.

1. What specific type of mathematical operation is heavily utilized in AI and therefore hardwired into Tensor Processing Units (TPUs)?

.....

2. Briefly explain the concept of "Edge Computing" in the context of IoT devices.

.....

3. List two methods by which future digital IoT sensors might power themselves without traditional batteries.

.....

22.5. SUMMARY

The future of digital electronics is driven by the necessity to overcome the physical limits of Moore's Law and the crippling energy inefficiency of the Von Neumann Bottleneck.

Quantum Computing discards binary logic, utilizing Qubits that leverage superposition and entanglement to evaluate vast, complex mathematical probabilities simultaneously.

Neuromorphic Computing seeks to replicate the extreme energy efficiency of the human brain by colocating memory and processing using Spiking Neural Networks (SNNs) and novel circuit elements like Memristors.

Artificial Intelligence (AI) is driving a massive shift from general-purpose microprocessors to highly specialized hardware accelerators (NPU/TPUs) optimized for massive parallel matrix mathematics.

The **Internet of Things (IoT)** relies on the development of ultra-low-power digital logic, enabling billions of autonomous, battery-less devices.

Edge Computing is shifting data processing away from centralized cloud servers directly to localized hardware, drastically reducing latency, improving security, and enabling real-time autonomous systems.

NOTES

22.6. GLOSSARY

- **Decoherence:** The rapid loss of a quantum state in a qubit, usually caused by thermal noise or environmental interference, which destroys the quantum computation.
- **Edge Computing:** A distributed computing paradigm that brings digital processing and data storage closer to the physical location where it is needed (the "edge" of the network), rather than relying on a central cloud server.
- **Entanglement:** A quantum phenomenon where pairs or groups of qubits interact in ways such that the state of each qubit cannot be described independently of the state of the others.
- **Memristor:** A non-linear, two-terminal electrical component relating electric charge and magnetic flux linkage, capable of retaining a memory of past voltages/currents without power; key to artificial synapses.
- **Moore's Law:** An empirical observation that the number of transistors in a dense integrated circuit doubles about every two years.
- **Neuromorphic Engineering:** The use of very-large-scale integration (VLSI) systems containing electronic analog and digital circuits to mimic neuro-biological architectures present in the nervous system.
- **Qubit:** The basic unit of quantum information, capable of existing in a superposition of both the $|0\rangle$ and $|1\rangle$ states simultaneously.
- **Spiking Neural Network (SNN):** An artificial neural network model that more closely mimics natural neural networks, processing data dynamically via discrete electrical "spikes" rather than continuous clock cycles.
- **Von Neumann Bottleneck:** The limitation on data processing speed caused by the limited throughput between the central processing unit (CPU) and memory in a traditional computer architecture.

22.7. ANSWERS TO CHECK YOUR PROGRESS

Check Your Progress 22.1

1. The Von Neumann Bottleneck.
2. Superposition and Entanglement.
3. Traditional circuits consume power continuously on every clock tick, whereas an SNN is event-driven and only consumes active power when a "spike" (or event) occurs, saving massive amounts of energy.

Check Your Progress 22.2

1. Matrix multiplications (Multiply-Accumulate operations).
2. Edge computing means the data is processed locally right where the data is generated (on the device itself), rather than sending it across the internet to a centralized cloud server.
3. Energy harvesting via thermal gradients, ambient indoor lighting (solar), or capturing stray radio frequency (RF) energy.

NOTES**22.8. SUGGESTED READINGS - OER**

Sr. No.	Title	URL	QR Code
1	Open Educational Resource	http://books.aisectionline.com/User/UnitUrls?unitId=7141	

22.9. REFERENCE

1. Nielsen, M. A., & Chuang, I. L. (2010). *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press.
2. Mead, C. (1989). *Analog VLSI and Neural Systems*. Addison-Wesley. (The foundational text on Neuromorphic engineering).
3. Hennessy, J. L., & Patterson, D. A. (2017). *Computer Architecture: A Quantitative Approach* (6th ed.). Morgan Kaufmann. (Covers the end of Moore's law and domain-specific AI architectures).
4. Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2020). *Efficient Processing of Deep Neural Networks*. Synthesis Lectures on Computer Architecture.
5. Buyya, R., & Dastjerdi, A. V. (Eds.). (2016). *Internet of Things: Principles and Paradigms*. Morgan Kaufmann.

22.10. EXERCISE**PART 1****A. Multiple choice question**

1. Which physical phenomenon causes significant unreliability in standard silicon CMOS logic gates when transistors are shrunk below 3 nanometers?
 - (a) Electromagnetic interference
 - (b) Quantum tunneling of electrons
 - (c) Excess gravitational pull
 - (d) Alpha PARTICle radiation

NOTES

2. In Neuromorphic computing, what novel circuit component is frequently utilized to artificially mimic the memory and learning functions of a biological synapse?
 - (a) A traditional NAND gate
 - (b) A crystal oscillator
 - (c) A Memristor
 - (d) A Step-down transformer
3. A Qubit differs fundamentally from a classical binary bit because it can exist in a state of 0 and 1 simultaneously. This property is known as:
 - (a) Decoherence
 - (b) Entanglement
 - (c) Superposition
 - (d) Amplitude modulation
4. Why are dedicated Neural Processing Units (NPUs) or Tensor Processing Units (TPUs) preferred over general-purpose CPUs for training Artificial Intelligence models?
 - (a) They use vacuum tubes which do not overheat.
 - (b) They are optimized specifically for massive, parallel matrix multiplications, stripping away unnecessary general-purpose control logic
 - (c) They store data on magnetic tape, which is safer for AI.
 - (d) They rely on clock speeds over 100 GHz.
5. Which of the following is a primary advantage of utilizing "Edge Computing" for Internet of Things (IoT) devices?
 - (a) It requires massive, centralized air-conditioned data centers.
 - (b) It increases the latency of data processing.
 - (c) It processes data locally on the device, reducing internet bandwidth congestion and enabling immediate, real-time responses.
 - (d) It ensures the device consumes the maximum amount of power possible.

Answers with explanation:

1. **Answer : (b)**

Explanation : As silicon channels become microscopically thin (only a few atoms wide), electrons can physically "teleport" through the physical barrier due to quantum tunneling, making it impossible to hold a distinct "0" or "1" logic state.

2. **Answer : (c)**

Explanation : The Memristor (memory resistor) can change its resistance based on past current flow and retain that state without

power, perfectly mimicking how biological synapses strengthen or weaken based on learning stimuli.

3. **Answer : (c)**

Explanation : Superposition allows a quantum PARTICle to hold multiple probabilistic states simultaneously until it is observed or measured, giving quantum computers their massive parallel processing potential.

4. **Answer : (b)**

Explanation : AI algorithms, PARTICULARly deep learning, require millions of simultaneous multiply-accumulate operations. NPUs are custom-designed datapaths built explicitly to handle these matrix operations far more efficiently than standard CPUs.

5. **Answer : (c)**

Explanation : By analyzing data directly on the sensor (at the "edge" of the network), the system avoids the time delay of sending data to the cloud, allowing critical systems (like autonomous car brakes) to react instantly.

B. Short answer type question

1. Define the "Von Neumann Bottleneck" and explain how it limits processing speed.
2. What does it mean when two qubits are "entangled"?
3. State one major difference between a traditional synchronous digital circuit and a Spiking Neural Network (SNN).
4. Identify two ambient energy sources an IoT sensor might harvest to operate without a battery.
5. What is the fundamental difference between a classical Bit and a Qubit?

PART 2

C. Long answer type question (200 words)

1. Analyze the twilight of Moore's Law. Discuss the specific thermal constraints and physical quantum limitations that are forcing engineers to abandon traditional scaling of silicon transistors in favor of alternative computing paradigms.
2. Compare and contrast traditional von Neumann computer architectures with Neuromorphic computing architectures. Focus specifically on how each architecture handles the physical location of memory versus processing logic.
3. Explain why the rise of Artificial Intelligence (AI) led to the development of Domain-Specific Architectures like Tensor Processing Units (TPUs). Why was the traditional, general-purpose microprocessor deemed insufficient for modern deep learning tasks?

PART 3**NOTES****D. Long answer type question (300 words)**

1. Provide a comprehensive breakdown of Quantum Computing mechanics. Explain how superposition and entanglement theoretically allow a quantum processor to break modern internet encryption protocols using Shor's Algorithm.
2. Evaluate the architectural principles of Spiking Neural Networks (SNNs) and Memristor arrays. Detail how these technologies enable extreme energy efficiency in hardware.
3. Outline the systemic shift from Cloud Computing to Edge Computing in the Internet of Things (IoT). Describe the specific latency, privacy, and power benefits of processing AI algorithms locally on the hardware device.



DR. C.V. RAMAN UNIVERSITY

Chhattisgarh, Bilaspur

AN AISECT GROUP UNIVERSITY

Approved by: PCI | AICTE | NCTE | BCI | Member of: AU | Joint Committee: JGC | BCI | AICTE | Recognized by: UGC | A MAC Accredited University



DR. C. V. RAMAN UNIVERSITY (C.G.)

SLM FEEDBACK FORM

PROGRAMME : _____

COURSE : _____

BLOCK : _____

UNIT : _____

Dear Student,

While studying the units of this block, you may have found certain portions of the text difficult to comprehend. We wish to know your difficulties and suggestions, in order to improve the course. Therefore, we request you to fill up and send us the following questionnaire, which pertains to this block. If you find the space provided insufficient, kindly use a separate sheet.

Please mail to aisectcontent@aisect.org / for online submission please scan QR code:

Questionnaire

1. Name : _____

2. Enrolment No. :

3. How many hours did you need for studying the units? :

4. Please give your reactions to the following items based on your reading of the block:

Items	Excellent	Very Good	Good	Poor	Give specific Examples (If poor)
Presentation Quality					
Language and Style					
Illustrations					
Used (diagrams, tables, etc.)					
Conceptual Clarity					
Check Your Progress Questions					
Suggested Readings - OER					

5. Any other comments:
