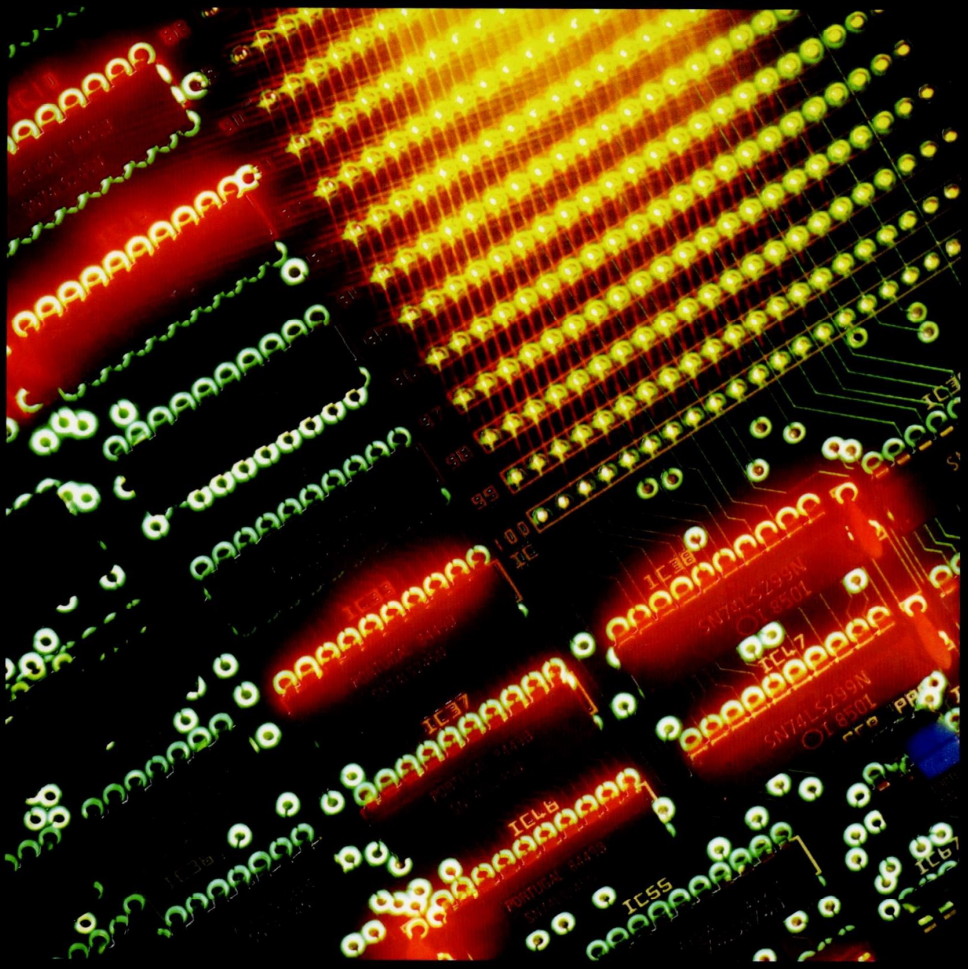
TENTH EDITION

# DIGITAL SYSTEMS

## PRINCIPLES AND APPLICATIONS



SOFTWARE ENCLOSED

RONALD J. TOCCI
NEAL S. WIDMER
GREGORY L. MOSS



ANKER 1009


TOCCI
WIDMER
MOSS
DIGITAL SYSTEMS
PRINCIPLES AND APPLICATIONS
TENTH
EDITION
TK7868
.D5
T62
2007

RONALD J. TOCCI
NEAL S. WIDMER
GREGORY L. MOSS

# DIGITAL SYSTEMS

TENTH EDITION

## PRINCIPLES AND APPLICATIONS

Prentice Hall and the successful author team of Widmer, Moss, and Tocci are proud to present the 10th edition of this popular text, written for engineering and technology students. For this new edition, the authors have meticulously worked to provide the right balance between existing and new material while keeping the size of the book within reason. This is a growing challenge due to revolutionary digital technology. Industry's movement from using schematics to using hardware description languages (HDLs) to describe complex digital systems has rendered obsolete many topics previously considered to be foundational. In addition, new technology demands the expansion and emphasis of other traditional concepts as it introduces many new tools and techniques for developing and analyzing digital systems.

***Uncompromised!*** The 10th edition of this classic text continues to offer all of the necessary coverage for a solid foundation in Digital Electronics, retaining all of the following:

- Clear and thorough presentation of all the vital principles that have proven effective with students for decades
- Coverage of conventional, standard IC technology to allow a smooth transition to current digital technologies
- The greatest amount of flexibility for the depth and breadth of individual courses
- The capacity to be an excellent reference resource in settings outside the classroom
- Techniques for applying HDLs to develop digital circuits and subsystems at the end of appropriate chapters
- Coverage of Altera's AHDL, a powerful introductory-level language that is great for beginners
- Thorough introduction to VHDL, an industry-standard design language
- FREE! Easy-to-use MAX+PLUS® II *with schematic capture, simulator, and HDL* on the enclosed CD

***NEW in this edition!***

- Improved analysis of combinational circuits
- Expanded coverage of the 555 timer in Chapter 5
- Improved coverage of signed numbers in Chapter 6
- Greater emphasis on synchronous counters in Chapter 7
- More thorough coverage of state machines, with a practical example of a functional system
- Description of recent IC technology in Chapter 8
- Revised and improved AHDL and VHDL coverage
- FREE! Powerful new Quartus® II web edition software on the enclosed CD

**For information about additional Prentice Hall Electronics Technology resources, visit** www.prenhall.com

StudentAid.ed.gov
FUNDING YOUR FUTURE

PEARSON Prentice Hall






TENTH EDITION

# Digital Systems

## Principles and Applications

**Ronald J. Tocci**
Monroe Community College

**Neal S. Widmer**
Purdue University

**Gregory L. Moss**
Purdue University




Upper Saddle River, New Jersey
Columbus, Ohio

**Library of Congress Cataloging-in-Publication Data**

Tocci, Ronald J.
Digital systems : principles and applications / Ronald J. Tocci, Neal S. Widmer, Gregory L. Moss.—10th ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-13-172579-3
1. Digital electronics—Textbooks. I. Widmer, Neal S. II. Moss, Gregory L. III. Title.
TK7868.D5T62 2007
621.381—dc22

2005035835

**Director of Development:** Vern Anthony
**Editorial Assistant:** Lara Dimmick
**Production Editor:** Stephen C. Robb
**Production Coordination:** Peggy Hood, TechBooks/GTS
**Design Coordinator:** Diane Y. Ernsberger
**Cover Designer:** Jason Moore
**Cover Art:** Getty One
**Production Manager:** Matt Ottenweller
**Marketing Manager:** Ben Leonard

This book was set in TimesEuropa Roman by *TechBooks/GTS* York, PA Campus. It was printed and bound by Courier Kendallville, Inc. The cover was printed by Phoenix Color Corp.

MultiSIM® is a trademark of Electronics Workbench.

Altera is a trademark and service mark of Altera Corporation in the United States and other countries. Altera products are the intellectual property of Altera Corporation and are protected by copyright laws and one or more U.S. and foreign patents and patent applications.

---

**Copyright © 2007, 2004, 2001, 1998, 1995, 1991, 1988, 1985, 1980, 1970 by Pearson Education, Inc., Upper Saddle River, New Jersey 07458.** Pearson Prentice Hall. All rights reserved. Printed in the United States of America. This publication is protected by Copyright and permission should be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permission(s), write to: Rights and Permissions Department.

**Pearson Prentice Hall™** is a trademark of Pearson Education, Inc.
**Pearson®** is a registered trademark of Pearson plc
**Prentice Hall®** is a registered trademark of Pearson Education, Inc.

Pearson Education Ltd.
Pearson Education Singapore, Pte. Ltd.
Pearson Education Canada, Ltd.
Pearson Education—Japan
Pearson Education Australia Pty. Limited
Pearson Education North Asia Ltd.
Pearson Educación de Mexico, S.A. de C.V.
Pearson Education Malaysia, Pte. Ltd.




10 9 8 7 6 5 4 3 2 1
ISBN: 0-13-172579-3

# BRIEF CONTENTS


| | | |
|---|---|---|
| CHAPTER 1 | Introductory Concepts | 2 |
| CHAPTER 2 | Number Systems and Codes | 24 |
| CHAPTER 3 | Describing Logic Circuits | 54 |
| CHAPTER 4 | Combinational Logic Circuits | 118 |
| CHAPTER 5 | Flip-Flops and Related Devices | 208 |
| CHAPTER 6 | Digital Arithmetic: Operations and Circuits | 296 |
| CHAPTER 7 | Counters and Registers | 360 |
| CHAPTER 8 | Integrated-Circuit Logic Families | 488 |
| CHAPTER 9 | MSI Logic Circuits | 576 |
| CHAPTER 10 | Digital System Projects Using HDL | 676 |
| CHAPTER 11 | Interfacing with the Analog World | 718 |
| CHAPTER 12 | Memory Devices | 786 |
| CHAPTER 13 | Programmable Logic Device Architectures | 868 |
| | Glossary | 898 |
| | Answers to Selected Problems | 911 |
| | Index of ICs | 919 |
| | Index | 922 |

# CONTENTS


## CHAPTER 1 Introductory Concepts 2

1-1 Numerical Representations 4
1-2 Digital and Analog Systems 5
1-3 Digital Number Systems 10
1-4 Representing Binary Quantities 13
1-5 Digital Circuits/Logic Circuits 15
1-6 Parallel and Serial Transmission 17
1-7 Memory 18
1-8 Digital Computers 19

## CHAPTER 2 Number Systems and Codes 24

2-1 Binary-to-Decimal Conversions 26
2-2 Decimal-to-Binary Conversions 26
2-3 Hexadecimal Number System 29
2-4 BCD Code 33
2-5 The Gray Code 35
2-6 Putting It All Together 37
2-7 The Byte, Nibble, and Word 37
2-8 Alphanumeric Codes 39
2-9 Parity Method for Error Detection 41
2-10 Applications 44

**Chapter 3 Describing Logic Circuits 54**

3-1 Boolean Constants and Variables 57
3-2 Truth Tables 57
3-3 OR Operation with OR Gates 58
3-4 AND Operation with AND Gates 62
3-5 NOT Operation 65
3-6 Describing Logic Circuits Algebraically 66
3-7 Evaluating Logic-Circuit Outputs 68
3-8 Implementing Circuits from Boolean Expressions 71
3-9 NOR Gates and NAND Gates 73
3-10 Boolean Theorems 76
3-11 DeMorgan's Theorems 80
3-12 Universality of NAND Gates and NOR Gates 83
3-13 Alternate Logic-Gate Representations 86
3-14 Which Gate Representation to Use 89
3-15 IEEE/ANSI Standard Logic Symbols 95
3-16 Summary of Methods to Describe Logic Circuits 96
3-17 Description Languages Versus Programming Languages 98
3-18 Implementing Logic Circuits with PLDs 100
3-19 HDL Format and Syntax 102
3-20 Intermediate Signals 105

**Chapter 4 Combinational Logic Circuits 118**

4-1 Sum-of-Products Form 120
4-2 Simplifying Logic Circuits 121
4-3 Algebraic Simplification 121
4-4 Designing Combinational Logic Circuits 127
4-5 Karnaugh Map Method 133
4-6 Exclusive-OR and Exclusive-NOR Circuits 144
4-7 Parity Generator and Checker 149
4-8 Enable/Disable Circuits 151
4-9 Basic Characteristics of Digital ICs 153
4-10 Troubleshooting Digital Systems 160
4-11 Internal Digital IC Faults 162
4-12 External Faults 166
4-13 Troubleshooting Case Study 168
4-14 Programmable Logic Devices 170
4-15 Representing Data in HDL 177
4-16 Truth Tables Using HDL 181
4-17 Decision Control Structures in HDL 184

## Chapter 5 Flip-Flops and Related Devices 208

5-1 NAND Gate Latch 211
5-2 NOR Gate Latch 216
5-3 Troubleshooting Case Study 219
5-4 Digital Pulses 220
5-5 Clock Signals and Clocked Flip-Flops 221
5-6 Clocked S-R Flip-Flop 224
5-7 Clocked J-K Flip-Flop 227
5-8 Clocked D Flip-Flop 230
5-9 *D* Latch (Transparent Latch) 232
5-10 Asynchronous Inputs 233
5-11 IEEE/ANSI Symbols 236
5-12 Flip-Flop Timing Considerations 238
5-13 Potential Timing Problem in FF Circuits 241
5-14 Flip-Flop Applications 243
5-15 Flip-Flop Synchronization 243
5-16 Detecting an Input Sequence 244
5-17 Data Storage and Transfer 245
5-18 Serial Data Transfer: Shift Registers 247
5-19 Frequency Division and Counting 250
5-20 Microcomputer Application 254
5-21 Schmitt-Trigger Devices 256
5-22 One-Shot (Monostable Multivibrator) 256
5-23 Clock Generator Circuits 260
5-24 Troubleshooting Flip-Flop Circuits 264
5-25 Sequential Circuits Using HDL 268
5-26 Edge-Triggered Devices 272
5-27 HDL Circuits with Multiple Components 277

## Chapter 6 Digital Arithmetic: Operations and Circuits 296

6-1 Binary Addition 298
6-2 Representing Signed Numbers 299
6-3 Addition in the 2's-Complement System 306
6-4 Subtraction in the 2's-Complement System 307
6-5 Multiplication of Binary Numbers 310
6-6 Binary Division 311
6-7 BCD Addition 312
6-8 Hexadecimal Arithmetic 314
6-9 Arithmetic Circuits 317
6-10 Parallel Binary Adder 318
6-11 Design of a Full Adder 320

6-12 Complete Parallel Adder with Registers 323
6-13 Carry Propagation 325
6-14 Integrated-Circuit Parallel Adder 326
6-15 2's-Complement System 328
6-16 ALU Integrated Circuits 331
6-17 Troubleshooting Case Study 335
6-18 Using TTL Library Functions with HDL 337
6-19 Logical Operations on Bit Arrays 338
6-20 HDL Adders 340
6-21 Expanding the Bit Capacity of a Circuit 343

**Chapter 7 Counters and Registers 360**

7-1 Asynchronous (Ripple) Counters 362
7-2 Propagation Delay in Ripple Counters 365
7-3 Synchronous (Parallel) Counters 367
7-4 Counters with MOD Numbers $<2^N$ 370
7-5 Synchronous Down and Up/Down Counters 377
7-6 Presettable Counters 379
7-7 IC Synchronous Counters 380
7-8 Decoding a Counter 389
7-9 Analyzing Synchronous Counters 393
7-10 Synchronous Counter Design 396
7-11 Basic Counters Using HDLs 405
7-12 Full-Featured Counters in HDL 412
7-13 Wiring HDL Modules Together 417
7-14 State Machines 425
7-15 Integrated-Circuit Registers 437
7-16 Parallel In/Parallel Out—The 74ALS174/74HC174 437
7-17 Serial In/Serial Out—The 74ALS166/74HC166 439
7-18 Parallel In/Serial Out—The 74ALS165/74HC165 441
7-19 Serial In/Parallel Out—The 74ALS164/74HC164 443
7-20 Shift-Register Counters 445
7-21 Troubleshooting 450
7-22 HDL Registers 452
7-23 HDL Ring Counters 459
7-24 HDL One-Shots 461

**Chapter 8 Integrated-Circuit Logic Families 488**

8-1 Digital IC Terminology 490
8-2 The TTL Logic Family 498
8-3 TTL Data Sheets 502
8-4 TTL Series Characteristics 506

8-5 TTL Loading and Fan-Out 509
8-6 Other TTL Characteristics 514
8-7 MOS Technology 518
8-8 Complementary MOS Logic 521
8-9 CMOS Series Characteristics 523
8-10 Low-Voltage Technology 530
8-11 Open-Collector/Open-Drain Outputs 533
8-12 Tristate (Three-State) Logic Outputs 538
8-13 High-Speed Bus Interface Logic 541
8-14 The ECL Digital IC Family 543
8-15 CMOS Transmission Gate (Bilateral Switch) 546
8-16 IC Interfacing 548
8-17 Mixed-Voltage Interfacing 553
8-18 Analog Voltage Comparators 554
8-19 Troubleshooting 556

**Chapter 9 MSI Logic Circuits 576**

9-1 Decoders 577
9-2 BCD-to-7-Segment Decoder/Drivers 584
9-3 Liquid-Crystal Displays 587
9-4 Encoders 591
9-5 Troubleshooting 597
9-6 Multiplexers (Data Selectors) 599
9-7 Multiplexer Applications 604
9-8 Demultiplexers (Data Distributors) 610
9-9 More Troubleshooting 617
9-10 Magnitude Comparator 621
9-11 Code Converters 624
9-12 Data Busing 628
9-13 The 74ALS173/HC173 Tristate Register 629
9-14 Data Bus Operation 632
9-15 Decoders Using HDL 638
9-16 The HDL 7-Segment Decoder/Driver 642
9-17 Encoders Using HDL 645
9-18 HDL Multiplexers and Demultiplexers 648
9-19 HDL Magnitude Comparators 652
9-20 HDL Code Converters 653

**Chapter 10 Digital System Projects Using HDL 676**

10-1 Small-Project Management 678
10-2 Stepper Motor Driver Project 679
10-3 Keypad Encoder Project 687

10-4 Digital Clock Project 693
10-5 Frequency Counter Project 710

**Chapter 11 Interfacing with the Analog World 718**

11-1 Review of Digital Versus Analog 719
11-2 Digital-to-Analog Conversion 721
11-3 D/A-Converter Circuitry 728
11-4 DAC Specifications 733
11-5 An Integrated-Circuit DAC 735
11-6 DAC Applications 736
11-7 Troubleshooting DACs 738
11-8 Analog-to-Digital Conversion 739
11-9 Digital-Ramp ADC 740
11-10 Data Acquisition 745
11-11 Successive-Approximation ADC 749
11-12 Flash ADCs 755
11-13 Other A/D Conversion Methods 757
11-14 Sample-and-Hold Circuits 761
11-15 Multiplexing 762
11-16 Digital Storage Oscilloscope 764
11-17 Digital Signal Processing (DSP) 765

**Chapter 12 Memory Devices 784**

12-1 Memory Terminology 786
12-2 General Memory Operation 790
12-3 CPU–Memory Connections 793
12-4 Read-Only Memories 795
12-5 ROM Architecture 796
12-6 ROM Timing 799
12-7 Types of ROMs 800
12-8 Flash Memory 808
12-9 ROM Applications 811
12-10 Semiconductor RAM 814
12-11 RAM Architecture 815
12-12 Static RAM (SRAM) 818
12-13 Dynamic RAM (DRAM) 823
12-14 Dynamic RAM Structure and Operation 824
12-15 DRAM Read/Write Cycles 829
12-16 DRAM Refreshing 831
12-17 DRAM Technology 834
12-18 Expanding Word Size and Capacity 836
12-19 Special Memory Functions 844

12-20 Troubleshooting RAM Systems 847
12-21 Testing ROM 852

**Chapter 13 Programmable Logic Device Architectures 868**

13-1 Digital Systems Family Tree 870
13-2 Fundamentals of PLD Circuitry 875
13-3 PLD Architectures 877
13-4 The GAL 16V8 (Generic Array Logic) 881
13-5 The Altera EPM7128S CPLD 885
13-6 The Altera FLEX10K Family 890
13-7 The Altera Cyclone Family 894

**Glossary 898**

**Answers to Selected Problems 911**

**Index of ICs 919**

**Index 922**

# Digital Systems

## Principles and Applications

often electronic, but they can also be mechanical, magnetic, or pneumatic. Some of the more familiar digital systems include digital computers and calculators, digital audio and video equipment, and the telephone system—the world's largest digital system.

An **analog system** contains devices that manipulate physical quantities that are represented in analog form. In an analog system, the quantities can vary over a continuous range of values. For example, the amplitude of the output signal to the speaker in a radio receiver can have any value between zero and its maximum limit. Other common analog systems are audio amplifiers, magnetic tape recording and playback equipment, and a simple light dimmer switch.

## Advantages of Digital Techniques

An increasing majority of applications in electronics, as well as in most other technologies, use digital techniques to perform operations that were once performed using analog methods. The chief reasons for the shift to digital technology are:

1. *Digital systems are generally easier to design.* The circuits used in digital systems are *switching circuits,* where *exact* values of voltage or current are not important, only the range (HIGH or LOW) in which they fall.
2. *Information storage is easy.* This is accomplished by special devices and circuits that can latch onto digital information and hold it for as long as necessary, and mass storage techniques that can store billions of bits of information in a relatively small physical space. Analog storage capabilities are, by contrast, extremely limited.
3. *Accuracy and precision are easier to maintain throughout the system.* Once a signal is digitized, the information it contains does not deteriorate as it is processed. In analog systems, the voltage and current signals tend to be distorted by the effects of temperature, humidity, and component tolerance variations in the circuits that process the signal.
4. *Operation can be programmed.* It is fairly easy to design digital systems whose operation is controlled by a set of stored instructions called a *program.* Analog systems can also be *programmed,* but the variety and the complexity of the available operations are severely limited.
5. *Digital circuits are less affected by noise.* Spurious fluctuations in voltage (noise) are not as critical in digital systems because the exact value of a voltage is not important, as long as the noise is not large enough to prevent us from distinguishing a HIGH from a LOW.
6. *More digital circuitry can be fabricated on IC chips.* It is true that analog circuitry has also benefited from the tremendous development of IC technology, but its relative complexity and its use of devices that cannot be economically integrated (high-value capacitors, precision resistors, inductors, transformers) have prevented analog systems from achieving the same high degree of integration.

## Limitations of Digital Techniques

There are really very few drawbacks when using digital techniques. The two biggest problems are:

**The real world is analog.**
**Processing digitized signals takes time.**

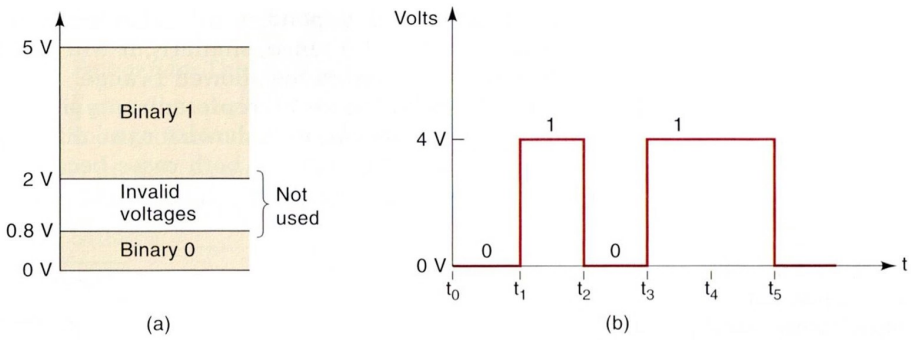



**FIGURE 1-8** (a) Typical voltage assignments in digital system; (b) typical digital signal timing diagram.

for example, for the voltage assignments of Figure 1-8(a), a voltage of 3.6 V means the same as a voltage of 4.3 V. In analog systems, the exact value of a voltage *is* important. For instance, if the analog voltage is proportional to the temperature measured by a transducer, the 3.6 V would represent a different temperature than would 4.3 V. In other words, the voltage value carries significant information. This characteristic means that the design of accurate analog circuitry is generally more difficult than that of digital circuitry because of the way in which exact voltage values are affected by variations in component values, temperature, and noise (random voltage fluctuations).

## Digital Signals and Timing Diagrams

Figure 1-8(b) shows a typical digital signal and how it varies over time. It is actually a graph of voltage versus time *(t)* and is called a **timing diagram**. The horizontal time scale is marked off at regular intervals beginning at $t_0$ and proceeding to $t_1$, $t_2$, and so on. For the example timing diagram shown here, the signal starts at 0 V (a binary 0) at time $t_0$ and remains there until time $t_1$. At $t_1$, the signal makes a rapid transition (jump) up to 4 V (a binary 1). At $t_2$, it jumps back down to 0 V. Similar transitions occur at $t_3$ and $t_5$. Note that the signal does not change at $t_4$ but stays at 4 V from $t_3$ to $t_5$.

The transitions on this timing diagram are drawn as vertical lines, and so they appear to be instantaneous, when in reality they are not. In many situations, however, the transition times are so short compared to the times between transitions that we can show them on the diagram as vertical lines. We will encounter situations later where it will be necessary to show the transitions more accurately on an expanded time scale.

Timing diagrams are used extensively to show how digital signals change with time, and especially to show the relationship between two or more digital signals in the same circuit or system. By displaying one or more digital signals on an *oscilloscope* or *logic analyzer*, we can compare the signals to their expected timing diagrams. This is a very important part of the testing and troubleshooting procedures used in digital systems.

# 1-5 DIGITAL CIRCUITS/LOGIC CIRCUITS

**Digital circuits** are designed to produce output voltages that fall within the prescribed 0 and 1 voltage ranges such as those defined in Figure 1-8. Likewise, digital circuits are designed to respond predictably to input voltages that are within the defined 0 and 1 ranges. What this means is that a

digital circuit will respond in the same way to all input voltages that fall within the allowed 0 range; similarly, it will not distinguish between input voltages that lie within the allowed 1 range.

To illustrate, Figure 1-9 represents a typical digital circuit with input $v_i$ and output $v_o$. The output is shown for two different input signal waveforms. Note that $v_o$ is the same for both cases because the two input waveforms, while differing in their exact voltage levels, are at the same binary levels.

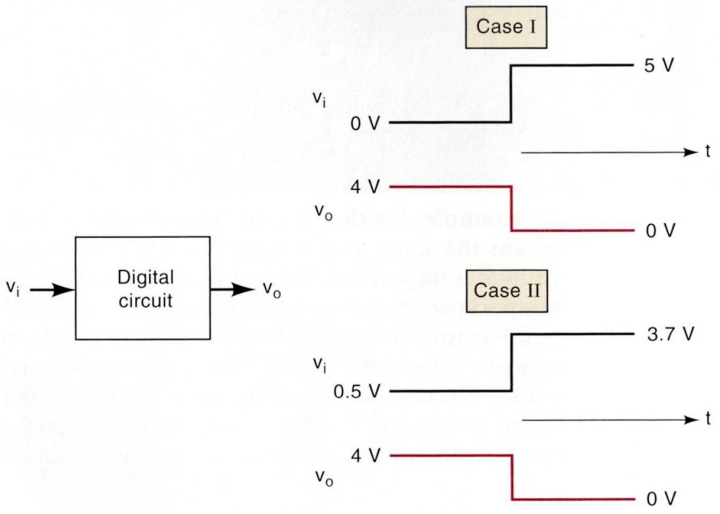



**FIGURE 1-9** A digital circuit responds to an input's binary level (0 or 1) and not to its actual voltage.

## Logic Circuits

The manner in which a digital circuit responds to an input is referred to as the circuit's *logic*. Each type of digital circuit obeys a certain set of logic rules. For this reason, digital circuits are also called **logic circuits**. We will use both terms interchangeably throughout the text. In Chapter 3, we will see more clearly what is meant by a circuit's "logic."

We will be studying all the types of logic circuits that are currently used in digital systems. Initially, our attention will be focused only on the logical operation that these circuits perform—that is, the relationship between the circuit inputs and outputs. We will defer any discussion of the internal circuit operation of these logic circuits until after we have developed an understanding of their logical operation.

## Digital Integrated Circuits

Almost all of the digital circuits used in modern digital systems are integrated circuits (ICs). The wide variety of available logic ICs has made it possible to construct complex digital systems that are smaller and more reliable than their discrete-component counterparts.

Several integrated-circuit fabrication technologies are used to produce digital ICs, the most common being CMOS, TTL, NMOS, and ECL. Each differs in the type of circuitry used to provide the desired logic operation. For example, TTL (transistor-transistor logic) uses the bipolar transistor as its main circuit element, while CMOS (complementary metal-oxide-semiconductor) uses the enhancement-mode MOSFET as its principal circuit element. We will learn about the various IC technologies, their characteristics, and their relative advantages and disadvantages after we master the basic logic circuit types.

## 3-1 BOOLEAN CONSTANTS AND VARIABLES

Boolean algebra differs in a major way from ordinary algebra because Boolean constants and variables are allowed to have only two possible values, 0 or 1. A Boolean variable is a quantity that may, at different times, be equal to either 0 or 1. Boolean variables are often used to represent the voltage level present on a wire or at the input/output terminals of a circuit. For example, in a certain digital system, the Boolean value of 0 might be assigned to any voltage in the range from 0 to 0.8 V, while the Boolean value of 1 might be assigned to any voltage in the range 2 to 5 V.*

Thus, Boolean 0 and 1 do not represent actual numbers but instead represent the state of a voltage variable, or what is called its **logic level**. A voltage in a digital circuit is said to be at the logic 0 level or the logic 1 level, depending on its actual numerical value. In digital logic, several other terms are used synonymously with 0 and 1. Some of the more common ones are shown in Table 3-1. We will use the 0/1 and LOW/HIGH designations most of the time.

**TABLE 3-1**

| Logic 0 | Logic 1 |
|---|---|
| False | True |
| Off | On |
| Low | High |
| No | Yes |
| Open switch | Closed switch |

As we said in the introduction, **Boolean algebra** is a means for expressing the relationship between a logic circuit's inputs and outputs. The inputs are considered logic variables whose logic levels at any time determine the output levels. In all our work to follow, we shall use letter symbols to represent logic variables. For example, the letter $A$ might represent a certain digital circuit input or output, and at any time we must have either $A = 0$ or $A = 1$: if not one, then the other.

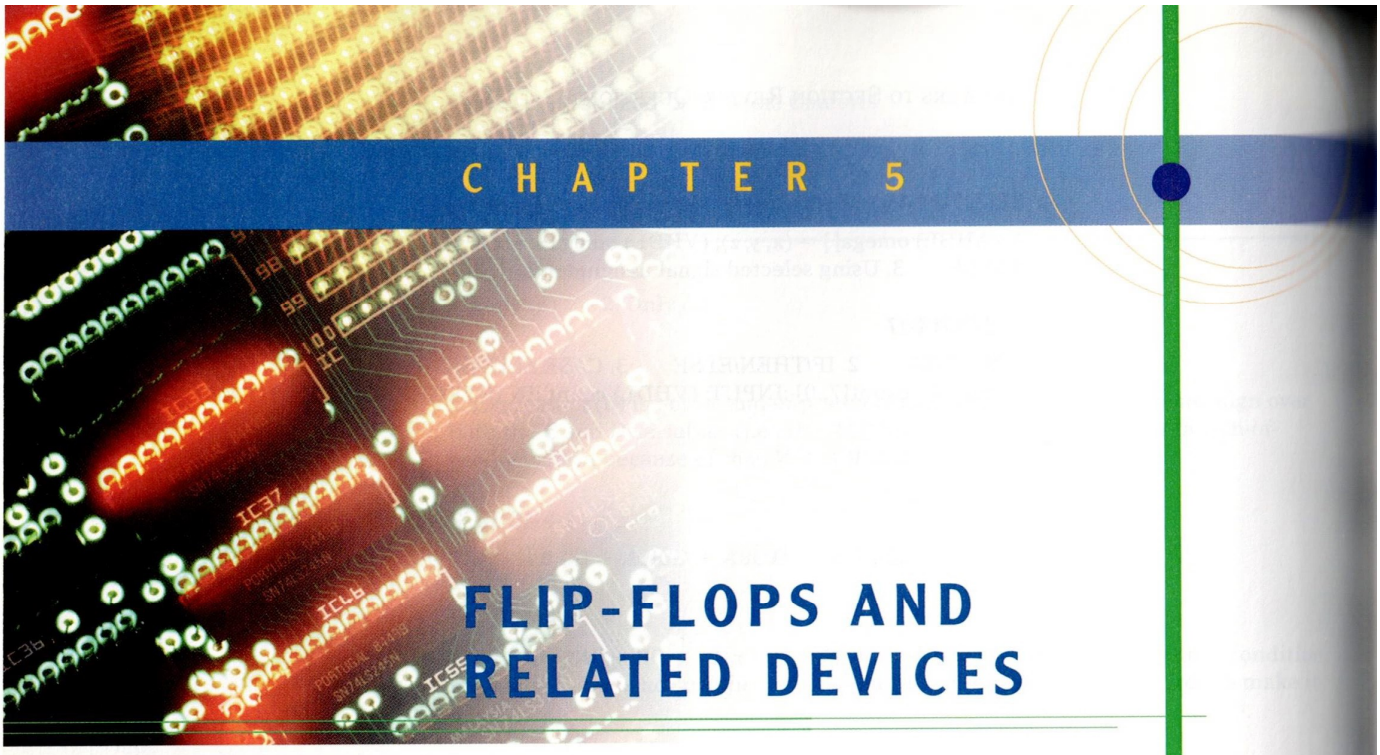
Because only two values are possible, Boolean algebra is relatively easy to work with compared with ordinary algebra. In Boolean algebra, there are no fractions, decimals, negative numbers, square roots, cube roots, logarithms, imaginary numbers, and so on. In fact, in Boolean algebra there are only *three* basic operations: *OR, AND,* and *NOT.*

These basic operations are called *logic operations.* Digital circuits called *logic gates* can be constructed from diodes, transistors, and resistors connected so that the circuit output is the result of a basic logic operation *(OR, AND, NOT)* performed on the inputs. We will be using Boolean algebra first to describe and analyze these basic logic gates, then later to analyze and design combinations of logic gates connected as logic circuits.

## 3-2 TRUTH TABLES

A **truth table** is a means for describing how a logic circuit's output depends on the logic levels present at the circuit's inputs. Figure 3-1(a) illustrates a truth table for one type of two-input logic circuit. The table lists all possible

*Voltages between 0.8 and 2 V are undefined (neither 0 nor 1) and should not occur under normal circumstances.



CHAPTER 5

# FLIP-FLOPS AND RELATED DEVICES

## OUTLINE


5-1 NAND Gate Latch
5-2 NOR Gate Latch
5-3 Troubleshooting Case Study
5-4 Digital Pulses
5-5 Clock Signals and Clocked Flip-Flops
5-6 Clocked S-R Flip-Flop
5-7 Clocked J-K Flip-Flop
5-8 Clocked D Flip-Flop
5-9 *D* Latch (Transparent Latch)
5-10 Asynchronous Inputs
5-11 IEEE/ANSI Symbols
5-12 Flip-Flop Timing Considerations
5-13 Potential Timing Problem in FF Circuits
5-14 Flip-Flop Applications
5-15 Flip-Flop Synchronization
5-16 Detecting an Input Sequence
5-17 Data Storage and Transfer
5-18 Serial Data Transfer: Shift Registers
5-19 Frequency Division and Counting
5-20 Microcomputer Application
5-21 Schmitt-Trigger Devices
5-22 One-Shot (Monostable Multivibrator)
5-23 Clock Generator Circuits
5-24 Troubleshooting Flip-Flop Circuits
5-25 Sequential Circuits Using HDL
5-26 Edge-Triggered Devices
5-27 HDL Circuits with Multiple Components

## OBJECTIVES

*Upon completion of this chapter, you will be able to:*

- Construct and analyze the operation of a latch flip-flop made from NAND or NOR gates.
- Describe the difference between synchronous and asynchronous systems.
- Understand the operation of edge-triggered flip-flops.
- Analyze and apply the various flip-flop timing parameters specified by the manufacturers.
- Understand the major differences between parallel and serial data transfers.
- Draw the output timing waveforms of several types of flip-flops in response to a set of input signals.
- Recognize the various IEEE/ANSI flip-flop symbols.
- Use state transition diagrams to describe counter operation.
- Use flip-flops in synchronization circuits.
- Connect shift registers as data transfer circuits.
- Employ flip-flops as frequency-division and counting circuits.
- Understand the typical characteristics of Schmitt triggers.
- Apply two different types of one-shots in circuit design.
- Design a free-running oscillator using a 555 timer.
- Recognize and predict the effects of clock skew on synchronous circuits.
- Troubleshoot various types of flip-flop circuits.
- Write HDL code for latches.
- Use logic primitives, components, and libraries in HDL code.
- Build structural level circuits from components.

## INTRODUCTION

The logic circuits considered thus far have been combinational circuits whose output levels at any instant of time are dependent on the levels present at the inputs at that time. Any prior input-level conditions have no effect on the present outputs because combinational logic circuits have no memory. Most digital systems consist of both combinational circuits and memory elements.

Figure 5-1 shows a block diagram of a general digital system that combines combinational logic gates with memory devices. The combinational portion accepts logic signals from external inputs and from the outputs of the memory elements. The combinational circuit operates on these inputs

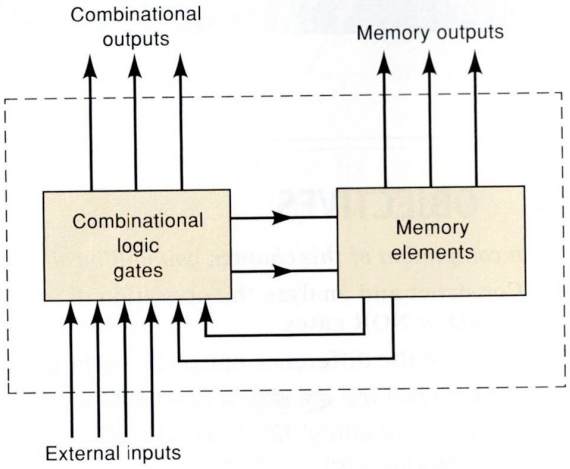



**FIGURE 5-1** General digital system diagram.

to produce various outputs, some of which are used to determine the binary values to be stored in the memory elements. The outputs of some of the memory elements, in turn, go to the inputs of logic gates in the combinational circuits. This process indicates that the external outputs of a digital system are functions of both its external inputs and the information stored in its memory elements.

The most important memory element is the **flip-flop**, which is made up of an assembly of logic gates. Even though a logic gate, by itself, has no storage capability, several can be connected together in ways that permit information to be stored. Several different gate arrangements are used to produce these flip-flops (abbreviated FF).

Figure 5-2(a) is the general type of symbol used for a flip-flop. It shows two outputs, labeled $Q$ and $\overline{Q}$, that are the inverse of each other. $Q/\overline{Q}$ are the most common designations used for a FF's outputs. From time to time, we will use other designations such as $X/\overline{X}$ and $A/\overline{A}$ for convenience in identifying different FFs in a logic circuit.

The $Q$ output is called the *normal* FF output, and $\overline{Q}$ is the *inverted* FF output. Whenever we refer to the state of a FF, we are referring to the state of its normal ($Q$) output; it is understood that its inverted output ($\overline{Q}$) is in the opposite state. For example, if we say that a FF is in the HIGH (1) state, we mean that $Q = 1$; if we say that a FF is in the LOW (0) state, we mean that $Q = 0$. Of course, the $\overline{Q}$ state will always be the inverse of $Q$.

The two possible operating states for a FF are summarized in Figure 5-2(b). Note that the HIGH or 1 state ($Q = 1/\overline{Q} = 0$) is also referred to as the **SET** state. Whenever the inputs to a FF cause it to go to the $Q = 1$ state, we call this *setting* the FF; the FF has been set. In a similar way, the LOW or

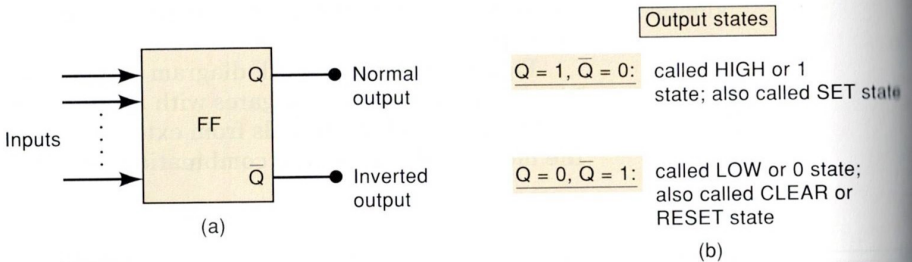



**FIGURE 5-2** General flip-flop symbol and definition of its two possible output states.

0 state ($Q = 0/\overline{Q} = 1$) is also referred to as the **CLEAR** or **RESET** state. Whenever the inputs to a FF cause it to go to the $Q = 0$ state, we call this *clearing* or *resetting* the FF; the FF has been cleared (reset). As we shall see, many FFs will have a **SET** input and/or a **CLEAR (RESET)** input that is used to drive the FF into a specific output state.

As the symbol in Figure 5-2(a) implies, a FF can have one or more inputs. These inputs are used to cause the FF to switch back and forth ("flip-flop") between its possible output states. We will find out that most FF inputs need only to be momentarily activated (pulsed) in order to cause a change in the FF output state, and the output will remain in that new state even after the input pulse is over. This is the FF's *memory* characteristic.

The flip-flop is known by other names, including *latch* and *bistable multivibrator.* The term *latch* is used for certain types of flip-flops that we will describe. The term *bistable multivibrator* is the more technical name for a flip-flop, but it is too much of a mouthful to be used regularly.

## 5-1 NAND GATE LATCH

The most basic FF circuit can be constructed from either two NAND gates or two NOR gates. The NAND gate version, called a **NAND gate latch** or simply a **latch**, is shown in Figure 5-3(a). The two NAND gates are cross-coupled so that the output of NAND-1 is connected to one of the inputs of NAND-2, and vice versa. The gate outputs, labeled $Q$ and $\overline{Q}$, respectively, are the latch outputs. Under normal conditions, these outputs will always be the inverse of each other. There are two latch inputs: the SET input is the input that *sets* $Q$ to the 1 state; the RESET input is the input that *resets* $Q$ to the 0 state.

The SET and RESET inputs are both normally resting in the HIGH state, and one of them will be pulsed LOW whenever we want to change the latch outputs. We begin our analysis by showing that there are two equally likely output states when SET = RESET = 1. One possibility is shown in Figure 5-3(a), where we have $Q = 0$ and $\overline{Q} = 1$. With $Q = 0$, the inputs to NAND-2 are 0 and 1, which produce $\overline{Q} = 1$. The 1 from $\overline{Q}$ causes NAND-1 to have a 1 at both inputs to produce a 0 output at $Q$. In effect, what we have is the LOW at the NAND-1 output producing a HIGH at the NAND-2 output, which, in turn, keeps the NAND-1 output LOW.

The second possibility is shown in Figure 5-3(b), where $Q = 1$ and $\overline{Q} = 0$. The HIGH from NAND-1 produces a LOW at the NAND-2 output, which, in turn, keeps the NAND-1 output HIGH. Thus, there are two possible output states when SET = RESET = 1; as we shall soon see, the one that actually exists will depend on what has occurred previously at the inputs.

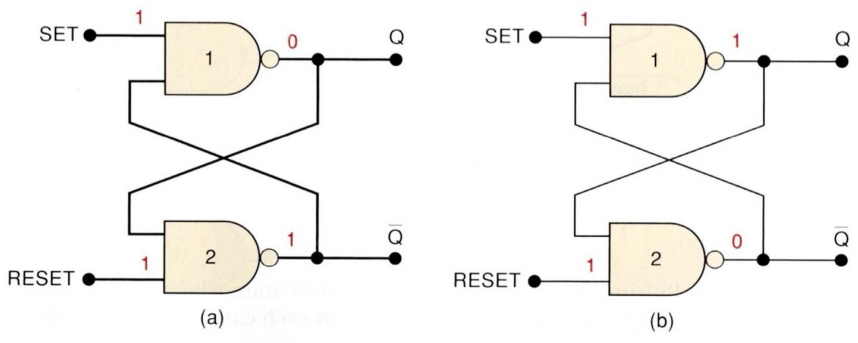



FIGURE 5-3 A NAND latch has two possible resting states when SET = RESET = 1.

## Setting the Latch (FF)

Now let's investigate what happens when the SET input is momentarily pulsed LOW while RESET is kept HIGH. Figure 5-4(a) shows what happens when $Q = 0$ prior to the occurrence of the pulse. As SET is pulsed LOW at time $t_0$, $Q$ will go HIGH, and this HIGH will force $\overline{Q}$ to go LOW so that NAND-1 now has two LOW inputs. Thus, when SET returns to the 1 state at $t_1$, the NAND-1 output *remains* HIGH, which, in turn, keeps the NAND-2 output LOW.

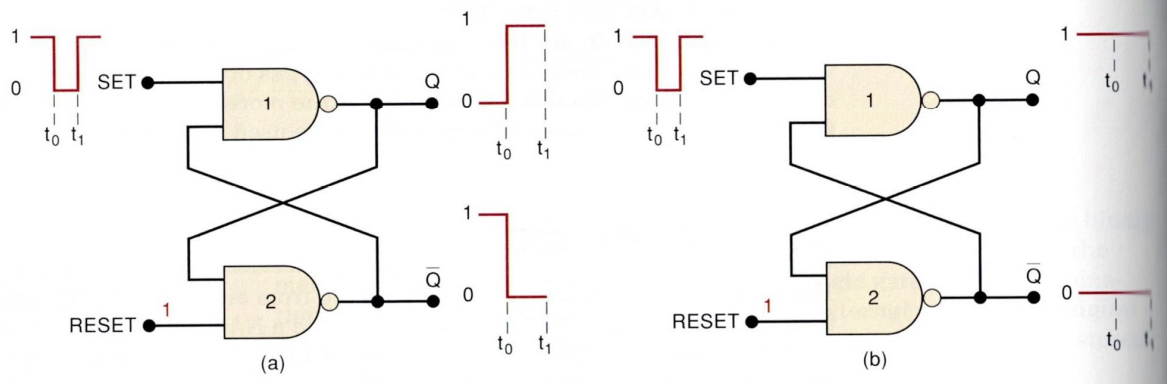



**FIGURE 5-4** Pulsing the SET input to the 0 state when (a) $Q = 0$ prior to SET pulse; (b) $Q = 1$ prior to SET pulse. Note that, in both cases, $Q$ ends up HIGH.

Figure 5-4(b) shows what happens when $Q = 1$ and $\overline{Q} = 0$ prior to the application of the SET pulse. Since $\overline{Q} = 0$ is already keeping the NAND-1 output HIGH, the LOW pulse at SET will not change anything. Thus, when SET returns HIGH, the latch outputs are still in the $Q = 1, \overline{Q} = 0$ state.

We can summarize Figure 5-4 by stating that a LOW pulse on the SET input will always cause the latch to end up in the $Q = 1$ state. This operation is called *setting* the latch or FF.

## Resetting the Latch (FF)

Now let's consider what occurs when the RESET input is pulsed LOW while SET is kept HIGH. Figure 5-5(a) shows what happens when $Q = 0$ and $\overline{Q} = 1$

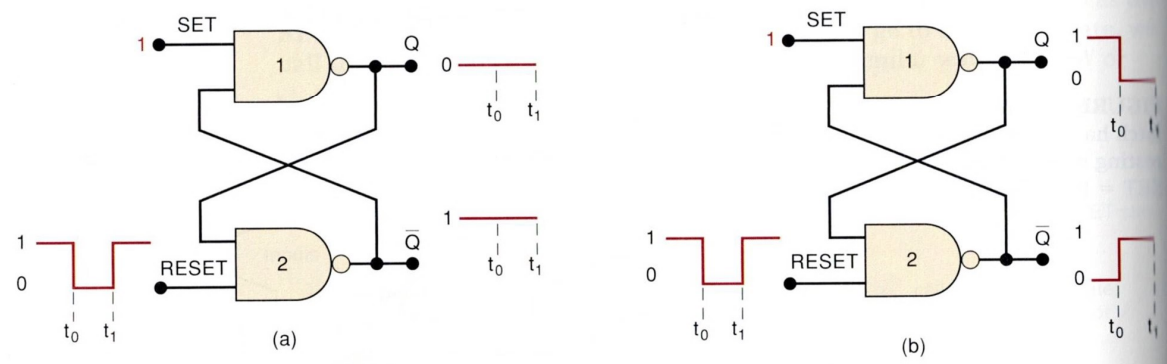



**FIGURE 5-5** Pulsing the RESET input to the LOW state when (a) $Q = 0$ prior to RESET pulse; (b) $Q = 1$ prior to RESET pulse. In each case, $Q$ ends up LOW.

prior to the application of the pulse. Since $Q = 0$ is already keeping the NAND-2 output HIGH, the LOW pulse at RESET will not have any effect. When RESET returns HIGH, the latch outputs are still $Q = 0$ and $\overline{Q} = 1$.

Figure 5-5(b) shows the situation where $Q = 1$ prior to the occurrence of the RESET pulse. As RESET is pulsed LOW at $t_0$, $\overline{Q}$ will go HIGH, and this HIGH forces $Q$ to go LOW so that NAND-2 now has two LOW inputs. Thus, when RESET returns HIGH at $t_1$, the NAND-2 output *remains* HIGH, which, in turn, keeps the NAND-1 output LOW.

Figure 5-5 can be summarized by stating that a LOW pulse on the RESET input will always cause the latch to end up in the $Q = 0$ state. This operation is called *clearing* or *resetting* the latch.

## Simultaneous Setting and Resetting

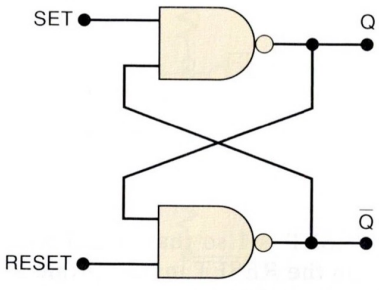
The last case to consider is the case where the SET and RESET inputs are simultaneously pulsed LOW. This will produce HIGH levels at both NAND outputs so that $Q = \overline{Q} = 1$. Clearly, this is an undesired condition because the two outputs are supposed to be inverses of each other. Furthermore, when the SET and RESET inputs return HIGH, the resulting output state will depend on which input returns HIGH first. Simultaneous transitions back to the 1 state will produce unpredictable results. For these reasons the SET = RESET = 0 condition is normally not used for the NAND latch.

## Summary of NAND Latch

The operation described above can be conveniently placed in a function table (Figure 5-6) and is summarized as follows:

1. SET = RESET = 1. This condition is the normal resting state, and it has no effect on the output state. The $Q$ and $\overline{Q}$ outputs will remain in whatever state they were in prior to this input condition.
2. SET = 0, RESET = 1. This will always cause the output to go to the $Q = 1$ state, where it will remain even after SET returns HIGH. This is called *setting* the latch.
3. SET = 1, RESET = 0. This will always produce the $Q = 0$ state, where the output will remain even after RESET returns HIGH. This is called *clearing* or *resetting* the latch.
4. SET = RESET = 0. This condition tries to set and clear the latch at the same time, and it produces $Q = \overline{Q} = 1$. If the inputs are returned to 1 simultaneously, the resulting state is unpredictable. This input condition should not be used.






(a)

| Set | Reset | Output |
|---|---|---|
| 1 | 1 | No change |
| 0 | 1 | Q = 1 |
| 1 | 0 | Q = 0 |
| 0 | 0 | Invalid* |

*Produces $Q = \overline{Q} = 1$.

(b)

FIGURE 5-6 (a) NAND latch; (b) function table.

## Alternate Representations

From the description of the NAND latch operation, it should be clear that the SET and RESET inputs are active-LOW. The SET input will set $Q = 1$ when SET goes LOW; the RESET input will clear $Q = 0$ when RESET goes LOW. For this reason, the NAND latch is often drawn using the alternate representation for each NAND gate, as shown in Figure 5-7(a). The bubbles on the inputs, as well as the labeling of the signals as $\overline{SET}$ and $\overline{RESET}$, indicate the active-LOW status of these inputs. (You may want to review Sections 3-13 and 3-14 on this topic.)

Figure 5-7(b) shows a simplified block representation that we will sometimes use. The $S$ and $R$ labels represent the SET and RESET inputs, and the bubbles indicate the active-LOW nature of these inputs. Whenever we use this block symbol, it represents a NAND latch.

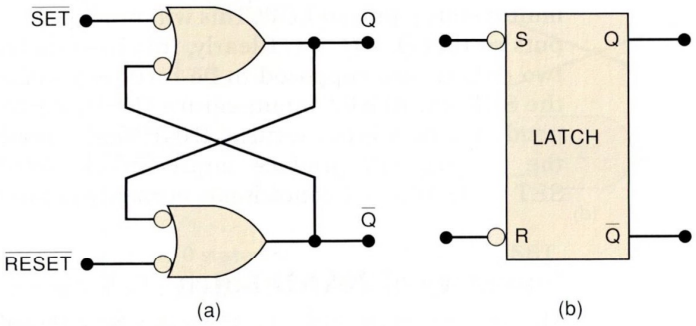



**FIGURE 5-7** (a) NAND latch equivalent representation; (b) simplified block symbol.

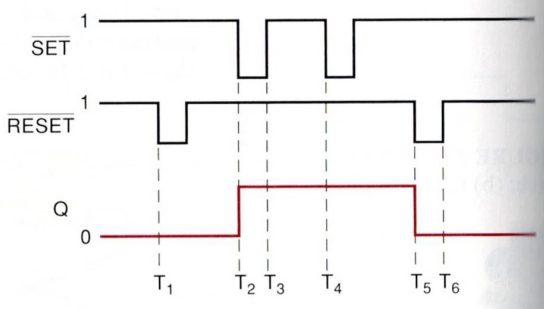
## Terminology

The action of *resetting* a FF or a latch is also called *clearing*, and both terms are used interchangeably in the digital field. In fact, a RESET input can also be called a CLEAR input, and a SET-RESET latch can be called a SET-CLEAR latch.

**EXAMPLE 5-1**

The waveforms of Figure 5-8 are applied to the inputs of the latch of Figure 5-7. Assume that initially $Q = 0$, and determine the $Q$ waveform.

**FIGURE 5-8** Example 5-1.




### Solution

Initially, $\overline{SET} = \overline{RESET} = 1$ so that $Q$ will remain in the 0 state. The LOW pulse that occurs on the $\overline{RESET}$ input at time $T_1$ will have no effect because $Q$ is already in the cleared (0) state.

The only way that $Q$ can go to the 1 state is by a LOW pulse on the $\overline{\text{SET}}$ input. This occurs at time $T_2$ when $\overline{\text{SET}}$ first goes LOW. When $\overline{\text{SET}}$ returns HIGH at $T_3$, $Q$ will remain in its new HIGH state.

At time $T_4$ when $\overline{\text{SET}}$ goes LOW again, there will be no effect on $Q$ because $Q$ is already set to the 1 state.

The only way to bring $Q$ back to the 0 state is by a LOW pulse on the $\overline{\text{RESET}}$ input. This occurs at time $T_5$. When $\overline{\text{RESET}}$ returns to 1 at time $T_6$, $Q$ remains in the LOW state.

Example 5-1 shows that the latch output "remembers" the last input that was activated and will not change states until the opposite input is activated.

**EXAMPLE 5-2**

It is almost impossible to obtain a "clean" voltage transition from a mechanical switch because of the phenomenon of **contact bounce**. This is illustrated in Figure 5-9(a), where the action of moving the switch from contact position 1 to 2 produces several output voltage transitions as the switch bounces (makes and breaks contact with contact 2 several times) before coming to rest on contact 2.

The multiple transitions on the output signal generally last no longer than a few milliseconds, but they would be unacceptable in many applications. A NAND latch can be used to prevent the presence of contact bounce from affecting the output. Describe the operation of the "switch debouncing" circuit in Figure 5-9(b).

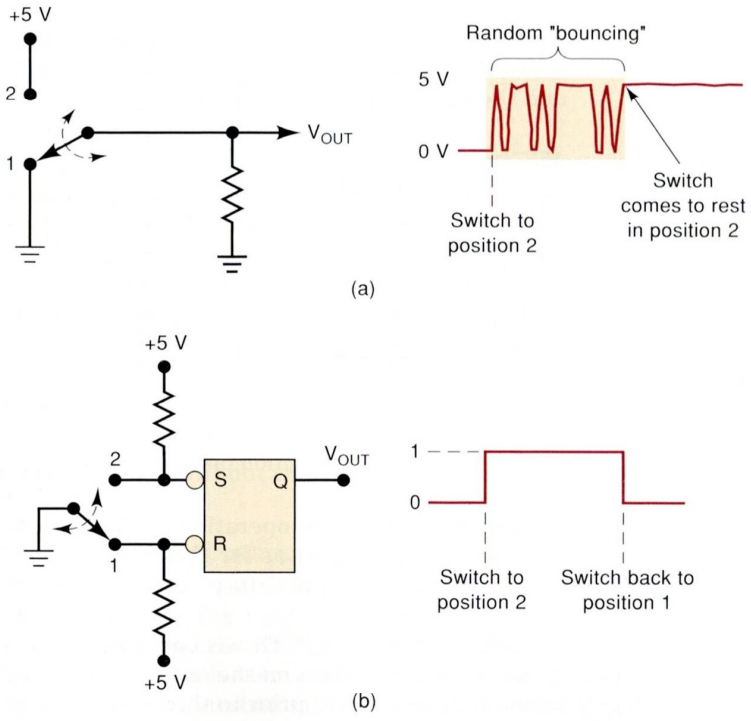



**FIGURE 5-9**
(a) Mechanical contact bounce will produce multiple transitions; (b) NAND latch used to debounce a mechanical switch.

### Solution

Assume that the switch is resting in position 1 so that the $\overline{\text{RESET}}$ input is LOW and $Q = 0$. When the switch is moved to position 2, $\overline{\text{RESET}}$ will go HIGH, and a LOW will appear on the $\overline{\text{SET}}$ input as the switch first makes contact. This will set $Q = 1$ within a matter of a few nanoseconds (the response time of the NAND gate). Now if the switch bounces off contact 2, $\overline{\text{SET}}$ and $\overline{\text{RESET}}$ will both be HIGH, and $Q$ will not be affected; it will stay HIGH. Thus, nothing will happen at $Q$ as the switch bounces on and off contact 2 before finally coming to rest in position 2.

Likewise, when the switch is moved from position 2 back to position 1, it will place a LOW on the $\overline{\text{RESET}}$ input as it first makes contact. This clears $Q$ to the LOW state, where it will remain even if the switch bounces on and off contact 1 several times before coming to rest.

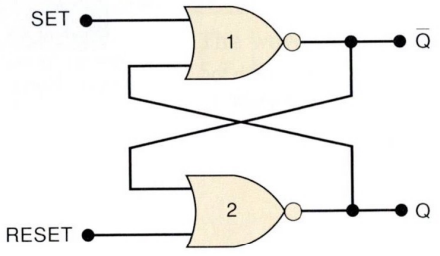
Thus, the output at $Q$ will consist of a single transition each time the switch is moved from one position to the other.

**REVIEW QUESTIONS**

1. What is the normal resting state of the $\overline{\text{SET}}$ and $\overline{\text{RESET}}$ inputs? What is the active state of each input?
2. What will be the states of $Q$ and $\overline{Q}$ after a FF has been reset (cleared)?
3. *True or false:* The $\overline{\text{SET}}$ input can never be used to make $Q = 0$.
4. When power is first applied to any FF circuit, it is impossible to predict the initial states of $Q$ and $\overline{Q}$. What can be done to ensure that a NAND latch always starts off in the $Q = 1$ state?

## 5-2 NOR GATE LATCH

Two cross-coupled NOR gates can be used as a **NOR gate latch**. The arrangement, shown in Figure 5-10(a), is similar to the NAND latch except that the $Q$ and $\overline{Q}$ outputs have reversed positions.

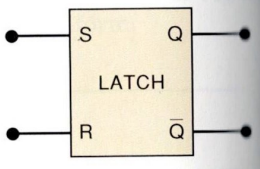



(a)

| Set | Reset | Output |
|---|---|---|
| 0 | 0 | No change |
| 1 | 0 | Q = 1 |
| 0 | 1 | Q = 0 |
| 1 | 1 | Invalid* |

*Produces Q = $\overline{Q}$ = 0.

(b)




(c)

**FIGURE 5-10** (a) NOR gate latch; (b) function table; (c) simplified block symbol.

The analysis of the operation of the NOR latch can be performed in exactly the same manner as for the NAND latch. The results are given in the function table in Figure 5-10(b) and are summarized as follows:

1. SET = RESET = 0. This is the normal resting state for the NOR latch, and it has no effect on the output state. $Q$ and $\overline{Q}$ will remain in whatever state they were in prior to the occurrence of this input condition.

2. SET = 1, RESET = 0. This will always set $Q = 1$, where it will remain even after SET returns to 0.
3. SET = 0, RESET = 1. This will always clear $Q = 0$, where it will remain even after RESET returns to 0.
4. SET = 1, RESET = 1. This condition tries to set and reset the latch at the same time, and it produces $Q = \overline{Q} = 0$. If the inputs are returned to 0 simultaneously, the resulting output state is unpredictable. This input condition should not be used.

The NOR gate latch operates exactly like the NAND latch except that the SET and RESET inputs are active-HIGH rather than active-LOW, and the normal resting state is SET = RESET = 0. $Q$ will be set HIGH by a HIGH pulse on the SET input, and it will be cleared LOW by a HIGH pulse on the RESET input. The simplified block symbol for the NOR latch in Figure 5-10(c) is shown with no bubbles on the $S$ and $R$ inputs; this indicates that these inputs are active-HIGH.

## EXAMPLE 5-3

Assume that $Q = 0$ initially, and determine the $Q$ waveform for the NOR latch inputs of Figure 5-11.

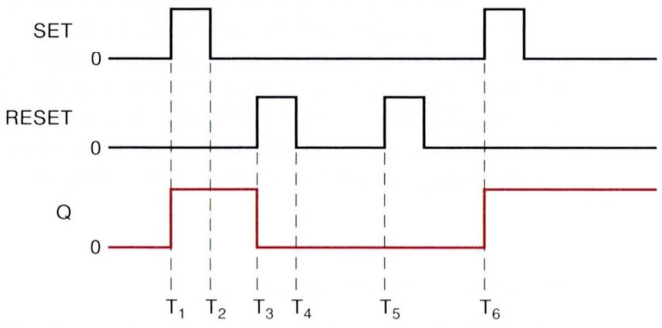



FIGURE 5-11 Example 5-3.

### Solution

Initially, SET = RESET = 0, which has no effect on $Q$, and $Q$ stays LOW. When SET goes HIGH at time $T_1$, $Q$ will be set to 1 and will remain there even after SET returns to 0 at $T_2$.

At $T_3$ the RESET input goes HIGH and clears $Q$ to the 0 state, where it remains even after RESET returns LOW at $T_4$.

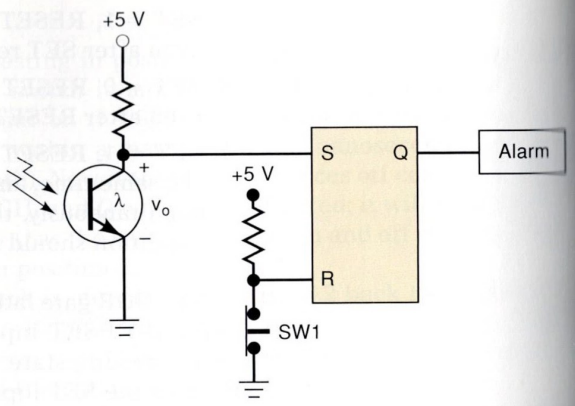
The RESET pulse at $T_5$ has no effect on $Q$ because $Q$ is already LOW. The SET pulse at $T_6$ again sets $Q$ back to 1, where it will stay.

Example 5-3 shows that the latch "remembers" the last input that was activated, and it will not change states until the opposite input is activated.

## EXAMPLE 5-4

Figure 5-12 shows a simple circuit that can be used to detect the interruption of a light beam. The light is focused on a phototransistor that is connected in the common-emitter configuration to operate as a switch. Assume that the latch has previously been cleared to the 0 state by momentarily opening switch SW1, and describe what happens if the light beam is momentarily interrupted.

**FIGURE 5-12**
Example 5-4.




### Solution

With light on the phototransistor, we can assume that it is fully conducting so that the resistance between the collector and the emitter is very small. Thus, $v_0$ will be close to 0 V. This places a LOW on the SET input of the latch so that SET = RESET = 0.

When the light beam is interrupted, the phototransistor turns off, and its collector-emitter resistance becomes very high (i.e., essentially an open circuit). This causes $v_0$ to rise to approximately 5 V; this activates the SET input, which sets $Q$ HIGH and turns on the alarm.

$Q$ will remain HIGH and the alarm will remain on even if $v_0$ returns to 0 V (i.e., the light beam was interrupted only momentarily) because SET and RESET will both be LOW, which will produce no change in $Q$.

In this application, the latch's memory characteristic is used to convert a momentary occurrence (beam interruption) into a constant output.

## Flip-Flop State on Power-Up

When power is applied to a circuit, it is not possible to predict the starting state of a flip-flop's output if its SET and RESET inputs are in their inactive state (e.g., $S = R = 1$ for a NAND latch, $S = R = 0$ for a NOR latch). There is just as much chance that the starting state will be $Q = 0$ as $Q = 1$. It will depend on factors such as internal propagation delays, parasitic capacitance, and external loading. If a latch or FF must start off in a particular state to ensure the proper operation of a circuit, then it must be placed in that state by momentarily activating the SET or RESET input at the start of the circuit's operation. This is often achieved by application of a pulse to the appropriate input.

**REVIEW QUESTIONS**

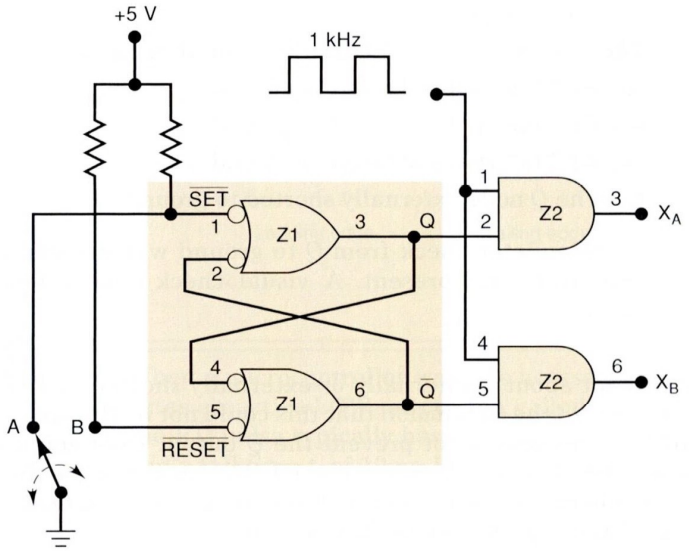
1. What is the normal resting state of the NOR latch inputs? What is the active state?
2. When a latch is set, what are the states of $Q$ and $\overline{Q}$?
3. What is the only way to cause the $Q$ output of a NOR latch to change from 1 to 0?
4. If the NOR latch in Figure 5-12 were replaced by a NAND latch, why wouldn't the circuit work properly?

## 5-3 TROUBLESHOOTING CASE STUDY

The following two examples present an illustration of the kinds of reasoning used in troubleshooting a circuit containing a latch.

**EXAMPLE 5-5**

Analyze and describe the operation of the circuit in Figure 5-13.




| Switch position | $X_A$ | $X_B$ |
|---|---|---|
| A | Pulses | LOW |
| B | LOW | Pulses |

**FIGURE 5-13** Examples 5-5 and 5-6.

### Solution

The switch is used to set or clear the NAND latch to produce clean, bounce-free signals at $Q$ and $\overline{Q}$. These latch outputs control the passage of the 1-kHz pulse signal through to the AND outputs $X_A$ and $X_B$.

When the switch moves to position $A$, the latch is set to $Q = 1$. This enables the 1-kHz pulses to pass through to $X_A$, while the LOW at $\overline{Q}$ keeps $X_B = 0$. When the switch moves to position $B$, the latch is cleared to $Q = 0$, which keeps $X_A = 0$, while the HIGH at $\overline{Q}$ enables the pulses to pass through to $X_B$.

**EXAMPLE 5-6**

A technician tests the circuit of Figure 5-13 and records the observations shown in Table 5-1. He notices that when the switch is in position $B$, the circuit functions correctly, but in position $A$ the latch does not set to the $Q = 1$ state. What are the possible faults that could produce this malfunction?

### Solution

There are several possibilities:

1. An internal open connection at Z1-1, which would prevent $Q$ from responding to the $\overline{\text{SET}}$ input.