Digital Fundamentals

ELEVENTH EDITION

Thomas L. Floyd

ALWAYS LEARNING

PEARSON

Eleventh Edition | Global Edition

Digital Fundamentals

Thomas L. Floyd

PEARSON

Boston Columbus Indianapolis New York San Francisco Hoboken
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal Toronto
Delhi Mexico City São Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Product Manager: Lindsey Prudhomme Gill
Program Manager: Maren Beckman
Project Manager: Rex Davidson
Editorial Assistant: Nancy Kesterson
Team Lead Program Manager: Laura Weaver
Team Lead Project Manager: JoEllen Gohr
Head of Learning Asset Acquisition, Global Editions: Laura Dent
Acquisitions Editor, Global Editions: Karthik Subramanian
Project Editor, Global Editions: K.K. Neelakantan
Senior Production Manufacturing Controller, Global Editions: Trudy Kimber
Director of Marketing: David Gesell
Senior Marketing Coordinator: Stacey Martinez
Senior Marketing Assistant: Les Roberts
Procurement Specialist: Deidra M. Skahill
Media Project Manager: Noelle Chun
Media Project Coordinator: April Cleland
Media Production Manager, Global Editions: Vikram Kumar
Creative Director: Andrea Nix
Art Director: Diane Y. Ernsberger
Cover Designer: Lumina Datamatics Ltd.
Cover Image: © echo3005/Shutterstock
Full-Service Project Management: Sherrill Redd/iEnergizer Aptara®, Inc.

Credits and acknowledgments for materials borrowed from other sources and reproduced, with permission, in this textbook appear on the appropriate page within text.

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies throughout the world

Visit us on the World Wide Web at:
www.pearsonglobaleditions.com

Authorized adaptation from the United States edition, entitled Digital Fundamentals,11th edition, ISBN 978-0-13-273796-8, by Thomas L. Floyd, published by Pearson Education © 2015.

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

ISBN 10: 1-292-07598-8
ISBN 13: 978-1-292-07598-3

Typeset by Aptara®, Inc. in Times Roman.
Printed and bound by Courier Kendallville in The United States of America.

PREFACE

This eleventh edition of *Digital Fundamentals* continues a long tradition of presenting a strong foundation in the core fundamentals of digital technology. This text provides basic concepts reinforced by plentiful illustrations, examples, exercises, and applications. Applied Logic features, Implementation features, troubleshooting sections, programmable logic and PLD programming, integrated circuit technologies, and the special topics of signal conversion and processing, data transmission, and data processing and control are included in addition to the core fundamentals. New topics and features have been added to this edition, and many other topics have been enhanced.

The approach used in *Digital Fundamentals* allows students to master the all-important fundamental concepts before getting into more advanced or optional topics. The range of topics provides the flexibility to accommodate a variety of program requirements. For example, some of the design-oriented or application-oriented topics may not be appropriate in some courses. Some programs may not cover programmable logic and PLD programming, while others may not have time to include data transmission or data processing. Also, some programs may not cover the details of "inside-the-chip" circuitry. These and other areas can be omitted or lightly covered without affecting the coverage of the fundamental topics. A background in transistor circuits is not a prerequisite for this textbook, and the coverage of integrated circuit technology (inside-the-chip circuits) is optionally presented.

New in This Edition

- New page layout and design for better visual appearance and ease of use
- Revised and improved topics
- Obsolete devices have been deleted.
- The *Applied Logic* features (formerly *System Applications*) have been revised and new topics added. Also, the VHDL code for PLD implementation is introduced and illustrated.
- A new boxed feature, entitled *Implementation*, shows how various logic functions can be implemented using fixed-function devices or by writing a VHDL program for PLD implementation.
- Boolean simplification coverage now includes the Quine-McCluskey method and the Espresso method is introduced.
- A discussion of Moore and Mealy state machines has been added.
- The chapter on programmable logic has been modified and improved.
- A discussion of memory hierarchy has been added.
- A new chapter on data transmission, including an extensive coverage of standard busses has been added.
- The chapter on computers has been completely revised and is now entitled "Data Processing and Control."
- A more extensive coverage and use of VHDL. There is a tutorial on the website at www.pearsonglobaleditions.com/floyd
- More emphasis on D flip-flops

Standard Features

- Full-color format
- Core fundamentals are presented without being intermingled with advanced or peripheral topics.
- *InfoNotes* are sidebar features that provide interesting information in a condensed form.
- A chapter outline, chapter objectives, introduction, and key terms list appear on the opening page of each chapter.
- Within the chapter, the key terms are highlighted in color boldface. Each key term is defined at the end of the chapter as well as in the comprehensive glossary at the end of the book. Glossary terms are indicated by black boldface in the text.
- Reminders inform students where to find the answers to the various exercises and problems throughout each chapter.
- Section introduction and objectives are at the beginning of each section within a chapter.
- Checkup exercises conclude each section in a chapter with answers at the end of the chapter.
- Each worked example has a *Related Problem* with an answer at the end of the chapter.
- *Hands-On Tips* interspersed throughout provide useful and practical information.
- Multisim files (newer versions) on the website provide circuits that are referenced in the text for optional simulation and troubleshooting.
- The operation and application of test instruments, including the oscilloscope, logic analyzer, function generator, and DMM, are covered.
- Troubleshooting sections in many chapters
- Introduction to programmable logic
- Chapter summary
- True/False quiz at end of each chapter
- Multiple-choice self-test at the end of each chapter
- Extensive sectionalized problem sets at the end of each chapter with answers to odd-numbered problems at the end of the book.
- Troubleshooting, applied logic, and special design problems are provided in many chapters.
- Coverage of bipolar and CMOS IC technologies. Chapter 15 is designed as a "floating chapter" to provide optional coverage of IC technology (inside-the-chip circuitry) at any point in the course. Chapter 15 is online at www.pearsonglobaleditions.com/floyd

Accompanying Student Resources

FIGURE P-1

- *Multisim Circuits*. The MultiSim files on the website includes selected circuits from the text that are indicated by the icon in Figure P-1.

Other student resources available on the website:

1. Chapter 15, "Integrated Circuit Technologies"
2. VHDL tutorial

3. Verilog tutorial
4. MultiSim tutorial
5. Altera Quartus II tutorial
6. Xilinx ISE tutorial
7. Five-variable Karnaugh map tutorial
8. Hamming code tutorial
9. Quine-McCluskey method tutorial
10. Espresso algorithm tutorial
11. Selected VHDL programs for downloading
12. Programming the elevator controller using Altera Quartus II

Using Website VHDL Programs

VHDL programs in the text that have a corresponding VHDL file on the website are indicated by the icon in Figure P-2. These website VHDL files can be downloaded and used in conjunction with the PLD development software (Altera Quartus II or Xilinx ISE) to implement a circuit in a programmable logic device.

FIGURE P-2

Instructor Resources

- *Image Bank* This is a download of all the images in the text.
- *Instructor's Resource Manual* Includes worked-out solutions to chapter problems, solutions to Applied Logic Exercises, and a summary of Multisim simulation results.
- *TestGen* This computerized test bank contains over 650 questions.
- **Download Instructor Resources from the Instructor Resource Center**
  To access supplementary materials online, instructors need to request an instructor access code. Go to www.pearsonglobaleditions.com/floyd to register for an instructor access code. Within 48 hours of registering, you will receive a confirming e-mail including an instructor access code. Once you have received your code, locate your text in the online catalog and click on the Instructor Resources button on the left side of the catalog product page. Select a supplement, and a login page will appear. Once you have logged in, you can access instructor material for all Pearson textbooks. If you have any difficulties accessing the site or downloading a supplement, please contact Customer Service at http://247pearsoned.custhelp.com/.

Illustration of Book Features

Chapter Opener Each chapter begins with an opener, which includes a list of the sections in the chapter, chapter objectives, introduction, a list of key terms, and a website reference for chapter study aids. A typical chapter opener is shown in Figure P-3.

Section Opener Each section in a chapter begins with a brief introduction that includes a general overview and section objectives. An illustration is shown in Figure P-4.

Section Checkup Each section ends with a review consisting of questions or exercises that emphasize the main concepts presented in the section. This feature is shown in Figure P-4. Answers to the Section Checkups are at the end of the chapter.

Worked Examples and Related Problems There is an abundance of worked out examples that help to illustrate and clarify basic concepts or specific procedures. Each example ends

CHAPTER 3

Logic Gates

CHAPTER OUTLINE

CHAPTER OBJECTIVES

- Describe the operation of the inverter, the AND gate, and the OR gate
- Describe the operation of the NAND gate and the NOR gate
- Express the operation of NOT, AND, OR, NAND, and NOR gates with Boolean algebra
- Describe the operation of the exclusive-OR and exclusive-NOR gates
- Use logic gates in simple applications
- Recognize and use both the distinctive shape logic gate symbols and the rectangular outline logic gate symbols of ANSI/IEEE Standard 91-1984/Std. 91a-1991
- Construct timing diagrams showing the proper time relationships of inputs and outputs for the various logic gates
- Discuss the basic concepts of programmable logic
- Make basic comparisons between the major IC technologies—CMOS and bipolar (TTL)
- Explain how the different series within the CMOS and bipolar (TTL) families differ from each other
- Define *propagation delay time*, *power dissipation*, *speed-power product*, and *fan-out* in relation to logic gates
- List specific fixed-function integrated circuit devices that contain the various logic gates
- Troubleshoot logic gates for opens and shorts by using the oscilloscope

KEY TERMS

Key terms are in order of appearance in the chapter.

- Inverter
- Truth table
- Boolean algebra
- Complement
- AND gate
- OR gate
- NAND gate
- NOR gate
- Exclusive-OR gate
- Exclusive-NOR gate
- AND array
- Fuse
- Antifuse
- EPROM
- EEPROM
- Flash
- SRAM
- Target device
- JTAG
- VHDL
- CMOS
- Bipolar
- Propagation delay time
- Fan-out
- Unit load

VISIT THE WEBSITE

Study aids for this chapter are available at http://www.pearsonhighered.com/careersresources/

INTRODUCTION

The emphasis in this chapter is on the operation, application, and troubleshooting of logic gates. The relationship of input and output waveforms of a gate using timing diagrams is thoroughly covered.

Logic symbols used to represent the logic gates are in accordance with ANSI/IEEE Standard 91-1984/Std. 91a-1991. This standard has been adopted by private industry and the military for use in internal documentation as well as published literature.

FIGURE P-3

SECTION 5–1 CHECKUP

Answers are at the end of the chapter.

1. Determine the output (1 or 0) of a 4-variable AND-OR-Invert circuit for each of the following input conditions:
   (a) $A = 1, B = 0, C = 1, D = 0$ (b) $A = 1, B = 1, C = 0, D = 1$
   (c) $A = 0, B = 1, C = 1, D = 1$
2. Determine the output (1 or 0) of an exclusive-OR gate for each of the following input conditions:
   (a) $A = 1, B = 0$ (b) $A = 1, B = 1$
   (c) $A = 0, B = 1$ (d) $A = 0, B = 0$
3. Develop the truth table for a certain 3-input logic circuit with the output expression $X = A\overline{B}C + \overline{A}BC + \overline{A}\,\overline{B}\,\overline{C} + AB\overline{C} + ABC$.
4. Draw the logic diagram for an exclusive-NOR circuit.

5–2 Implementing Combinational Logic

In this section, examples are used to illustrate how to implement a logic circuit from a Boolean expression or a truth table. Minimization of a logic circuit using the methods covered in Chapter 4 is also included.

After completing this section, you should be able to

- Implement a logic circuit from a Boolean expression
- Implement a logic circuit from a truth table
- Minimize a logic circuit

For every Boolean expression there is a logic circuit, and for every logic circuit there is a Boolean expression.

From a Boolean Expression to a Logic Circuit

Let's examine the following Boolean expression:

$$X = AB + CDE$$

A brief inspection shows that this expression is composed of two terms, AB and CDE, with a domain of five variables. The first term is formed by ANDing A with B, and the second term is formed by ANDing C, D, and E. The two terms are then ORed to form the output X. These operations are indicated in the structure of the expression as follows:

$X = AB + CDE$ (AND over AB and CDE; OR over the sum)

Note that in this particular expression, the AND operations forming the two individual terms, AB and CDE, must be performed *before* the terms can be ORed.

To implement this Boolean expression, a 2-input AND gate is required to form the term AB, and a 3-input AND gate is needed to form the term CDE. A 2-input OR gate is then required to combine the two AND terms. The resulting logic circuit is shown in Figure 5–9.

As another example, let's implement the following expression:

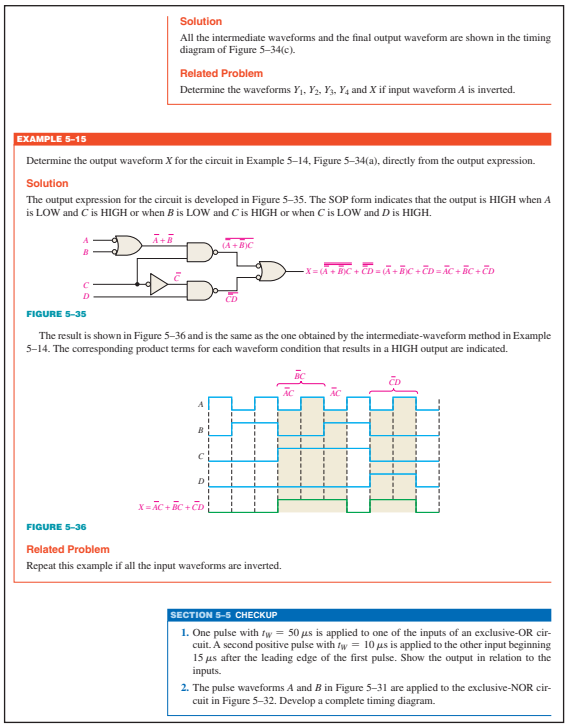
$$X = AB(C\overline{D} + EF)$$

InfoNote

Many control programs require logic operations to be performed by a computer. A driver program is a control program that is used with computer peripherals. For example, a mouse driver requires logic tests to determine if a button has been pressed and further logic operations to determine if it has moved, either horizontally or vertically. Within the heart of a microprocessor is the arithmetic logic unit (ALU), which performs these logic operations as directed by program instructions. All of the logic described in this chapter can also be performed by the ALU, given the proper instructions.

FIGURE P-4

with a *Related Problem* that reinforces or expands on the example by requiring the student to work through a problem similar to the example. A typical worked example with *Related Problem* is shown in Figure P-5.



Solution

All the intermediate waveforms and the final output waveform are shown in the timing diagram of Figure 5–34(c).

Related Problem

Determine the waveforms Y_1, Y_2, Y_3, Y_4 and X if input waveform A is inverted.

EXAMPLE 5–15

Determine the output waveform X for the circuit in Example 5–14, Figure 5–34(a), directly from the output expression.

Solution

The output expression for the circuit is developed in Figure 5–35. The SOP form indicates that the output is HIGH when A is LOW and C is HIGH or when B is LOW and C is HIGH or when C is LOW and D is HIGH.

FIGURE 5–35

The result is shown in Figure 5–36 and is the same as the one obtained by the intermediate-waveform method in Example 5–14. The corresponding product terms for each waveform condition that results in a HIGH output are indicated.

FIGURE 5–36

Related Problem

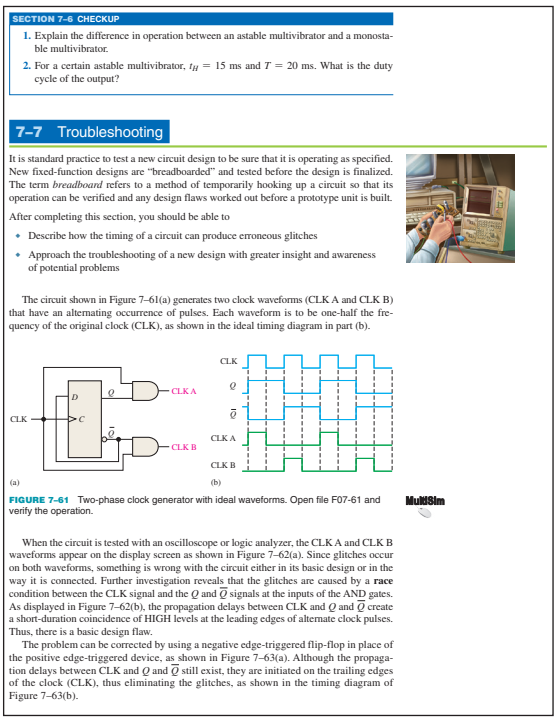
Repeat this example if all the input waveforms are inverted.

SECTION 5–5 CHECKUP

1. One pulse with $t_W = 50\ \mu s$ is applied to one of the inputs of an exclusive-OR circuit. A second positive pulse with $t_W = 10\ \mu s$ is applied to the other input beginning $15\ \mu s$ after the leading edge of the first pulse. Show the output in relation to the inputs.
2. The pulse waveforms A and B in Figure 5–31 are applied to the exclusive-NOR circuit in Figure 5–32. Develop a complete timing diagram.

FIGURE P-5

Troubleshooting Section Many chapters include a troubleshooting section that relates to the topics covered in the chapter and that emphasizes troubleshooting techniques and the use of test instruments and circuit simulation. A portion of a typical troubleshooting section is illustrated in Figure P-6.



SECTION 7–6 CHECKUP

1. Explain the difference in operation between an astable multivibrator and a monostable multivibrator.
2. For a certain astable multivibrator, $t_H = 15$ ms and $T = 20$ ms. What is the duty cycle of the output?

7–7 Troubleshooting

It is standard practice to test a new circuit design to be sure that it is operating as specified. New fixed-function designs are "breadboarded" and tested before the design is finalized. The term *breadboard* refers to a method of temporarily hooking up a circuit so that its operation can be verified and any design flaws worked out before a prototype unit is built.

After completing this section, you should be able to

- Describe how the timing of a circuit can produce erroneous glitches
- Approach the troubleshooting of a new design with greater insight and awareness of potential problems

The circuit shown in Figure 7–61(a) generates two clock waveforms (CLK A and CLK B) that have an alternating occurrence of pulses. Each waveform is to be one-half the frequency of the original clock (CLK), as shown in the ideal timing diagram in part (b).

FIGURE 7–61 Two-phase clock generator with ideal waveforms. Open file F07-61 and verify the operation.

MultiSim

When the circuit is tested with an oscilloscope or logic analyzer, the CLK A and CLK B waveforms appear on the display screen as shown in Figure 7–62(a). Since glitches occur on both waveforms, something is wrong with the circuit either in its basic design or in the way it is connected. Further investigation reveals that the glitches are caused by a **race** condition between the CLK signal and the Q and $\overline{Q}$ signals at the inputs of the AND gates. As displayed in Figure 7–62(b), the propagation delays between CLK and Q and $\overline{Q}$ create a short-duration coincidence of HIGH levels at the leading edges of alternate clock pulses. Thus, there is a basic design flaw.

The problem can be corrected by using a negative edge-triggered flip-flop in place of the positive edge-triggered device, as shown in Figure 7–63(a). Although the propagation delays between CLK and Q and $\overline{Q}$ still exist, they are initiated on the trailing edges of the clock (CLK), thus eliminating the glitches, as shown in the timing diagram of Figure 7–63(b).

(a) Oscilloscope display of CLK A and CLK B waveforms with glitches indicated by the "spikes".

(b) Oscilloscope display showing propagation delay that creates glitch on CLK A waveform

FIGURE 7–62 Oscilloscope displays for the circuit in Figure 7–61.

MultiSim

FIGURE 7–63 Two-phase clock generator using negative edge-triggered flip-flop to eliminate glitches. Open file F07-63 and verify the operation.

HandsOnTip

Glitches that occur in digital systems are very fast (extremely short in duration) and can be difficult to see on an oscilloscope, particularly at lower sweep rates. A logic analyzer, however, can show a glitch easily. To look for glitches using a logic analyzer, select "latch" mode or (if available) transitional sampling. In the latch mode, the analyzer looks for a voltage level change. When a change occurs, even if it is of extremely short duration (a few nanoseconds), the information is "latched" into the analyzer's memory as another sampled data point. When the data are displayed, the glitch will show as an obvious change in the sampled data, making it easy to identify.

SECTION 7–7 CHECKUP

1. Can a negative edge-triggered J-K flip-flop be used in the circuit of Figure 7–63?
2. What device can be used to provide the clock for the circuit in Figure 7–63?

FIGURE P-6

Applied Logic Appearing at the end of many chapters, this feature presents a practical application of the concepts and procedures covered in the chapter. In most chapters, this feature presents a "real-world" application in which analysis, troubleshooting, design, VHDL programming, and simulation are implemented. Figure P-7 shows a portion of a typical Applied Logic feature.

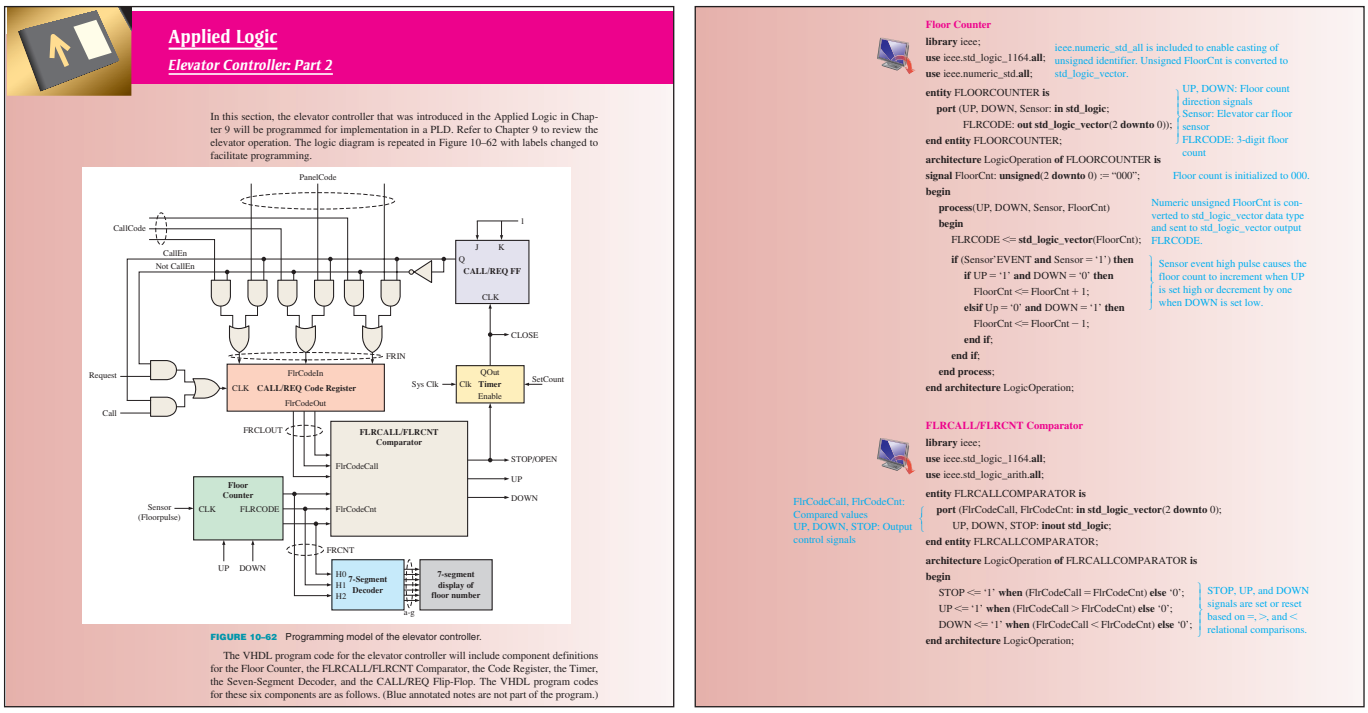


FIGURE P-7

End of Chapter

The following features are at the end of each chapter:

- Summary
- Key term glossary
- True/false quiz
- Self-test
- Problem set that includes some or all of the following categories in addition to core problems: Troubleshooting, Applied Logic, Design, and Multisim Troubleshooting Practice.
- Answers to Section Checkups
- Answers to Related Problems for Examples
- Answers to True/False quiz
- Answers to Self-Test

End of Book

The following features are at the end of the book.

- Answers to selected odd-numbered problems
- Comprehensive glossary
- Index

To the Student

Digital technology pervades almost everything in our daily lives. For example, cell phones and other types of wireless communications, television, radio, process controls, automotive electronics, consumer electronics, aircraft navigation— to name only a few applications—depend heavily on digital electronics.

A strong grounding in the fundamentals of digital technology will prepare you for the highly skilled jobs of the future. The single most important thing you can do is to understand the core fundamentals. From there you can go anywhere.

In addition, programmable logic is important in many applications and that topic in introduced in this book and example programs are given along with an online tutorial. Of course, efficient troubleshooting is a skill that is also widely sought after by potential employers. Troubleshooting and testing methods from traditional prototype testing to more advanced techniques such as boundary scan are covered.

To the Instructor

Generally, time limitations or program emphasis determines the topics to be covered in a course. It is not uncommon to omit or condense topics or to alter the sequence of certain topics in order to customize the material for a particular course. This textbook is specifically designed to provide great flexibility in topic coverage.

Certain topics are organized in separate chapters, sections, or features such that if they are omitted the rest of the coverage is not affected. Also, if these topics are included, they flow seamlessly with the rest of the coverage. The book is organized around a core of fundamental topics that are, for the most part, essential in any digital course. Around this core, there are other topics that can be included or omitted, depending on the course emphasis and/or other factors. Even within the core, selected topics can be omitted. Figure P-8 illustrates this concept.

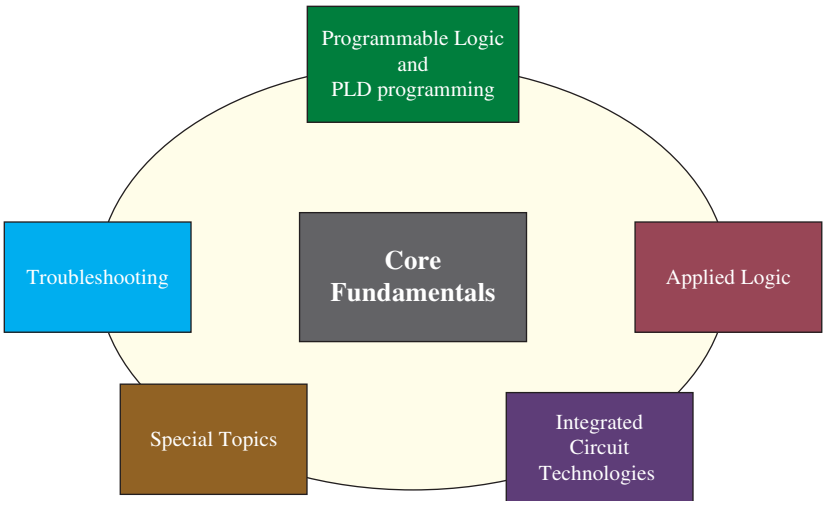


FIGURE P-8

- ***Core Fundamentals*** The fundamental topics of digital technology should be covered in all programs. Linked to the core are several "satellite" topics that may be considered for omission or inclusion, depending on your course goals. All topics presented in this text are important in digital technology, but each block surrounding the core can be omitted, depending on your particular goals, without affecting the core fundamentals.
- ***Programmable Logic and PLD Programming*** Although they are important topics, programmable logic and VHDL can be omitted; however, it is highly recommended that you cover this topic if at all possible. You can cover as little or as much as you consider appropriate for your program.

- ***Troubleshooting*** Troubleshooting sections appear in many chapters and include the application and operation of laboratory instruments.
- ***Applied Logic*** Selected real-world applications appear in many chapters.
- ***Integrated Circuit Technologies*** Chapter 15 is an online chapter. Some or all of the topics in Chapter 15 can be covered at selected points if you wish to discuss details of the circuitry that make up digital integrated circuits. Chapter 15 can be omitted without any impact on the rest of the book.
- ***Special Topics*** These topics are *Signal Interfacing and Processing, Data Transmission,* and *Data Processing and Control* in Chapters 12, 13, and 14 respectively, as well as selected topics in other chapters. These are topics that may not be essential for your course or are covered in another course. Also, within each block in Figure P-8 you can choose to omit or deemphasize some topics because of time constraints or other priorities in your particular program. For example in the core fundamentals, the Quine-McCluskey method, cyclic redundancy code, carry look-ahead adders, or sequential logic design could possibly be omitted. Additionally, any or all of Multisim features throughout the book can be treated as optional. Other topics may also be candidates for omission or light coverage. Whether you choose a minimal coverage of only core fundamentals, a full-blown coverage of all the topics, or anything in between, this book can be adapted to your needs.

Acknowledgments

This revision of *Digital Fundamentals* has been made possible by the work and skills of many people. I think that we have accomplished what we set out to do, and that was to further improve an already very successful textbook and make it even more useful to the student and instructor by presenting not only basics but also up-to-date and leading-edge technology.

Those at Pearson Education who have, as always, contributed a great amount of time, talent, and effort to move this project through its many phases in order to produce the book as you see it, include, but are not limited to, Rex Davidson, Lindsey Gill, and Vern Anthony. Lois Porter has done another excellent job of manuscript editing. Doug Joksch contributed the VHDL programming. Gary Snyder revised and updated the Multisim circuit files. My thanks and appreciation go to all of these and others who were indirectly involved in the project.

In the revision of this and all textbooks, I depend on expert input from many users as well as nonusers. My sincere thanks to the following reviewers who submitted many valuable suggestions and provided lots of constructive criticism:

Dr. Cuiling Gong,
Texas Christian University;

Zane Gastineau,
Harding University; and

Jonathan White,
Harding University;

Dr. Eric Bothur,
Midlands Technical College.

I also want to thank all of the members of the Pearson sales force whose efforts have helped make this text available to a large number of users. In addition, I am grateful to all of you who have adopted this text for your classes or for your own use. Without you we would not be in business. I hope that you find this eleventh edition of *Digital Fundamentals* to be even better than earlier editions and that it will continue to be a valuable learning tool and reference for the student.

Tom Floyd

Pearson would like to thank and acknowledge Sanjay H.S., M.S. Ramaiah Institute of Technology for his contributions to the Global Edition, and Moumita Mitra Manna, Bangabasi College, and Piyali Sengupta for reviewing the Global Edition.

CONTENTS

CHAPTER 1

Introductory Concepts

CHAPTER OUTLINE

CHAPTER OBJECTIVES

- Explain the basic differences between digital and analog quantities
- Show how voltage levels are used to represent digital quantities
- Describe various parameters of a pulse waveform such as rise time, fall time, pulse width, frequency, period, and duty cycle
- Explain the basic logic functions of NOT, AND, and OR
- Describe several types of logic operations and explain their application in an example system
- Describe programmable logic, discuss the various types, and describe how PLDs are programmed
- Identify fixed-function digital integrated circuits according to their complexity and the type of circuit packaging
- Identify pin numbers on integrated circuit packages
- Recognize various instruments and understand how they are used in measurement and troubleshooting digital circuits and systems
- Describe basic troubleshooting methods

KEY TERMS

Key terms are in order of appearance in the chapter.

- Analog
- Digital
- Binary
- Bit
- Pulse
- Duty cycle
- Clock
- Timing diagram
- Data
- Serial
- Parallel
- Logic
- Input
- Output
- Gate
- NOT
- Inverter
- AND
- OR
- Programmable logic
- SPLD
- CPLD
- FPGA
- Microcontroller
- Embedded system
- Compiler
- Integrated circuit (IC)
- Fixed-function logic
- Troubleshooting

VISIT THE WEBSITE

Study aids for this chapter are available at http://www.pearsonglobaleditions.com/floyd

INTRODUCTION

The term *digital* is derived from the way operations are performed, by counting digits. For many years, applications of digital electronics were confined to computer systems. Today, digital technology is applied in a wide range of areas in addition to computers. Such applications as television, communications systems, radar, navigation and guidance systems, military systems, medical instrumentation, industrial process control, and consumer electronics use digital techniques. Over the years digital technology has progressed from vacuum-tube circuits

to discrete transistors to complex integrated circuits, many of which contain millions of transistors, and many of which are programmable.

This chapter introduces you to digital electronics and provides a broad overview of many important concepts, components, and tools.

1–1 Digital and Analog Quantities

Electronic circuits can be divided into two broad categories, digital and analog. Digital electronics involves quantities with discrete values, and analog electronics involves quantities with continuous values. Although you will be studying digital fundamentals in this book, you should also know something about analog because many applications require both; and interfacing between analog and digital is important.

After completing this section, you should be able to

- Define *analog*
- Define *digital*
- Explain the difference between digital and analog quantities
- State the advantages of digital over analog
- Give examples of how digital and analog quantities are used in electronics

An **analog*** quantity is one having continuous values. A **digital** quantity is one having a discrete set of values. Most things that can be measured quantitatively occur in nature in analog form. For example, the air temperature changes over a continuous range of values. During a given day, the temperature does not go from, say, 70° to 71° instantaneously; it takes on all the infinite values in between. If you graphed the temperature on a typical summer day, you would have a smooth, continuous curve similar to the curve in Figure 1–1. Other examples of analog quantities are time, pressure, distance, and sound.

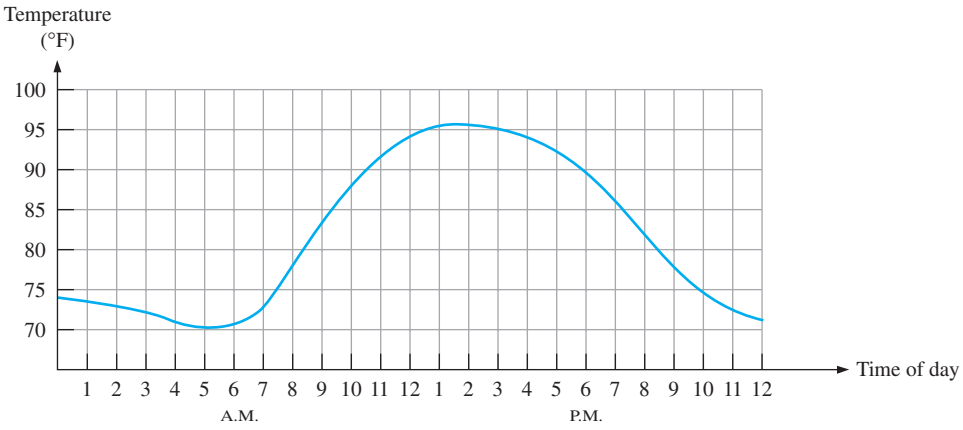


FIGURE 1–1 Graph of an analog quantity (temperature versus time).

Rather than graphing the temperature on a continuous basis, suppose you just take a temperature reading every hour. Now you have sampled values representing the temperature at discrete points in time (every hour) over a 24-hour period, as indicated in Figure 1–2.

*All bold terms are important and are defined in the end-of-book glossary. The blue bold terms are key terms and are included in a Key Term glossary at the end of each chapter.

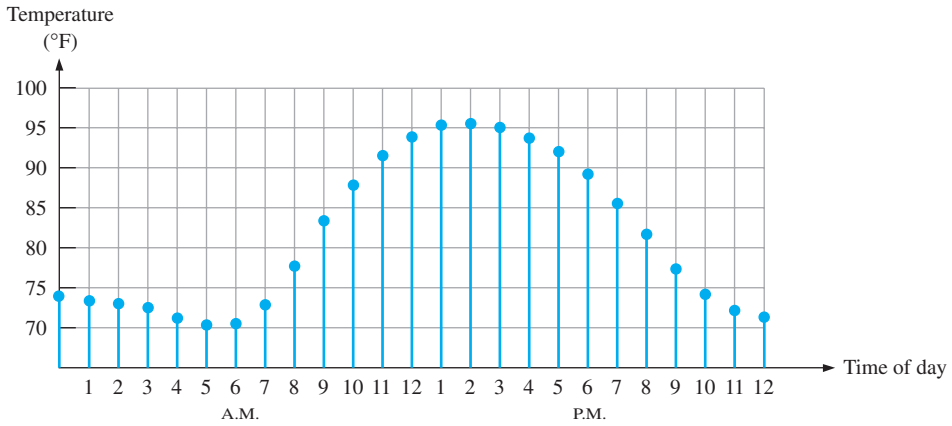


FIGURE 1–2 Sampled-value representation (quantization) of the analog quantity in Figure 1–1. Each value represented by a dot can be digitized by representing it as a digital code that consists of a series of 1s and 0s.

You have effectively converted an analog quantity to a form that can now be digitized by representing each sampled value by a digital code. It is important to realize that Figure 1–2 itself is not the digital representation of the analog quantity.

The Digital Advantage

Digital representation has certain advantages over analog representation in electronics applications. For one thing, digital data can be processed and transmitted more efficiently and reliably than analog data. Also, digital data has a great advantage when storage is necessary. For example, music when converted to digital form can be stored more compactly and reproduced with greater accuracy and clarity than is possible when it is in analog form. Noise (unwanted voltage fluctuations) does not affect digital data nearly as much as it does analog signals.

An Analog System

A public address system, used to amplify sound so that it can be heard by a large audience, is one simple example of an application of analog electronics. The basic diagram in Figure 1–3 illustrates that sound waves, which are analog in nature, are picked up by a microphone and converted to a small analog voltage called the audio signal. This voltage varies continuously as the volume and frequency of the sound changes and is applied to the input of a linear amplifier. The output of the amplifier, which is an increased reproduction of input voltage, goes to the speaker(s). The speaker changes the amplified audio signal back to sound waves that have a much greater volume than the original sound waves picked up by the microphone.

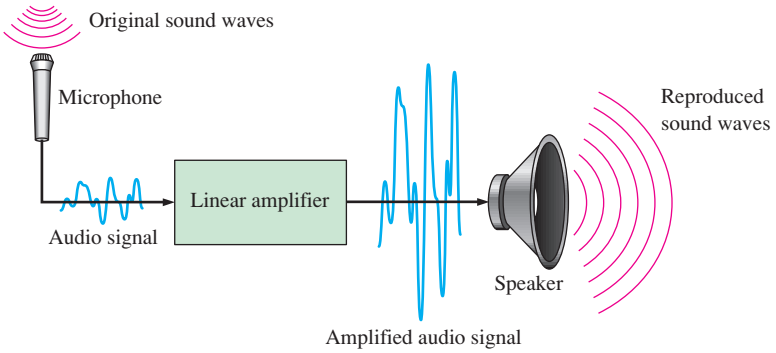


FIGURE 1–3 A basic audio public address system.

A System Using Digital and Analog Methods

The compact disk (CD) player is an example of a system in which both digital and analog circuits are used. The simplified block diagram in Figure 1–4 illustrates the basic principle. Music in digital form is stored on the compact disk. A laser diode optical system picks up the digital data from the rotating disk and transfers it to the **digital-to-analog converter (DAC)**. The DAC changes the digital data into an analog signal that is an electrical reproduction of the original music. This signal is amplified and sent to the speaker for you to enjoy. When the music was originally recorded on the CD, a process, essentially the reverse of the one described here, using an **analog-to-digital converter (ADC)** was used.

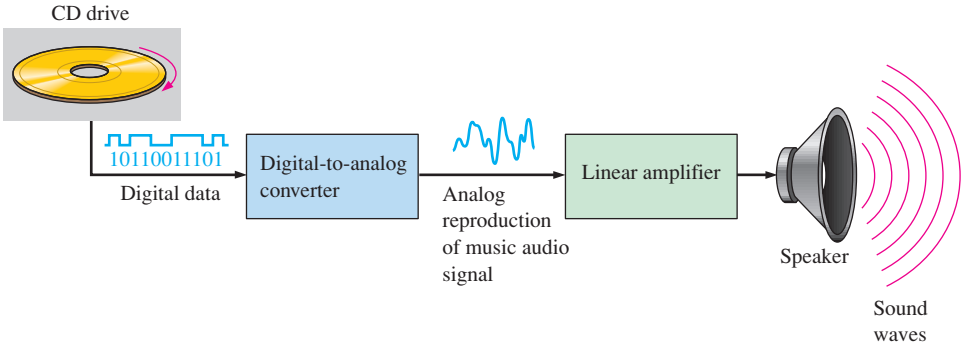


FIGURE 1–4 Basic block diagram of a CD player. Only one channel is shown.

Mechatronics

Both digital and analog electronics are used in the control of various mechanical systems. The interdisciplinary field that comprises both mechanical and electronic components is known as **mechatronics**.

Mechatronic systems are found in homes, industry, and transportation. Most home appliances consist of both mechanical and electronic components. Electronics controls the operation of a washing machine in terms of water flow, temperature, and type of cycle. Manufacturing industries rely heavily on mechatronics for process control and assembly. In automotive and other types of manufacturing, robotic arms perform precision welding, painting, and other functions on the assembly line. Automobiles themselves are mechatronic machines; a digital computer controls functions such as braking, engine parameters, fuel flow, safety features, and monitoring.

Figure 1–5(a) is a basic block diagram of a mechatronic system. A simple robotic arm is shown in Figure 1–5(b), and robotic arms on an automotive assembly line are shown in part (c).

(a) Mechatronic system block diagram

(b) Robotic arm

(c) Automotive assembly line

FIGURE 1–5 Example of a mechatronic system and application. Part (b) Beawolf/Fotolia; Part (c) Small Town Studio/Fotolia.

The movement of the arm in any quadrant and to any specified position is accomplished with some type of digital control such as a microcontroller.

SECTION 1–1 CHECKUP

Answers are at the end of the chapter.

1. Define *analog.*
2. Define *digital.*
3. Explain the difference between a digital quantity and an analog quantity.
4. Give an example of a system that is analog and one that is a combination of both digital and analog. Name a system that is entirely digital.
5. What does a mechatronic system consist of?

1–2 Binary Digits, Logic Levels, and Digital Waveforms

Digital electronics involves circuits and systems in which there are only two possible states. These states are represented by two different voltage levels: A HIGH and a LOW. The two states can also be represented by current levels, bits and bumps on a CD or DVD, etc. In digital systems such as computers, combinations of the two states, called *codes,* are used to represent numbers, symbols, alphabetic characters, and other types of information. The two-state number system is called *binary,* and its two digits are 0 and 1. A binary digit is called a *bit.*

After completing this section, you should be able to

- Define *binary*
- Define *bit*
- Name the bits in a binary system
- Explain how voltage levels are used to represent bits
- Explain how voltage levels are interpreted by a digital circuit
- Describe the general characteristics of a pulse
- Determine the amplitude, rise time, fall time, and width of a pulse
- Identify and describe the characteristics of a digital waveform
- Determine the amplitude, period, frequency, and duty cycle of a digital waveform
- Explain what a timing diagram is and state its purpose
- Explain serial and parallel data transfer and state the advantage and disadvantage of each

InfoNote

The concept of a digital computer can be traced back to Charles Babbage, who developed a crude mechanical computation device in the 1830s. John Atanasoff was the first to apply electronic processing to digital computing in 1939. In 1946, an electronic digital computer called ENIAC was implemented with vacuum-tube circuits. Even though it took up an entire room, ENIAC didn't have the computing power of your handheld calculator.

Binary Digits

Each of the two digits in the **binary** system, 1 and 0, is called a **bit**, which is a contraction of the words *binary digit.* In digital circuits, two different voltage levels are used to represent the two bits. Generally, 1 is represented by the higher voltage, which we will refer to as a HIGH, and a 0 is represented by the lower voltage level, which we will refer to as a LOW. This is called **positive logic** and will be used throughout the book.

HIGH = 1 and LOW = 0

Another system in which a 1 is represented by a LOW and a 0 is represented by a HIGH is called *negative logic.*

Groups of bits (combinations of 1s and 0s), called *codes,* are used to represent numbers, letters, symbols, instructions, and anything else required in a given application.

Logic Levels

The voltages used to represent a 1 and a 0 are called *logic levels.* Ideally, one voltage level represents a HIGH and another voltage level represents a LOW. In a practical digital circuit, however, a HIGH can be any voltage between a specified minimum value and a specified maximum value. Likewise, a LOW can be any voltage between a specified minimum and a specified maximum. There can be no overlap between the accepted range of HIGH levels and the accepted range of LOW levels.

Figure 1–6 illustrates the general range of LOWs and HIGHs for a digital circuit. The variable $V_{H(max)}$ represents the maximum HIGH voltage value, and $V_{H(min)}$ represents the minimum HIGH voltage value. The maximum LOW voltage value is represented by $V_{L(max)}$, and the minimum LOW voltage value is represented by $V_{L(min)}$. The voltage values between $V_{L(max)}$ and $V_{H(min)}$ are unacceptable for proper operation. A voltage in the unacceptable range can appear as either a HIGH or a LOW to a given circuit. For example, the HIGH input values for a certain type of digital circuit technology called CMOS may range from 2 V to 3.3 V and the LOW input values may range from 0 V to 0.8 V. If a voltage of 2.5 V is applied, the circuit will accept it as a HIGH or binary 1. If a voltage of 0.5 V is applied, the circuit will accept it as a LOW or binary 0. For this type of circuit, voltages between 0.8 V and 2 V are unacceptable.

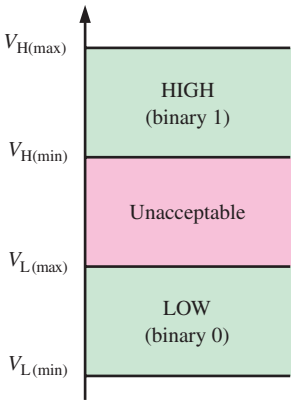
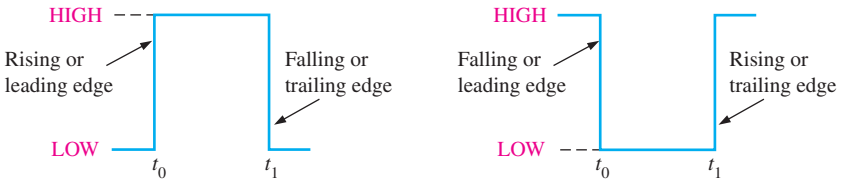


FIGURE 1–6 Logic level ranges of voltage for a digital circuit.

Digital Waveforms

Digital waveforms consist of voltage levels that are changing back and forth between the HIGH and LOW levels or states. Figure 1–7(a) shows that a single positive-going **pulse** is generated when the voltage (or current) goes from its normally LOW level to its HIGH level and then back to its LOW level. The negative-going pulse in Figure 1–7(b) is generated when the voltage goes from its normally HIGH level to its LOW level and back to its HIGH level. A digital waveform is made up of a series of pulses.



(a) Positive–going pulse

(b) Negative–going pulse

FIGURE 1–7 Ideal pulses.

The Pulse

As indicated in Figure 1–7, a pulse has two edges: a **leading edge** that occurs first at time t_0 and a **trailing edge** that occurs last at time t_1. For a positive-going pulse, the leading edge is a rising edge, and the trailing edge is a falling edge. The pulses in Figure 1–7 are ideal because the rising and falling edges are assumed to change in zero time (instantaneously). In practice, these transitions never occur instantaneously, although for most digital work you can assume ideal pulses.

Figure 1–8 shows a nonideal pulse. In reality, all pulses exhibit some or all of these characteristics. The overshoot and ringing are sometimes produced by stray inductive and

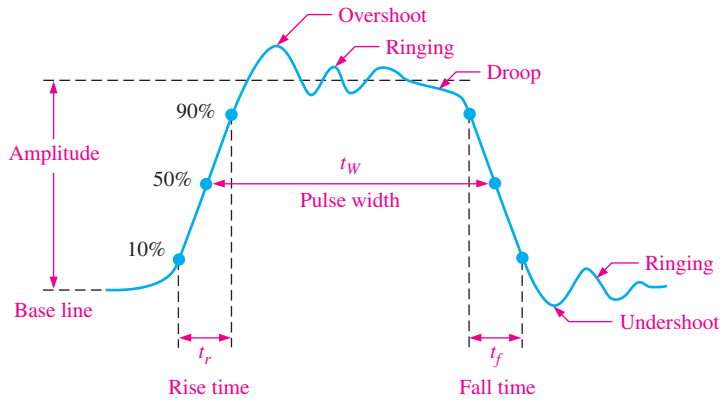


FIGURE 1–8 Nonideal pulse characteristics.

capacitive effects. The droop can be caused by stray capacitive and circuit resistance, forming an *RC* circuit with a low time constant.

The time required for a pulse to go from its LOW level to its HIGH level is called the **rise time (t_r)**, and the time required for the transition from the HIGH level to the LOW level is called the **fall time (t_f)**. In practice, it is common to measure rise time from 10% of the pulse **amplitude** (height from baseline) to 90% of the pulse amplitude and to measure the fall time from 90% to 10% of the pulse amplitude, as indicated in Figure 1–8. The bottom 10% and the top 10% of the pulse are not included in the rise and fall times because of the nonlinearities in the waveform in these areas. The **pulse width (t_W)** is a measure of the duration of the pulse and is often defined as the time interval between the 50% points on the rising and falling edges, as indicated in Figure 1–8.

Waveform Characteristics

Most waveforms encountered in digital systems are composed of series of pulses, sometimes called *pulse trains,* and can be classified as either periodic or nonperiodic. A **periodic** pulse waveform is one that repeats itself at a fixed interval, called a **period (*T*)**. The **frequency (*f*)** is the rate at which it repeats itself and is measured in hertz (Hz). A nonperiodic pulse waveform, of course, does not repeat itself at fixed intervals and may be composed of pulses of randomly differing pulse widths and/or randomly differing time intervals between the pulses. An example of each type is shown in Figure 1–9.

(a) Periodic (square wave)

(b) Nonperiodic

FIGURE 1–9 Examples of digital waveforms.

The frequency (f) of a pulse (digital) waveform is the reciprocal of the period. The relationship between frequency and period is expressed as follows:

$$f = \frac{1}{T} \qquad \text{Equation 1–1}$$

$$T = \frac{1}{f} \qquad \text{Equation 1–2}$$

An important characteristic of a periodic digital waveform is its **duty cycle**, which is the ratio of the pulse width (t_W) to the period (T). It can be expressed as a percentage.

$$\textbf{Duty cycle} = \left(\frac{t_W}{T}\right)100\% \qquad \text{Equation 1–3}$$

EXAMPLE 1–1

A portion of a periodic digital waveform is shown in Figure 1–10. The measurements are in milliseconds. Determine the following:

(a) period **(b)** frequency **(c)** duty cycle

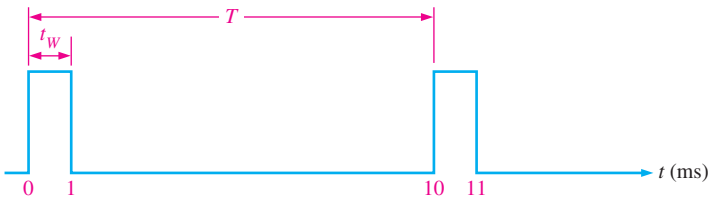


FIGURE 1–10

Solution

(a) The period (T) is measured from the edge of one pulse to the corresponding edge of the next pulse. In this case T is measured from leading edge to leading edge, as indicated. T equals **10 ms**.

(b) $f = \dfrac{1}{T} = \dfrac{1}{10\text{ ms}} = \mathbf{100\ Hz}$

(c) $\text{Duty cycle} = \left(\dfrac{t_W}{T}\right)100\% = \left(\dfrac{1\text{ ms}}{10\text{ ms}}\right)100\% = \mathbf{10\%}$

Related Problem*

A periodic digital waveform has a pulse width of 25 μs and a period of 150 μs. Determine the frequency and the duty cycle.

*Answers are at the end of the chapter.

A Digital Waveform Carries Binary Information

InfoNote

The speed at which a computer can operate depends on the type of microprocessor used in the system. The speed specification, for example 3.5 GHz, of a computer is the maximum clock frequency at which the microprocessor can run.

Binary information that is handled by digital systems appears as waveforms that represent sequences of bits. When the waveform is HIGH, a binary 1 is present; when the waveform is LOW, a binary 0 is present. Each bit in a sequence occupies a defined time interval called a **bit time**.

The Clock

In digital systems, all waveforms are synchronized with a basic timing waveform called the **clock.** The clock is a periodic waveform in which each interval between pulses (the period) equals the time for one bit.

An example of a clock waveform is shown in Figure 1–11. Notice that, in this case, each change in level of waveform *A* occurs at the leading edge of the clock waveform. In other cases, level changes occur at the trailing edge of the clock. During each bit time of the clock, waveform *A* is either HIGH or LOW. These HIGHs and LOWs represent a sequence

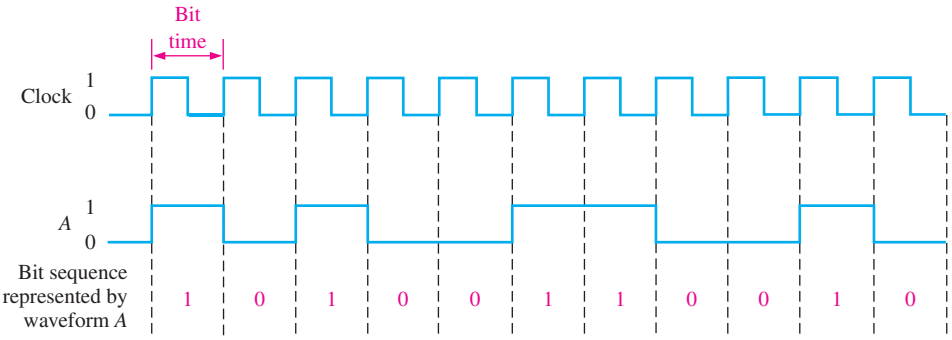


FIGURE 1–11 Example of a clock waveform synchronized with a waveform representation of a sequence of bits.

of bits as indicated. A group of several bits can contain binary information, such as a number or a letter. The clock waveform itself does not carry information.

Timing Diagrams

A **timing diagram** is a graph of digital waveforms showing the actual time relationship of two or more waveforms and how each waveform changes in relation to the others. By looking at a timing diagram, you can determine the states (HIGH or LOW) of all the waveforms at any specified point in time and the exact time that a waveform changes state relative to the other waveforms. Figure 1–12 is an example of a timing diagram made up of four waveforms. From this timing diagram you can see, for example, that the three waveforms *A*, *B*, and *C* are HIGH only during bit time 7 (shaded area) and they all change back LOW at the end of bit time 7.

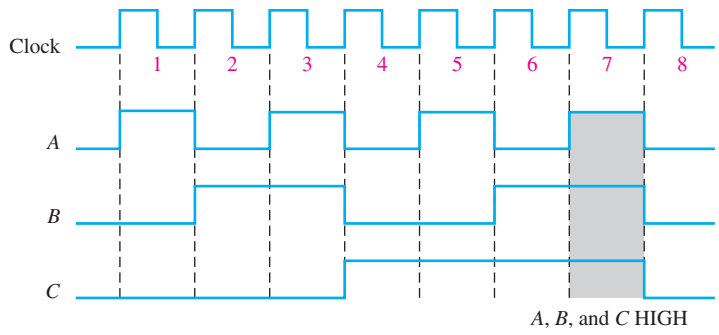


FIGURE 1–12 Example of a timing diagram.

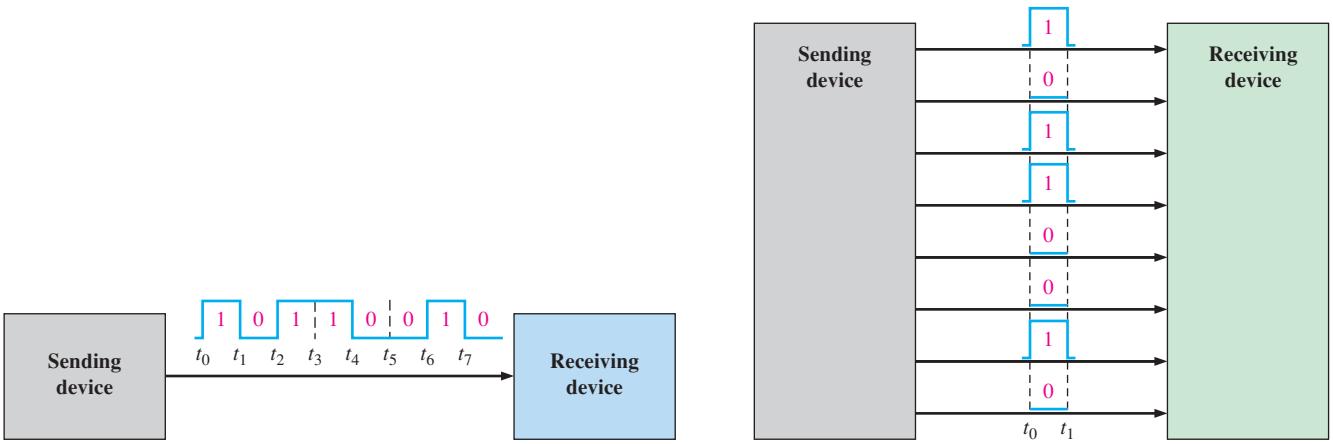
Data Transfer

Data refers to groups of bits that convey some type of information. Binary data, which are represented by digital waveforms, must be transferred from one device to another within a digital system or from one system to another in order to accomplish a given purpose. For example, numbers stored in binary form in the memory of a computer must be transferred to the computer's central processing unit in order to be added. The sum of the addition must then be transferred to a monitor for display and/or transferred back to the memory. As illustrated in Figure 1–13, binary data are transferred in two ways: serial and parallel.

When bits are transferred in **serial** form from one point to another, they are sent one bit at a time along a single line, as illustrated in Figure 1–13(a). During the time interval from t_0 to t_1, the first bit is transferred. During the time interval from t_1 to t_2, the second bit is transferred, and so on. To transfer eight bits in series, it takes eight time intervals.

InfoNote

Universal Serial Bus (USB) is a serial bus standard for device interfacing. It was originally developed for the personal computer but has become widely used on many types of handheld and mobile devices. USB is expected to replace other serial and parallel ports. USB operated at 12 Mbps (million bits per second) when first introduced in 1995, but it now provides transmission speeds of up to 5 Gbps.



(a) Serial transfer of 8 bits of binary data. Interval t_0 to t_1 is first.

(b) Parallel transfer of 8 bits of binary data. The beginning time is t_0.

FIGURE 1–13 Illustration of serial and parallel transfer of binary data. Only the data lines are shown.

When bits are transferred in **parallel** form, all the bits in a group are sent out on separate lines at the same time. There is one line for each bit, as shown in Figure 1–13(b) for the example of eight bits being transferred. To transfer eight bits in parallel, it takes one time interval compared to eight time intervals for the serial transfer.

To summarize, an advantage of serial transfer of binary data is that a minimum of only one line is required. In parallel transfer, a number of lines equal to the number of bits to be transferred at one time is required. A disadvantage of serial transfer is that it takes longer to transfer a given number of bits than with parallel transfer at the same clock frequency. For example, if one bit can be transferred in 1 μs, then it takes 8 μs to serially transfer eight bits but only 1 μs to parallel transfer eight bits. A disadvantage of parallel transfer is that it takes more lines than serial transfer.

EXAMPLE 1–2

(a) Determine the total time required to serially transfer the eight bits contained in waveform *A* of Figure 1–14, and indicate the sequence of bits. The left-most bit is the first to be transferred. The 1 MHz clock is used as reference.

(b) What is the total time to transfer the same eight bits in parallel?

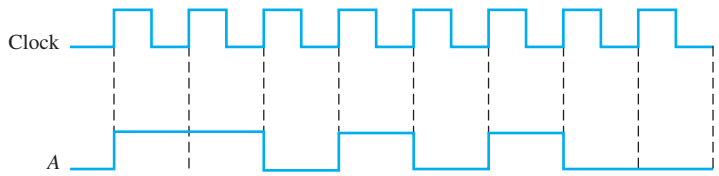


FIGURE 1–14

Solution

(a) Since the frequency of the clock is 1 MHz, the period is

$$T = \frac{1}{f} = \frac{1}{1\text{ MHz}} = 1\ \mu\text{s}$$

It takes 1 μs to transfer each bit in the waveform. The total transfer time for 8 bits is

$$8 \times 1\ \mu\text{s} = \mathbf{8\ \mu s}$$

To determine the sequence of bits, examine the waveform in Figure 1–14 during each bit time. If waveform *A* is HIGH during the bit time, a 1 is transferred. If waveform *A* is LOW during the bit time, a 0 is transferred. The bit sequence is illustrated in Figure 1–15. The left-most bit is the first to be transferred.

FIGURE 1–15

(b) A parallel transfer would take **1 μs** for all eight bits.

Related Problem

If binary data are transferred on a USB at the rate of 480 million bits per second (480 Mbps), how long will it take to serially transfer 16 bits?

SECTION 1–2 CHECKUP

1. Define *binary.*
2. What does *bit* mean?
3. What are the bits in a binary system?
4. How are the rise time and fall time of a pulse measured?
5. Knowing the period of a waveform, how do you find the frequency?
6. Explain what a clock waveform is.
7. What is the purpose of a timing diagram?
8. What is the main advantage of parallel transfer over serial transfer of binary data?

1–3 Basic Logic Functions

In its basic form, logic is the realm of human reasoning that tells you a certain proposition (declarative statement) is true if certain conditions are true. Propositions can be classified as true or false. Many situations and processes that you encounter in your daily life can be expressed in the form of propositional, or logic, functions. Since such functions are true/false or yes/no statements, digital circuits with their two-state characteristics are applicable.

After completing this section, you should be able to

- List three basic logic functions
- Define the NOT function
- Define the AND function
- Define the OR function

Several propositions, when combined, form propositional, or logic, functions. For example, the propositional statement "The light is on" will be true if "The bulb is not burned out" is true and if "The switch is on" is true. Therefore, this logical statement can be made: *The light is on only if the bulb is not burned out and the switch is on.* In this example the first statement is true only if the last two statements are true. The first statement ("The light is on")

is then the basic proposition, and the other two statements are the conditions on which the proposition depends.

In the 1850s, the Irish logician and mathematician George Boole developed a mathematical system for formulating logic statements with symbols so that problems can be written and solved in a manner similar to ordinary algebra. Boolean algebra, as it is known today, is applied in the design and analysis of digital systems and will be covered in detail in Chapter 4.

The term **logic** is applied to digital circuits used to implement logic functions. Several kinds of digital logic **circuits** are the basic elements that form the building blocks for such complex digital systems as the computer. We will now look at these elements and discuss their functions in a very general way. Later chapters will cover these circuits in detail.

Three basic logic functions (NOT, AND, and OR) are indicated by standard distinctive shape symbols in Figure 1–16. Alternate standard symbols for these logic functions will be introduced in Chapter 3. The lines connected to each symbol are the **inputs** and **outputs**. The inputs are on the left of each symbol and the output is on the right. A circuit that performs a specified logic function (AND, OR) is called a logic **gate**. AND and OR gates can have any number of inputs, as indicated by the dashes in the figure.

FIGURE 1–16 The basic logic functions and symbols.

In logic functions, the true/false conditions mentioned earlier are represented by a HIGH (true) and a LOW (false). Each of the three basic logic functions produces a unique response to a given set of conditions.

NOT

The **NOT** function changes one logic level to the opposite logic level, as indicated in Figure 1–17. When the input is HIGH (1), the output is LOW (0). When the input is LOW, the output is HIGH. In either case, the output is *not* the same as the input. The NOT function is implemented by a logic circuit known as an **inverter**.

FIGURE 1–17 The NOT function.

AND

The **AND** function produces a HIGH output only when all the inputs are HIGH, as indicated in Figure 1–18 for the case of two inputs. When one input is HIGH *and* the other input is HIGH, the output is HIGH. When any or all inputs are LOW, the output is LOW. The AND function is implemented by a logic circuit known as an *AND gate.*

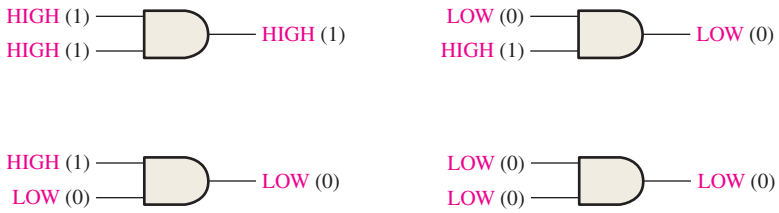


FIGURE 1–18 The AND function.

OR

The **OR** function produces a HIGH output when one or more inputs are HIGH, as indicated in Figure 1–19 for the case of two inputs. When one input is HIGH *or* the other input is HIGH *or* both inputs are HIGH, the output is HIGH. When both inputs are LOW, the output is LOW. The OR function is implemented by a logic circuit known as an *OR gate*.

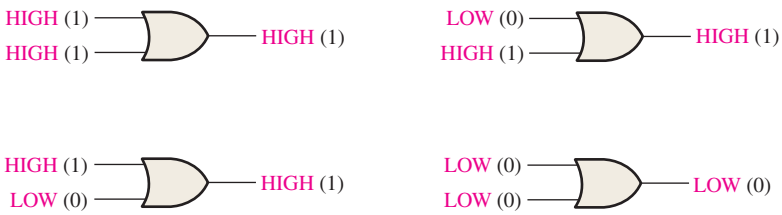


FIGURE 1–19 The OR function.

SECTION 1–3 CHECKUP

1. When does the NOT function produce a HIGH output?
2. When does the AND function produce a HIGH output?
3. When does the OR function produce a HIGH output?
4. What is an inverter?
5. What is a logic gate?

1–4 Combinational and Sequential Logic Functions

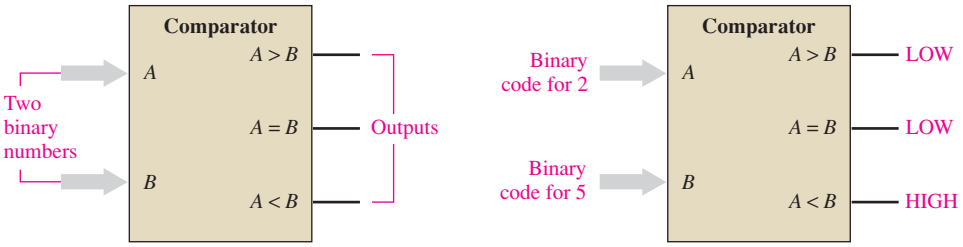
The three basic logic functions AND, OR, and NOT can be combined to form various other types of more complex logic functions, such as comparison, arithmetic, code conversion, encoding, decoding, data selection, counting, and storage. A digital system is an arrangement of the individual logic functions connected to perform a specified operation or produce a defined output. This section provides an overview of important logic functions and illustrates how they can be used in a specific system.

After completing this section, you should be able to

- List several types of logic functions
- Describe comparison and list the four arithmetic functions
- Describe code conversion, encoding, and decoding
- Describe multiplexing and demultiplexing
- Describe the counting function
- Describe the storage function
- Explain the operation of the tablet-bottling system

The Comparison Function

Magnitude comparison is performed by a logic circuit called a **comparator**, covered in Chapter 6. A comparator compares two quantities and indicates whether or not they are equal. For example, suppose you have two numbers and wish to know if they are equal or not equal and, if not equal, which is greater. The comparison function is represented in



(a) Basic magnitude comparator

(b) Example: A is less than B (2 < 5) as indicated by the HIGH output ($A < B$)

FIGURE 1–20 The comparison function.

Figure 1–20. One number in binary form (represented by logic levels) is applied to input A, and the other number in binary form (represented by logic levels) is applied to input B. The outputs indicate the relationship of the two numbers by producing a HIGH level on the proper output line. Suppose that a binary representation of the number 2 is applied to input A and a binary representation of the number 5 is applied to input B. (The binary representation of numbers and symbols is discussed in Chapter 2.) A HIGH level will appear on the $A < B$ (A is less than B) output, indicating the relationship between the two numbers (2 is less than 5). The wide arrows represent a group of parallel lines on which the bits are transferred.

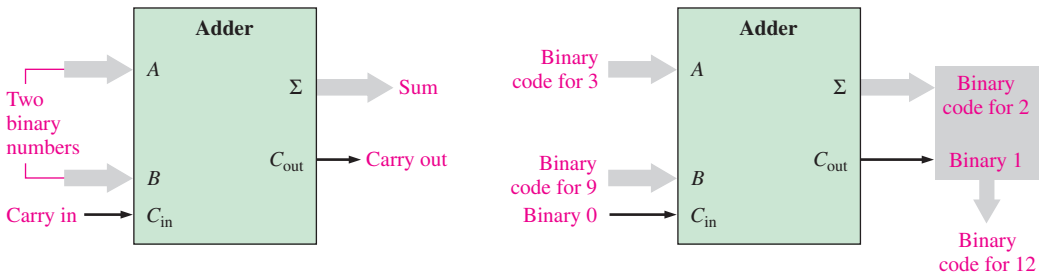
InfoNote

In a microprocessor, the arithmetic logic unit (ALU) performs the operations of add, subtract, multiply, and divide as well as the logic operations on digital data as directed by a series of instructions. A typical ALU is constructed of many thousands of logic gates.

The Arithmetic Functions

Addition

Addition is performed by a logic circuit called an **adder**, covered in Chapter 6. An adder adds two binary numbers (on inputs A and B with a carry input C_{in}) and generates a sum (Σ) and a carry output (C_{out}), as shown in Figure 1–21(a). Figure 1–21(b) illustrates the addition of 3 and 9. You know that the sum is 12; the adder indicates this result by producing 2 on the sum output and 1 on the carry output. Assume that the carry input in this example is 0.



(a) Basic adder

(b) Example: A plus B (3 + 9 = 12)

FIGURE 1–21 The addition function.

Subtraction

Subtraction is also performed by a logic circuit. A **subtracter** requires three inputs: the two numbers that are to be subtracted and a borrow input. The two outputs are the difference and the borrow output. When, for instance, 5 is subtracted from 8 with no borrow input, the difference is 3 with no borrow output. You will see in Chapter 2 how subtraction can actually be performed by an adder because subtraction is simply a special case of addition.

Multiplication

Multiplication is performed by a logic circuit called a *multiplier.* Numbers are always multiplied two at a time, so two inputs are required. The output of the multiplier is the product. Because multiplication is simply a series of additions with shifts in the positions of the partial products, it can be performed by using an adder in conjunction with other circuits.

Division

Division can be performed with a series of subtractions, comparisons, and shifts, and thus it can also be done using an adder in conjunction with other circuits. Two inputs to the divider are required, and the outputs generated are the quotient and the remainder.

The Code Conversion Function

A **code** is a set of bits arranged in a unique pattern and used to represent specified information. A code converter changes one form of coded information into another coded form. Examples are conversion between binary and other codes such as the binary coded decimal (BCD) and the Gray code. Various types of codes are covered in Chapter 2, and code converters are covered in Chapter 6.

The Encoding Function

The encoding function is performed by a logic circuit called an **encoder**, covered in Chapter 6. The encoder converts information, such as a decimal number or an alphabetic character, into some coded form. For example, one certain type of encoder converts each of the decimal digits, 0 through 9, to a binary code. A HIGH level on the input corresponding to a specific decimal digit produces logic levels that represent the proper binary code on the output lines.

Figure 1–22 is a simple illustration of an encoder used to convert (encode) a calculator keystroke into a binary code that can be processed by the calculator circuits.

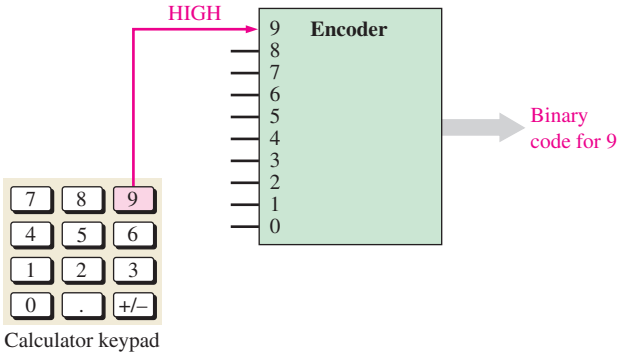


FIGURE 1–22 An encoder used to encode a calculator keystroke into a binary code for storage or for calculation.

The Decoding Function

The decoding function is performed by a logic circuit called a **decoder**, covered in Chapter 6. The decoder converts coded information, such as a binary number, into a noncoded form, such as a decimal form. For example, one particular type of decoder converts a 4-bit binary code into the appropriate decimal digit.

Figure 1–23 is a simple illustration of one type of decoder that is used to activate a 7-segment display. Each of the seven segments of the display is connected to an output line from the decoder. When a particular binary code appears on the decoder inputs, the appropriate output lines are activated and light the proper segments to display the decimal digit corresponding to the binary code.

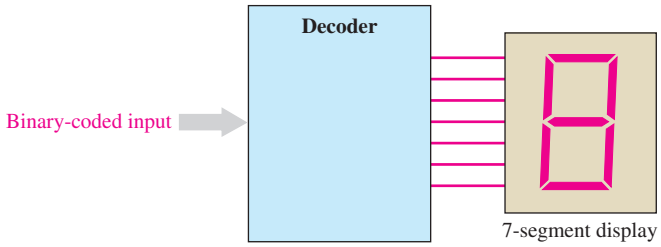


FIGURE 1–23 A decoder used to convert a special binary code into a 7-segment decimal readout.

The Data Selection Function

Two types of circuits that select data are the multiplexer and the demultiplexer. The **multiplexer**, or mux for short, is a logic circuit that switches digital data from several input lines onto a single output line in a specified time sequence. Functionally, a multiplexer can be represented by an electronic switch operation that sequentially connects each of the input lines to the output line. The **demultiplexer** (demux) is a logic circuit that switches digital data from one input line to several output lines in a specified time sequence. Essentially, the demux is a mux in reverse.

Multiplexing and demultiplexing are used when data from several sources are to be transmitted over one line to a distant location and redistributed to several destinations. Figure 1–24 illustrates this type of application where digital data from three sources are sent out along a single line to three terminals at another location.

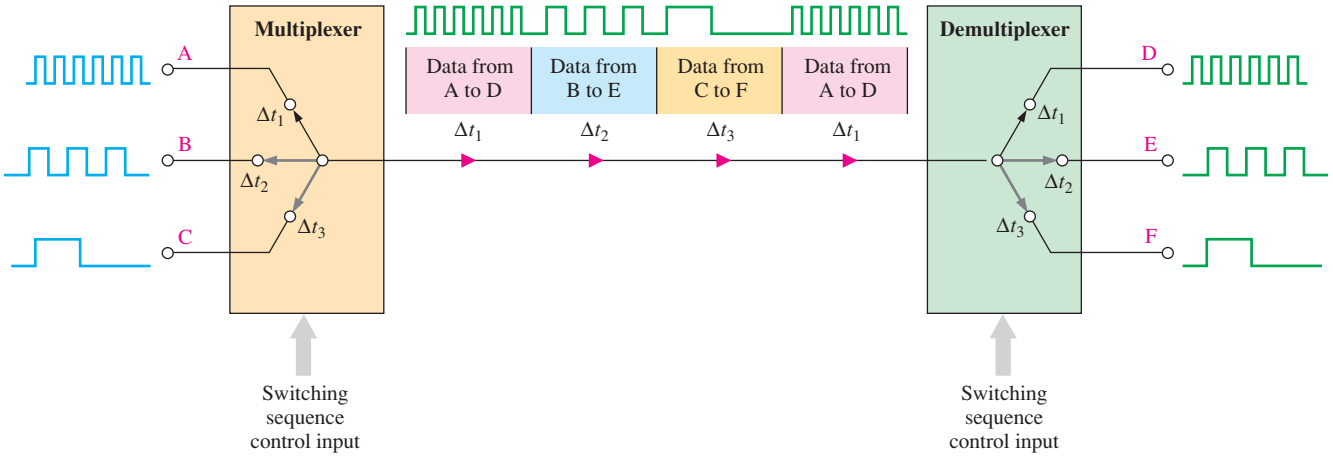


FIGURE 1–24 Illustration of a basic multiplexing/demultiplexing application.

In Figure 1–24, data from input A are connected to the output line during time interval Δt_1 and transmitted to the demultiplexer that connects them to output D. Then, during interval Δt_2, the multiplexer switches to input B and the demultiplexer switches to output E. During interval Δt_3, the multiplexer switches to input C and the demultiplexer switches to output F.

To summarize, during the first time interval, input A data go to output D. During the second time interval, input B data go to output E. During the third time interval, input C data go to output F. After this, the sequence repeats. Because the time is divided up among several sources and destinations where each has its turn to send and receive data, this process is called *time division multiplexing* (TDM).

InfoNote

The internal computer memories, RAM and ROM, as well as the smaller caches are semiconductor memories. The registers in a microprocessor are constructed of semiconductor flip-flops. Optomagnetic disk memories are used in the internal hard drive and for the CD-ROM.

The Storage Function

Storage is a function that is required in most digital systems, and its purpose is to retain binary data for a period of time. Some storage devices are used for short-term storage and some