

**Fundamentals of**  
**Digital Logic and**  
**Microcomputer Design**

Fifth Edition

This Page Intentionally Left Blank

**Fundamentals of  
Digital Logic and  
Microcomputer Design**

This Page Intentionally Left Blank

# **Fundamentals of Digital Logic and Microcomputer Design**

Fifth Edition

**M. RAFIQUZZAMAN, Ph.D.**

Professor

California State Polytechnic University

Pomona, California

and

President

Rafi Systems, Inc.



**WILEY-INTERSCIENCE**

A JOHN WILEY & SONS, INC., PUBLICATION

Copyright © 2005 by John Wiley & Sons, Inc. All rights reserved.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.  
Published simultaneously in Canada.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600, or on the web at [www.copyright.com](http://www.copyright.com). Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008.

**Limit of Liability/Disclaimer of Warranty:** While the publisher and author have used their best efforts in preparing this book, they make no representation or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993 or fax 317-572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print, however, may not be available in electronic format.

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

Rafiquzzaman, Mohamed.

Fundamentals of digital logic and microcomputer design / M. Rafiquzzaman.—5th ed.  
p. cm.

Includes bibliographical references and index.

ISBN 0-471-72784-9 (cloth)

1. Logic circuits. 2. Microcomputers—Design and construction. 3. Electronic digital computers—Circuits. I. Title.

TK7888.4.R34 2005

621.39'5—dc22

2004065974

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

*In memory of my beloved parents, who gave me  
tremendous support, encouragement, and  
guidance in achieving my career goals.  
I will always miss them.*

*To my wife, Kusum, and brother, Elan*

This Page Intentionally Left Blank



# Contents

<b>PREFACE</b>	<b>xv</b>
<b>1. INTRODUCTION TO DIGITAL SYSTEMS</b>	<b>1</b>
1.1 Explanation of Terms	2
1.2 Design Levels	4
1.3 Combinational vs. Sequential Systems	4
1.4 Digital Integrated Circuits	5
1.4.1 Diodes	5
1.4.2 Transistors	6
1.4.3 MOS Transistors	13
1.5 Integrated Circuits (ICs)	15
1.6 Evolution of Computers	17
1.7 A Typical Microcomputer-Based Application	19
1.8 Trends and Perspectives in Digital Technology	19
<b>2. NUMBER SYSTEMS AND CODES</b>	<b>23</b>
2.1 Number Systems	23
2.1.1 General Number Representation	23
2.1.2 Converting Numbers from One Base to Another	26
2.2 Unsigned and Signed Binary Numbers	28
2.3 Codes	32
2.3.1 Binary-Coded-Decimal Code (8421 Code)	32
2.3.2 Alphanumeric Codes	32
2.3.3 Excess-3 Code	34
2.3.4 Gray Code	35
2.3.5 Unicode	36
2.4 Fixed-Point and Floating-Point Representations	37
2.5 Arithmetic Operations	37
2.5.1 Binary Arithmetic	38
2.5.2 BCD Arithmetic	47
2.5.3 Multiword Binary Addition and Subtraction	48
2.6 Error Correction and Detection	49
Questions and Problems	50

<b>3. BOOLEAN ALGEBRA AND DIGITAL LOGIC GATES</b>	<b>53</b>
3.1 Basic Logic Operations	53
3.1.1 NOT Operation	53
3.1.2 OR Operation	54
3.1.3 AND Operation	55
3.2 Other Logic Operations	58
3.2.1 NOR Operation	58
3.2.2 NAND Operation	58
3.2.3 Exclusive-OR Operation (XOR)	60
3.2.4 Exclusive-NOR Operation (XNOR)	61
3.3 IEEE Symbols for Logic Gates	62
3.4 Positive and Negative Logic	63
3.5 Boolean Algebra	64
3.5.1 Boolean Identities	65
3.5.2 Simplification Using Boolean Identities	67
3.5.3 Consensus Theorem	68
3.5.4 Complement of a Boolean Function	70
3.6 Standard Representations	71
3.7 Karnaugh Maps	75
3.7.1 Two-Variable K-Map	76
3.7.2 Three-Variable K-Map	76
3.7.3 Four-Variable K-Map	79
3.7.4 Prime Implicants	81
3.7.5 Expressing a Function in Product-of-Sums Form Using a K-Map	83
3.7.6 Don't Care Conditions	83
3.7.7 Five-Variable K-Map	85
3.8 Quine–McCluskey Method	86
3.9 Implementation of Digital Circuits with NAND, NOR, and Exclusive-OR/Exclusive-NOR Gates	88
3.9.1 NAND Gate Implementation	88
3.9.2 NOR Gate Implementation	91
3.9.3 XOR / XNOR Implementations	91
Questions and Problems	95
<b>4. COMBINATIONAL LOGIC DESIGN</b>	<b>99</b>
4.1 Basic Concepts	99
4.2 Analysis of a Combinational Logic Circuit	100
4.3 Design of a Combinational Circuit	101
4.4 Multiple-Output Combinational Circuits	102
4.5 Typical Combinational Circuits	106
4.5.1 Binary / BCD Adders and Binary Subtractors	106
4.5.2 Comparators	110
4.5.3 Decoders	112
4.5.4 Encoders	115
4.5.5 Multiplexers	116
4.5.6 Demultiplexers	118
4.6 IEEE Standard Symbols	118
4.7 Read-Only Memories (ROMs)	121

4.8	Programmable Logic Devices (PLDs)	123
4.9	Commercially Available Field Programmable Devices (FPDs)	126
4.10	Hardware Description Language (HDL)	127
	Questions and Problems	129
<b>5.</b>	<b>SEQUENTIAL LOGIC DESIGN</b>	<b>135</b>
5.1	Basic Concepts	135
5.2	Flip-Flops	136
5.2.1	SR Latch	136
5.2.2	RS Flip-Flop	138
5.2.3	D Flip-Flop	139
5.2.4	JK Flip-Flop	139
5.2.5	T Flip-Flop	140
5.3	Master-Slave Flip-Flop	140
5.4	Preset and Clear Inputs	141
5.5	Summary of Flip-Flops	143
5.6	Analysis of Synchronous Sequential Circuits	145
5.7	Types of Synchronous Sequential Circuits	148
5.8	Minimization of States	148
5.9	Design of Synchronous Sequential Circuits	150
5.10	Design of Counters	156
5.11	Examples of Synchronous Sequential Circuits	161
5.11.1	Registers	162
5.11.2	Modulo- $n$ Counters	164
5.11.3	Random-Access Memory (RAM)	166
5.12	Algorithmic State Machines (ASM) Chart	168
5.13	Asynchronous Sequential Circuits	176
	Questions and Problems	178
<b>6.</b>	<b>MICROCOMPUTER ARCHITECTURE, PROGRAMMING, AND SYSTEM DESIGN CONCEPTS</b>	<b>185</b>
6.1	Basic Blocks of a Microcomputer	185
6.2	Typical Microcomputer Architecture	186
6.2.1	The Microcomputer Bus	186
6.2.2	Clock Signals	187
6.3	The Single-Chip Microprocessor	188
6.3.1	Register Section	188
6.3.2	Control Unit	198
6.3.3	Arithmetic and Logic Unit (ALU)	199
6.3.4	Functional Representations of a Simple and a Typical Microprocessor	199
6.3.5	Microprogramming the Control Unit (A Simplified Explanation)	201
6.4	The Memory	204
6.4.1	Random-Access Memory (RAM)	205
6.4.2	Read-Only Memory (ROM)	206
6.4.3	READ and WRITE Operations	207
6.4.4	Memory Organization	209
6.5	Input/Output	209

6.6	Microcomputer Programming Concepts	210
6.6.1	Microcomputer Programming Languages	210
6.6.2	Machine Language	211
6.6.3	Assembly Language	212
6.6.4	High-Level Languages	222
6.7	Monitors	227
6.8	Flowcharts	228
6.9	Basic Features of Microcomputer Development Systems	228
6.10	System Development Flowchart	232
	Questions and Problems	233
<b>7.</b>	<b>DESIGN OF COMPUTER INSTRUCTION SET AND THE CPU</b>	<b>237</b>
7.1	Design of the Computer Instructions	237
7.2	Reduced Instruction Set Computer (RISC)	239
7.3	Design of the CPU	242
7.3.1	Register Design	242
7.3.2	Adders	244
7.3.3	Addition, Subtraction, Multiplication and Division of Unsigned and Signed Numbers	250
7.3.4	ALU Design	254
7.3.5	Design of the Control Unit	257
7.4	Design of a Microprogrammed CPU	277
	Questions and Problems	286
<b>8.</b>	<b>MEMORY, I/O, AND PARALLEL PROCESSING</b>	<b>299</b>
8.1	Memory Organization	299
8.1.1	Introduction	299
8.1.2	Main Memory Array Design	300
8.1.3	Virtual Memory and Memory Management Concepts	304
8.1.4	Cache Memory Organization	326
8.2	Input/Output	335
8.2.1	Programmed I/O	336
8.2.2	Interrupt I/O	340
8.2.3	Direct Memory Access (DMA)	345
8.3	Summary of I/O	347
8.4	Fundamentals of Parallel Processing	347
8.4.1	General Classifications of Computer Architectures	348
8.4.2	Pipeline Processing	351
	Questions and Problems	359
<b>9.</b>	<b>INTEL 8086</b>	<b>367</b>
9.1	Introduction	367
9.2	8086 Main Memory	369
9.3	8086 Registers	370
9.4	8086 Addressing Modes	373
9.4.1	Register and Immediate Modes	374
9.4.2	Memory Addressing Modes	374
9.4.3	Port Addressing	376

9.4.4	Relative Addressing Mode	376
9.4.5	Implied Addressing Mode	376
9.5	8086 Instruction Set	376
9.5.1	Data Transfer Instructions	377
9.5.2	Arithmetic Instructions	379
9.5.3	Bit Manipulation Instructions	384
9.5.4	String Instructions	386
9.5.5	Unconditional Transfer Instructions	388
9.5.6	Conditional Branch Instructions	391
9.5.7	Iteration Control Instructions	393
9.5.8	Interrupt Instructions	394
9.5.9	Processor Control Instructions	395
9.6	8086 Assembler-Dependent Instructions	395
9.7	Typical 8086 Assembler Pseudo-Instructions or Directives	397
9.7.1	SEGMENT and ENDS Directives	397
9.7.2	ASSUME Directive	397
9.7.3	DUP, LABEL, and Other Directives	398
9.7.4	8086 Stack	399
9.8	8086 Delay Routine	399
9.9	System Design Using the 8086	414
9.9.1	8086 Pins and Signals	414
9.9.2	Basic 8086 System Concepts	421
9.9.3	Interfacing with Memories	425
9.9.4	8086 I/O Ports	428
9.9.5	Important Points To Be Considered for 8086 Interface to Memory and I/O	430
9.10	8086-Based Microcomputer	434
9.11	8086 Interrupts	436
9.11.1	Predefined Interrupts	436
9.11.2	Internal Interrupts	437
9.11.3	External Maskable Interrupts	437
9.11.4	Interrupt Procedures	438
9.11.5	Interrupt Priorities	438
9.11.6	Interrupt Pointer Table	439
9.12	8086 DMA	439
9.13	Interfacing an 8086-Based Microcomputer to a Hexadecimal Keyboard and Seven-Segment Displays	445
9.13.1	Basics of Keyboard and Display Interface to a Microcomputer	445
9.13.2	Hex Keyboard Interface to an 8086-Based Microcomputer	447
	Questions and Problems	451
<b>10.</b>	<b>MOTOROLA MC68000</b>	<b>457</b>
10.1	Introduction	457
10.2	68000 Registers	460
10.3	68000 Memory Addressing	461
10.4	68000 Addressing Modes	461
10.4.1	Register Direct Addressing	463
10.4.2	Address Register Indirect Addressing	463

10.4.3	Absolute Addressing	465
10.4.4	Program Counter Relative Addressing	465
10.4.5	Immediate Data Addressing	465
10.4.6	Implied Addressing	466
10.5	Functional Categories of 68000 Addressing Modes	466
10.6	68000 Instruction Set	467
10.6.1	Data Movement Instructions	469
10.6.2	Arithmetic Instructions	472
10.6.3	Logical Instructions	477
10.6.4	Shift and Rotate Instructions	479
10.6.5	Bit Manipulation Instructions	482
10.6.6	Binary-Coded-Decimal Instructions	482
10.6.7	Program Control Instructions	483
10.6.8	System Control Instructions	486
10.6.9	68000 Stack	487
10.7	68000 Delay Routine	489
10.8	68000 Pins And Signals	498
10.8.1	Synchronous and Asynchronous Control Lines	500
10.8.2	System Control Lines	502
10.8.3	Interrupt Control Lines	503
10.8.4	DMA Control Lines	503
10.8.5	Status Lines	503
10.9	68000 Clock and Reset Signals	503
10.9.1	68000 Clock Signals	503
10.9.2	68000 Reset Circuit	504
10.10	68000 Read and Write Cycle Timing Diagrams	509
10.11	68000 Memory Interface	511
10.12	68000 I/O	514
10.12.1	68000 Programmed I/O	514
10.12.2	68000 Interrupt System	521
10.12.3	68000 DMA	526
10.13	68000 Exception Handling	526
10.14	68000/2732/6116/6821-Based Microcomputer	529
10.15	Multiprocessing with the 68000 Using the TAS Instruction and the AS Signal	532
	Questions and Problems	535
<b>11.</b>	<b>INTEL AND MOTOROLA 32- &amp; 64-BIT MICROPROCESSORS</b>	<b>543</b>
11.1	Typical Features of 32-Bit and 64-Bit Microprocessors	543
11.2	Intel 32-Bit and 64-Bit Microprocessors	545
11.3	Intel 80386	546
11.3.1	Internal 80386 Architecture	547
11.3.2	Processing Modes	547
11.3.3	Basic 80386 Programming Model	548
11.3.4	80386 Addressing Modes	550
11.3.5	80386 Instruction Set	551
11.3.6	80386 Pins and Signals	560
11.3.7	80386 Modes	561

11.3.8	80386 System Design	562
11.3.9	80386 I/O	564
11.4	Intel 80486 Microprocessor	565
11.4.1	Intel 80486/80386 Comparison	565
11.4.2	Special Features of the 80486	565
11.4.3	80486 New Instructions Beyond Those of the 80386	567
11.5	Intel Pentium Microprocessor	568
11.5.1	Pentium Registers	570
11.5.2	Pentium Addressing Modes and Instructions	570
11.5.3	Pentium versus 80486: Basic Differences in Registers, Paging, Stack Operations, and Exceptions	571
11.5.4	Pentium Input/Output	571
11.5.5	Applications with the Pentium	572
11.5.6	Pentium versus Pentium Pro	572
11.5.7	Pentium II / Celeron / Pentium II Xeon™/ Pentium III / Pentium 4	573
11.6	Merced/IA-64	575
11.7	Overview of Motorola 32- and 64-Bit Microprocessors	576
11.7.1	Motorola MC68020	576
11.7.2	Motorola MC68030	610
11.7.3	Motorola MC68040 / MC68060	610
11.7.4	PowerPC Microprocessor	611
11.7.5	Motorola's State-of-the-Art Microprocessors	619
	Questions and Problems	620
<b>APPENDIX A—ANSWERS TO SELECTED PROBLEMS</b>		<b>627</b>
<b>APPENDIX B—GLOSSARY</b>		<b>633</b>
<b>APPENDIX C—MOTOROLA 68000 and SUPPORT CHIPS</b>		<b>649</b>
<b>APPENDIX D—68000 EXECUTION TIMES</b>		<b>661</b>
<b>APPENDIX E—INTEL 8086 AND SUPPORT CHIPS</b>		<b>671</b>
<b>APPENDIX F—8086 INSTRUCTION SET REFERENCE DATA</b>		<b>677</b>
<b>APPENDIX G—68000 INSTRUCTION SET</b>		<b>695</b>
<b>APPENDIX H—8086 INSTRUCTION SET</b>		<b>701</b>
<b>APPENDIX I—VERILOG</b>		<b>713</b>
I.1	Introduction to Verilog	713
I.1.1	Structural Modeling	717
I.1.2	Dataflow Modeling	719
I.1.3	Behavioral Modeling	719
I.2	Verilog Descriptions of Typical Combinational Logic Circuits	721
I.3	Verilog Descriptions of Typical Synchronous Sequential Circuits	728

I.4	Status Register Design Using Verilog	741
I.5	CPU Design Using Verilog	743
	Questions and Problems	753
<b>APPENDIX J—VHDL</b>		<b>757</b>
J.1	Introduction to VHDL	757
J.1.1	Structural Modeling	759
J.1.2	Behavioral Modeling	761
J.1.3	Dataflow Modeling	763
J.1.4	Mixed Modeling	765
J.2	VHDL Descriptions of Typical Combinational Logic Circuits	766
J.3	VHDL Descriptions of Typical Synchronous Sequential Circuits	769
J.4	Status Register Design Using VHDL	777
J.5	CPU Design Using VHDL	778
	Questions and Problems	805
<b>BIBLIOGRAPHY</b>		<b>807</b>
<b>CREDITS</b>		<b>811</b>
<b>INDEX</b>		<b>813</b>



# Preface

In this book we cover all basic concepts of computer engineering and science, from digital logic circuits to the design of a complete microcomputer system in a systematic and simplified manner. We have endeavored to present a clear understanding of the principles and basic tools required to design typical digital systems such as microcomputers.

To accomplish this goal, the computer is first defined as consisting of three blocks: central processing unit (CPU), memory, and I/O. We point out that the CPU is analogous to the brain of a human being. Computer memory is similar to human memory. A question asked of a human being is analogous to entering a program into a computer using an input device such as a keyboard, and answering the question by the human is similar in concept to outputting the result required by the program to a computer output device such as a printer. The main difference is that human beings can think independently whereas computers can only answer questions for which they are programmed. Due to advances in semiconductor technology, it is possible to fabricate the CPU on a single chip. The result is the microprocessor. Intel's Pentium and Motorola's Power PC are typical examples of microprocessors. Memory and I/O chips must be connected to the microprocessor chip to implement a microcomputer so that these microprocessors will be able to perform meaningful operations.

We clearly point out that computers understand only 0's and 1's. It is therefore important that students be familiar with binary numbers. Furthermore, we focus on the fact that computers can normally only add. Hence, all other operations such as subtraction are performed via addition. This can be accomplished via two's-complement arithmetic for binary numbers. This topic is therefore also included, along with a clear explanation of signed and unsigned binary numbers.

As far as computer programming is concerned, assembly language programming is covered in this book for typical Intel and Motorola microprocessors. An overview of C, C++, and Java high-level languages is also included. These are the only high-level languages that can perform I/O operations. We point out the advantages and disadvantages of programming typical microprocessors in C and assembly languages.

Three design levels are covered in this book: device level, logic level, and system level. Device-level design, which designs logic gates such as AND, OR, and NOT using transistors, is included from a basic point of view. Logic-level design is the design technique in which logic gates are used to design a digital component such as an adder. Finally, system-level design is covered for typical Intel and Motorola microprocessors. Micro-

computers have been designed by interfacing memory and I/O chips to these microprocessors.

Digital systems at the logic level are classified into two types of circuits, combinational and sequential. Combinational circuits have no memory whereas sequential circuits contain memory. Microprocessors are designed using both combinational and sequential circuits. Therefore, these topics are covered in detail. The fifth edition of this book contains an introduction to synthesizing digital logic circuits using popular hardware description languages such as Verilog and VHDL. These two languages are included in Appendices I and J, independently of each other in such a way that either Verilog or VHDL can be covered in a course without confusion.

The material included in this book is divided into three sections. The first section contains Chapters 1 through 5. In these chapters we describe digital circuits at the gate and flip-flop levels and describe the analysis and design of combinational and sequential circuits. The second section contains Chapters 6 through 8. Here we describe microcomputer organization/architecture, programming, design of computer instruction sets, CPU, memory, and I/O. The third section contains Chapters 9 through 11. These chapters contain typical 16-, 32-, and 64-bit microprocessors manufactured by Intel and Motorola. Future plans of Intel and Motorola are also included. Details of the topics covered in the 11 chapters of this book follow.

- Chapter 1 presents an explanation of basic terminologies, fundamental concepts of digital integrated circuits using transistors; a comparison of LSTTL, HC, and HCT IC characteristics, the evolution of computers, and technological forecasts.
- Chapter 2 provides various number systems and codes suitable for representing information in microprocessors.
- Chapter 3 covers Boolean algebra along with map simplification of Boolean functions. The basic characteristics of digital logic gates are also presented.
- Chapter 4 presents the analysis and design of combinational circuits. Typical combinational circuits such as adders, decoders, encoders, multiplexers, demultiplexers and, ROMs/PLDs are included.
- Chapter 5 covers various types of flip-flops. Analysis and design of sequential circuits such as counters are provided.
- Chapter 6 presents typical microcomputer architecture, internal microprocessor organization, memory, I/O, and programming concepts.
- Chapter 7 covers the fundamentals of instruction set design. The design of registers and ALU is presented. Furthermore, control unit design using both hardwired control and microprogrammed approaches is included. Nanomemory concepts are covered.
- Chapter 8 explains the basics of memory, I/O, and parallel processing. Topics such as main memory array design, memory management concepts, cache memory organization, and pipelining are included.
- Chapters 9 and 10 contain detailed descriptions of the architectures, addressing modes, instruction sets, I/O, and system design concepts associated with the Intel 8086 and Motorola MC68000.
- Chapter 11 provides a summary of the basic features of Intel and Motorola 32- and 64-bit microprocessors. Overviews of the Intel 80486/Pentium/Pentium Pro/Pentium II/Celeron/Pentium III, Pentium 4, and the Motorola 68030/68040/68060/PowerPC

(32- and 64-bit) microprocessors are included. Finally, future plans by both Intel and Motorola are discussed.

The book can be used in a number of ways. Because the materials presented are basic and do not require an advanced mathematical background, the book can easily be adopted as a text for three quarter or two semester courses. These courses can be taught at the undergraduate level in engineering and computer science. The recommended course sequence can be digital logic design in the first course, with topics that include selected portions from Chapters 1 through 5; followed by a second course on computer architecture/organization (Chapters 6 through 8). The third course may include selected topics from Chapters 9 through 11, covering Intel and/or Motorola microprocessors.

The audience for this book can also be graduate students or practicing microprocessor system designers in the industry. Portions of Chapters 9 through 11 can be used as an introductory graduate text in electrical/computer engineering or computer science. Practitioners of microprocessor system design in the industry will find more simplified explanations, together with examples and comparison considerations, than are found in manufacturers' manuals.

Because of increased costs of college textbooks, this book covers several topics including digital logic, computer architecture, assembly language programming, and microprocessor-based system design in a single book. Adequate details are provided. Coverage of certain topics listed below makes the book very unique:

- i) A clear explanation of signed and unsigned numbers using computation of  $(X^2/255)$  as an example (Section 2.2). The same concepts are illustrated using assembly language programming with Intel 8086 microprocessor (Example 9.2), and Motorola 68000 microprocessor (Example 10.2).
- ii) Clarification of packed vs. unpacked BCD (Section 2.3.2). Also, clear explanation of ASCII vs. EBCDIC using an ASCII keyboard and an EBCDIC printer interfaced to a computer as an example (Section 2.3.2); illustration of the same concepts via Intel 8086 assembly language programming using the XLAT instruction (Section 9.5.1).
- iii) Simplified explanation of Digital Logic Design along with numerous examples (Chapters 2 through 5). A clear explanation of the BCD adder (Section 4.5.1). An introduction to basic features of Verilog (Appendix I) and VHDL (Appendix J) along with descriptions of several examples of Chapters 3 through 5. Verilog and VHDL descriptions and syntheses of an ALU and a typical CPU. Coverage of Verilog and VHDL independent of each other in separate appendices without any confusion.
- iv) CD containing a step by step procedure for installing and using Altera Quartus II software for synthesizing Verilog and VHDL descriptions of several combinational and sequential logic design. Screen shots included in CD providing the waveforms and tabular forms illustrating the simulation results.
- v) Application of C language vs. assembly language along with advantages and disadvantages of each (Section 6.6.4).
- vi) Numerous examples of assembly language programming for both Intel 8086 (Chapter 9) and Motorola 68000 (Chapter 10).
- vii) A CD containing a step by step procedure for installing and using MASM 6.11

(8086) and 68asmsim (68000). Screen shots are provided on CD verifying the correct operation of several assembly language programs (both 8086 and 68000) via simulations using test data. The screen shots are obtained by simulating the assembly language programs using DEBUG (8086) and SIM (68000).

- viii) A concise and simplified explanation of system design concepts including programmed I/O and interrupts with the Intel 8086 (Chapter 9) and Motorola 68000 (Chapter 10). Hardware aspects including design of reset circuitry and a simple microcomputer with these microprocessors from the chip level.
- ix) A simplified comparison of RISC vs. CISC relating to Pentium architecture which is comprised of both RISC and CISC (Section 7.3.5). Unique feature of the PowerPC (Section 11.7.4).

The author wishes to express his sincere appreciation to his students, Rami Yassine, Teren Abear, Vireak Ly, Henry Zhong, Roel Delos Reyes, Vu Tran, Henry Ongkoputro, Rega Setiawan, Xibin Wu, Ryan DeGuzman, Angelo Terracina, Javier Ruiz, Yi Ting Huang, Eric Fang, Cindy Yeh, King Lam, Luis Galdamez, Elias Younes, Benjamin Petreaca, and to all others for making constructive suggestions. The author is indebted to his colleagues Dr. R. Chandra, Dr. M. Davarpanah, Dr. T. Sacco, Dr. S. Monemi, and Dr. H. El Naga of California State Poly University, Pomona for their valuable comments. The author is also grateful to Dr. W. C. Miller of University of Windsor, Canada and to his good friends U.S. Congressman Duke Cunningham (TOPGUN, Vietnam) and U.S. Congressman Jerry Weller for their inspiration during the writing effort. Finally, the author would like to thank CJ Media of California for preparing the final version of the manuscript.

M. RAFIQUZZAMAN

*Pomona, California*

# INTRODUCTION TO DIGITAL SYSTEMS

---

Digital systems are designed to store, process, and communicate information in digital form. They are found in a wide range of applications, including process control, communication systems, digital instruments, and consumer products. The digital computer, more commonly called the “computer,” is an example of a typical digital system.

A computer manipulates information in digital, or more precisely, binary form. A binary number has only two discrete values — zero or one. Each of these discrete values is represented by the OFF and ON status of an electronic switch called a “transistor.” All computers, therefore, only understand binary numbers. Any decimal number (base 10, with ten digits from 0 to 9) can be represented by a binary number (base 2, with digits 0 and 1).

The basic blocks of a computer are the central processing unit (CPU), the memory, and the input/output (I/O). The CPU of the computer is basically the same as the brains of a human being. Computer memory is conceptually similar to human memory. A question asked to a human being is analogous to entering a program into the computer using an input device such as the keyboard, and answering the question by the human is similar in concept to outputting the result required by the program to a computer output device such as the printer. The main difference is that human beings can think independently, whereas computers can only answer questions that they are programmed for. Computer hardware refers to components of a computer such as memory, CPU, transistors, nuts, bolts, and so on. Programs can perform a specific task such as addition if the computer has an electronic circuit capable of adding two numbers. Programmers cannot change these electronic circuits but can perform tasks on them using instructions.

Computer software, on the other hand, consists of a collection of programs. Programs contain instructions and data for performing a specific task. These programs, written using any programming language such as C++, must be translated into binary prior to execution by the computer. This is because the computer only understands binary numbers. Therefore, a translator for converting such a program into binary is necessary. Hence, a translator program called the *compiler* is used for translating programs written in a programming language such as C++ into binary. These programs in binary form are then stored in the computer memory for execution because computers only understand 1's and 0's. Furthermore, computers can only add. This means that all operations such as subtraction, multiplication, and division are performed by addition.

Due to advances in semiconductor technology, it is possible to fabricate the CPU in a single chip. The result is the *microprocessor*. Both Metal Oxide Semiconductor (MOS) and Bipolar technologies were used in the fabrication process. The CPU can

be placed on a single chip when MOS technology is used. However, several chips are required with the bipolar technology. HCMOS (High Speed Complementary MOS) or BICMOS (Combination of Bipolar and HCMOS) technology (to be discussed later in this chapter) is normally used these days to fabricate the microprocessor in a single chip. Along with the microprocessor chip, appropriate memory and I/O chips can be used to design a *microcomputer*. The pins on each one of these chips can be connected to the proper lines on the system bus, which consists of address, data, and control lines. In the past, some manufacturers have designed a complete microcomputer on a single chip with limited capabilities. Single-chip microcomputers were used in a wide range of industrial and home applications.

“Microcontrollers” evolved from single-chip microcomputers. The microcontrollers are typically used for dedicated applications such as automotive systems, home appliances, and home entertainment systems. Typical microcontrollers, therefore, include a microcomputer, timers, and A/D (analog to digital) and D/A (digital to analog) converters — all in a single chip. Examples of typical microcontrollers are Intel 8751 (8-bit) / 8096 (16-bit) and Motorola HC11 (8-bit) / HC16 (16-bit).

In this chapter, we first define some basic terms associated with the computers. We then describe briefly the evolution of the computers and the microprocessors. Finally, a typical practical application, and technological forecasts are included.

## 1.1 Explanation of Terms

Before we go on, it is necessary to understand some basic terms.

- A *bit* is the abbreviation for the term binary digit. A binary digit can have only two values, which are represented by the symbols 0 and 1, whereas a decimal digit can have 10 values, represented by the symbols 0 through 9. The bit values are easily implemented in electronic and magnetic media by two-state devices whose states portray either of the binary digits, 0 or 1. Examples of such two-state devices are a transistor that is conducting or not conducting, a capacitor that is charged or discharged, and a magnetic material that is magnetized North-to-South or South-to-North.
- The *bit size* of a computer refers to the number of bits that can be processed simultaneously by the basic arithmetic circuits of the computer. A number of bits taken as a group in this manner is called a *word*. For example, a 32-bit computer can process a 32-bit word. An 8-bit word is referred to as a byte, and a 4-bit word is known as a nibble.
- An *arithmetic logic unit* (ALU) is a digital circuit which performs arithmetic and logic operations on two  $n$ -bit digital words. The value of  $n$  can be 4, 8, 16, 32, or 64. Typical operations performed by the ALU are addition, subtraction, ANDing, ORing, and comparison of two  $n$ -bit digital words. The size of the ALU defines the size of the computer. For example, a 32-bit computer contains a 32-bit ALU.
- A *microprocessor* is the CPU of a microcomputer contained in a single chip and must be interfaced with peripheral support chips in order to function. In general, the CPU contains several *registers* (memory elements), the ALU, and the control unit. Note that the control unit translates instructions and performs the desired task. The number of peripheral devices depends upon the particular application involved and even varies within one application. As the microprocessor industry matures, more of these functions are being integrated onto chips in order to reduce the system package count. In general, a *microcomputer* typically consists of a microprocessor (CPU) chip,

input and output chips, and memory chips in which programs (instructions and data) are stored. Note that a *microcontroller*, on the other hand, is implemented in a single chip containing typically a CPU, memory, I/O, timer, A/D and D/A converter circuits. Throughout this book the terms “computer” and “CPU” will be used interchangeably with “Microcomputer” and “Microprocessor” respectively.

- An *address* is a pattern of 0's and 1's that represents a specific location of memory or a particular I/O device. Typical 8-bit microprocessors have 16 address lines, and, these 16 lines can produce  $2^{16}$  unique 16-bit patterns from 0000000000000000 to 1111111111111111, representing 65,536 different address combinations.
- *Read-only memory (ROM)* is a storage medium for the groups of bits called *words*, and its contents cannot normally be altered once programmed. A typical ROM is fabricated on a chip and can store, for example, 2048 eight-bit words, which can be individually accessed by presenting one of 2048 addresses to it. This ROM is referred to as a 2K by 8-bit ROM. 10110111 is an example of an 8-bit word that might be stored in one location in this memory. A ROM is also a nonvolatile storage device, which means that its contents are retained in the event of power failure to the ROM chip. Because of this characteristic, ROMs are used to store programs (instructions and data) that must always be available to the microprocessor.
- *Random access memory (RAM)* is also a storage medium for groups of bits or words whose contents can not only be read but also altered at specific addresses. Furthermore, a RAM normally provides volatile storage, which means that its contents are lost in the event of a power failure. RAMs are fabricated on chips and have typical densities of 4096 bits to one megabit per chip. These bits can be organized in many ways, for example, as 4096-by-1-bit words, or as 2048-by-8-bit words. RAMs are normally used for the storage of temporary data and intermediate results as well as programs that can be reloaded from a back-up nonvolatile source. RAMs are capable of providing large storage capacity in the range of Megabits.
- A *register* can be considered as volatile storage for a number of bits. These bits may be entered into the register simultaneously (in parallel), or sequentially (serially) from right to left or from left to right, 1 bit at a time. An 8-bit register storing the bits 11110000 is represented as follows:

1	1	1	1	0	0	0	0
---	---	---	---	---	---	---	---

- The term *bus* refers to a number of conductors (wires) organized to provide a means of communication among different elements in a microcomputer system. The conductors in the bus can be grouped in terms of their functions. A microprocessor normally has an address bus, a data bus, and a control bus. The address bits to memory or to an external device are sent out on the address bus. Instructions from memory, and data to/from memory or external devices normally travel on the data bus. Control signals for the other buses and among system elements are transmitted on the control bus. Buses are sometimes bidirectional; that is, information can be transmitted in either direction on the bus, but normally only in one direction at a time.
- The *instruction set* of a microprocessor is the list of commands that the microprocessor is designed to execute. Typical instructions are ADD, SUBTRACT, and STORE. Individual instructions are coded as unique bit patterns, which are recognized and

executed by the microprocessor. If a microprocessor has 3 bits allocated to the representation of instructions, then the microprocessor will recognize a maximum of  $2^3$  or eight different instructions. The microprocessor will then have a maximum of eight instructions in its instruction set. It is obvious that some instructions will be more suitable to a particular application than others. For example, if a microprocessor is to be used in a calculating mode, instructions such as ADD, SUBTRACT, MULTIPLY, and DIVIDE would be desirable. In a control application, instructions inputting digitized signals into the processor and outputting digital control variables to external circuits are essential. The number of instructions necessary in an application will directly influence the amount of hardware in the chip set and the number and organization of the interconnecting bus lines.

- A microcomputer requires synchronization among its components, and this is provided by the *clock* or timing circuits. A clock is analogous to the heart beats of a human body.
- The *chip* is an integrated circuit (IC) package containing digital circuits.
- The term *gate* refers to digital circuits which perform logic operations such as AND, OR, and NOT. In an AND operation, the output of the AND gate is one if all inputs are one; the output is zero if one or more inputs are zero. The OR gate, on the other hand, provides a zero output if all inputs are zero; the output is one if one or more inputs are one. Finally, a NOT gate (also called an inverter) has one input and one output. The NOT gate produces one if the input is zero; the output is zero if the input is one.
- *Transistors* are basically electronic switching devices. There are two types of transistors. These are *bipolar junction transistors (BJTs)* and *metal-oxide semiconductor (MOS)* transistors. The operation of the BJT depends on the flow of two types of carriers: electrons (*n*-channel) and holes (*p*-channel), whereas the MOS transistor is unipolar and its operation depends on the flow of only one type of carrier, either electrons (*n*-channel) or holes (*p*-channel).
- The *speed power product (SPP)* is a measure of performance of a logic gate. It is expressed in picojoules (pJ). SPP is obtained by multiplying the speed (in ns) by the power dissipation (in mW) of a gate.

## 1.2 Design Levels

Three design levels can be defined for digital systems: systems level, logic level, and device level.

- *Systems level* is the type of design in which CPU, memory, and I/O chips are interfaced to build a computer.
- *Logic level*, on the other hand, is the design technique in which chips containing logic gates such as AND, OR, and NOT are used to design a digital component such as the ALU.
- Finally, *device level* utilizes transistors to design logic gates.

## 1.3 Combinational vs. Sequential Systems

Digital systems at the logic level can be classified into two types. These are *combinational* and *sequential*.

*Combinational* systems contain no memory whereas *sequential* systems require



memory to remember the present state in order to go to the next state. A binary adder capable of providing the sum upon application of the numbers to be added is an example of a combinational system. For example, consider a 4-bit adder. The inputs to this adder will be two 4-bit numbers; the output will be the 4-bit sum. In this case, the adder will generate the 4-bit sum output upon application of the two 4-bit inputs.

*Sequential* systems, on the other hand, require memory. The counter is an example of a sequential system. For instance, suppose that the counter is required to count in the sequence 0, 1, 2 and then repeat the sequence. In this case, the counter must have memory to remember the present count in order to go to the next. The counter must remember that it is at count 0 in order to go to the next count, 1. In order to count to 2, the counter must remember that it is counting 1 at the present state. In order to repeat the sequence, the counter must count back to 0 based on the present count, 2, and the process continues. A chip containing sequential circuit such as the counter will have a clock input pin.

In general, all computers contain both combinational and sequential circuits. However, most computers are regarded as clocked sequential systems. In these computers, almost all activities pertaining to instruction execution are synchronized with clocks.

1.4     **Digital Integrated Circuits**

The transistor can be considered as an electronic switch. The on and off states of a transistor are used to represent binary digits. Transistors, therefore, play an important role in the design of digital systems. This section describes the basic characteristics of digital devices and logic families. These include diodes, transistors, and a summary of digital logic families. These topics are covered from a very basic point of view. This will allow the readers with some background in digital devices to see how they are utilized in designing digital systems.

1.4.1     **Diodes**

A diode is an electronic switch. It is a two-terminal device. Figure 1.1 shows the symbolic representation.

The positive terminal (made with the *p*-type semiconductor material) is called the anode; the negative terminal (made with the *n*-type semiconductor material) is called

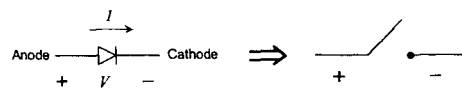


FIGURE 1.1     Symbolic representations of a diode

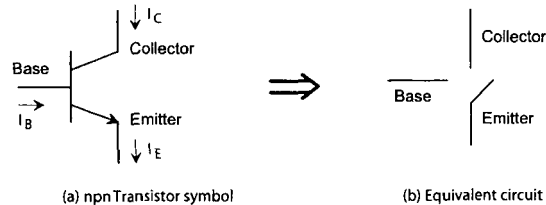


FIGURE 1.2     Symbolic representations of a *n*pn transistor

a cathode. When a voltage,  $V = 0.6$  volt is applied across the anode and the cathode, the switch closes and a current  $I$  flows from anode to the cathode.

### 1.4.2 Transistors

A bipolar junction transistor (BJT) or commonly called the transistor is also an electronic switch like the diode. Both electrons ( $n$ -channel) and holes ( $p$ -channel) are used for carrier flow; hence, the name “bipolar” is used. The BJT is used in transistor logic circuits that have several advantages over diode logic circuits. First of all, the transistor acts as a logic device called an inverter. Note that an inverter provides a LOW output for a HIGH input and a HIGH output for a LOW input. Secondly, the transistor is a current amplifier (buffer). Transistors can, therefore, be used to amplify these currents to control external devices such as a light emitting diode (LED) requiring high currents. Finally, transistor logic gates operate faster than diode gates.

There are two types of transistors, namely  $npn$  and  $pnp$ . The classification depends on the fabrication process.  $npn$  transistors are widely used in digital circuits.

Figure 1.2 shows the symbolic representation of an  $npn$  transistor. The transistor is a three-terminal device. These are base, emitter, and collector. The transistor is a current-controlled switch, which means that adequate current at the base will close the switch allowing a current to flow from the collector to the emitter. This current direction is identified on the  $npn$  transistor symbol in Figure 1.2(a) by a downward arrow on the emitter. Note that a base resistance is normally required to generate the base current.

The transistor has three modes of operation: cutoff, saturation, and active. In digital circuits, a transistor is used as a switch, which is either ON (closed) or OFF (open). When no base current flows, the emitter~collector switch is open and the transistor operates in the cutoff (OFF) mode. On the other hand, when a base current flows such that the voltage across the base and the emitter is at least 0.6 V, the switch closes. If the base current is further increased, there will be a situation in which  $V_{CE}$  (voltage across the collector and the emitter) attains a constant value of approximately 0.2 V. This is called the saturation (ON) mode of the transistor. The “active” mode is between the cutoff and saturation modes. In this mode, the base current ( $I_B$ ) is amplified so that the collector current,  $I_C = \beta I_B$ , where  $\beta$  is called the gain, and is in the range of 10 to 100 for typical transistors. Note that when the transistor reaches saturation, increasing  $I_B$  does not drop  $V_{CE}$  below  $V_{CE(\text{Sat.})}$  of 0.2 V. On the other hand,  $V_{CE}$  varies from 0.8 V to 5 V in the active mode. Therefore, the cutoff (OFF) and saturation (ON) modes of the transistor are used in designing digital circuits. The active mode of the transistor in which the transistor acts as a current amplifier (also called buffer) is used in digital output circuits.

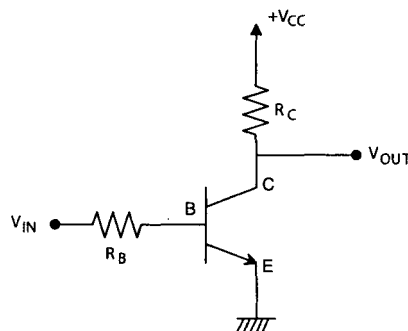


FIGURE 1.3 An inverter

TABLE 1.1      Current and Voltage Requirements of LEDs

LEDs	Red	Yellow	Green
Current	10 mA	10 mA	20 mA
Voltage	1.7 V	2.2V	2.4V

Operation of the Transistor as an Inverter

Figure 1.3 shows how to use the transistor as an inverter. When  $V_{IN} = 0$ , the transistor is in cutoff (OFF), and the collector-emitter switch is open. This means that no current flows from  $+V_{CC}$  to ground.  $V_{OUT}$  is equal to  $+V_{CC}$ . Thus,  $V_{OUT}$  is high.

On the other hand, when  $V_{IN}$  is HIGH, the emitter-collector switch is closed. A current flows from  $+V_{CC}$  to ground. The transistor operates in saturation, and  $V_{OUT} = V_{CE(Sat)} = 0.2\text{ V} \approx 0$ . Thus,  $V_{OUT}$  is basically connected to ground.

Therefore, for  $V_{IN} = \text{LOW}$ ,  $V_{OUT} = \text{HIGH}$ , and for  $V_{IN} = \text{HIGH}$ ,  $V_{OUT} = \text{LOW}$ . Hence, the *npn* transistor in Figure 1.3 acts as an inverter.

Note that  $V_{CC}$  is typically +5 V DC. The input voltage levels are normally in the range of 0 to 0.8 volts for LOW and 2 volts to 5 volts for HIGH. The output voltage levels, on the other hand, are normally 0.2 volts for LOW and 3.6 volts for HIGH.

Light Emitting Diodes (LEDs) and Seven Segment Displays

LEDs are extensively used as outputs in digital systems as status indicators. An LED is typically driven by low voltage and low current. This makes the LED a very attractive device for use with digital systems. Table 1.1 provides the current and voltage requirements of red, yellow, and green LEDs.

Basically, an LED will be ON, generating light, when its cathode is sufficiently negative with respect to its anode. A digital system such as a microcomputer can therefore

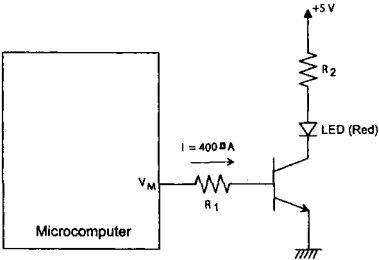


FIGURE 1.4      Microcomputer - LED interface

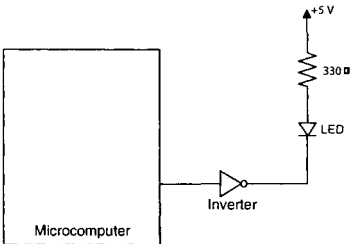


FIGURE 1.5      Microcomputer - LED interface via an inverter

light an LED either by grounding the cathode (if the anode is tied to +5 V) or by applying +5 V to the anode (if the cathode is grounded) through an appropriate resistor value. A typical hardware interface between a microcomputer and an LED is depicted in Figure 1.4.

A microcomputer normally outputs 400  $\mu\text{A}$  at a minimum voltage,  $V_M = 2.4$  volts for a HIGH. The red LED requires 10 mA at 1.7 volts. A buffer such as a transistor is required to turn the LED ON. Since the transistor is an inverter, a HIGH input to the transistor will turn the LED ON. We now design the interface; that is, the values of  $R_1$ ,  $R_2$ , and the gain  $\beta$  for the transistor will be determined.

A HIGH at the microcomputer output will turn the transistor ON into active mode. This will allow a path of current to flow from the +5 V source through  $R_2$  and the LED to the ground. The appropriate value of  $R_2$  needs to be calculated to satisfy the voltage and current requirements of the LED. Also, suppose that  $V_{BE} = 0.6$  V when the transistor is in active mode. This means that  $R_1$  needs to be calculated with the specified values of  $V_M = 2.4$  V and  $I = 400 \mu\text{A}$ . The values of  $R_1$ ,  $R_2$ , and  $\beta$  are calculated as follows:

$$R_1 = \frac{V_M - V_{BE}}{400 \mu\text{A}} = \frac{2.4 - 0.6}{400 \mu\text{A}} = 4.5 \text{ K}\Omega$$

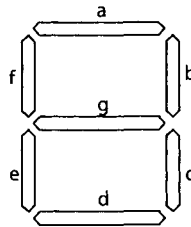
Assuming  $V_{CE} \cong 0$ ,

$$R_2 = \frac{5 - 1.7 - V_{CE}}{10 \text{ mA}} = \frac{5 - 1.7}{10 \text{ mA}} = 330 \Omega$$

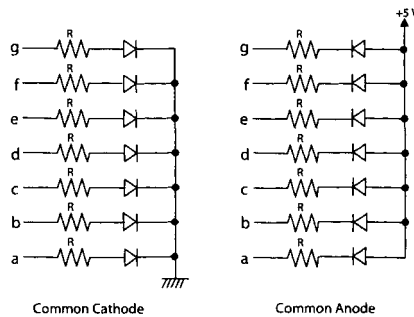
$$\beta = \frac{I_C}{I_B} = \frac{10 \text{ mA}}{400 \mu\text{A}} = \frac{10 \times 10^{-3}}{400 \times 10^{-6}} = 25$$

Therefore, the interface design is complete, and a transistor with a minimum  $\beta$  of 25,  $R_1 = 4.5 \text{ K}\Omega$ , and  $R_2 = 330 \Omega$  are required.


An inverting buffer chip such as 74LS368 can be used in place of a transistor in



**FIGURE 1.6** A seven-segment display



**FIGURE 1.7** Seven-segment display configurations

Figure 1.4. A typical interface of an LED to a microcomputer via an inverter is shown in Figure 1.5. Note that the transistor base resistance is inside the inverter. Therefore,  $R_1$  is not required to be connected to the output of the microcomputer. The symbol  is used to represent an inverter. Inverters will be discussed in more detail later. In figure 1.5, when the microcomputer outputs a HIGH, the transistor switch inside the inverter closes. A current flows from the +5 V source, through the 330-ohm resistor and the LED, into the ground inside the inverter. The LED is thus turned ON.

A seven-segment display can be used to display, for example, decimal numbers from 0 to 9. The name “seven segment” is based on the fact that there are seven LEDs — one in each segment of the display. Figure 1.6 shows a typical seven-segment display.

In Figure 1.6, each segment contains an LED. All decimal numbers from 0 to 9 can be displayed by turning the appropriate segment “ON” or “OFF”. For example, a zero can be displayed by turning the LED in segment *g* “OFF” and turning the other six LEDs in segments *a* through *f* “ON.” There are two types of seven segment displays. These are common cathode and common anode. Figure 1.7 shows these display configurations.

In a common cathode arrangement, the microcomputer can send a HIGH to light a segment and a LOW to turn it off. In a common anode configuration, on the other hand, the microcomputer sends a LOW to light a segment and a HIGH to turn it off. In both configurations,  $R = 330$  ohms can be used.

### Transistor Transistor Logic (TTL) and its Variations

The transistor transistor logic (TTL) family of chips evolved from diodes and transistors. This family used to be called DTL (diode transistor logic). The diodes were then replaced by transistors, and thus the name “TTL” evolved. The power supply voltage ( $V_{CC}$ ) for TTL is +5 V. The two logic levels are approximately 0 and 3.5 V.

There are several variations of the TTL family. These are based on the saturation mode (saturated logic) and active mode (nonsaturated logic) operations of the transistor. In the saturation mode, the transistor takes some time to come out of the saturation to switch to the cutoff mode. On the other hand, some TTL families define the logic levels in the active mode operation of the transistor and are called nonsaturated logic. Since the transistors do not go into saturation, these families do not have any saturation delay time for the switching operation. Therefore, the nonsaturated logic family is faster than saturated logic.

The saturated TTL family includes standard TTL (TTL), high-speed TTL (H-TTL), and low-power TTL (L-TTL). The nonsaturated TTL family includes Schottky TTL (S-TTL), low-power Schottky TTL (LS-TTL), advanced Schottky TTL (AS-TTL), and advanced low-power Schottky TTL (ALS-TTL). The development of LS-TTL made TTL, H-TTL, and L-TTL obsolete. Another technology, called emitter-coupled logic (ECL), utilizes nonsaturated logic. The ECL family provides the highest speed. ECL is used in digital systems requiring ultrahigh speed, such as supercomputers.

The important parameters of the digital logic families are fan-out, power dissipation, propagation delay, and noise margin.

Fan-out is defined as the maximum number of inputs that can be connected to the output of a gate. It is expressed as a number. The output of a gate is normally connected to the inputs of other similar gates. Typical fan-out for TTL is 10. On the other hand, fan-outs for S-TTL, LS-TTL, and ECL, are 10, 20, and 25, respectively.

*Power dissipation* is the power (milliwatts) required to operate the gate. This power must be supplied by the power supply and is consumed by the gate. Typical power

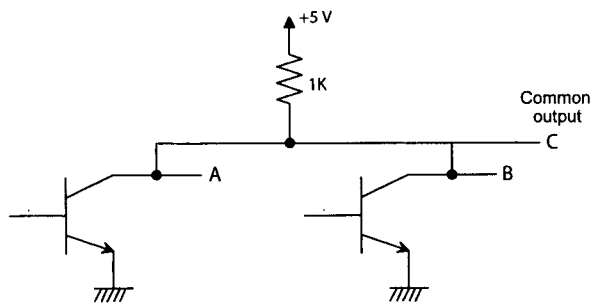


FIGURE 1.8 Two open-collector outputs A and B tied together

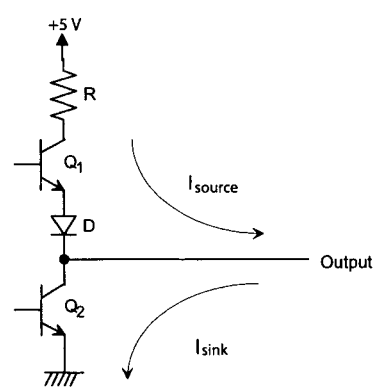


FIGURE 1.9 TTL Totem-pole output

consumed by TTL is 10 mW. On the other hand, S-TTL, LS-TTL, and ECL absorb 22 mW, 2 mW, and 25 mW respectively.

*Propagation delay* is the time required for a signal to travel from input to output when the binary output changes its value. Typical propagation delay for TTL is 10 nanoseconds (ns). On the other hand, S-TTL, LS-TTL, and ECL have propagation delays of 3 ns, 10 ns, and 2 ns, respectively.

*Noise margin* is defined as the maximum voltage due to noise that can be added to the input of a digital circuit without causing any undesirable change in the circuit output. Typical noise margin for TTL is 0.4 V. Noise margins for S-TTL, LS-TTL, and ECL are 0.4 V, 0.4 V, and 0.2 V , respectively.

**TTL Outputs**

There are three types of output configurations for TTL. These are open-collector output, totem-pole output, and tristate (three-state) output.

The open-collector output means that the TTL output is a transistor with nothing connected to the collector. The collector voltage provides the output of the gate. For the open-collector output to work properly, a resistor (called the pullup resistor), with a value of typically 1 Kohm, should be connected between the open collector output and a +5 V power supply.

If the outputs of several open-collector gates are tied together with an external