



WILEY-
INDIA
EDITION

EMBEDDED SYSTEM DESIGN

A Unified Hardware / Software
Introduction

WILEY
STUDENT
EDITION

RESTRICTED!
FOR SALE ONLY IN
INDIA, BANGLADESH, NEPAL,
PAKISTAN, SRI LANKA
& BHUTAN

Frank Vahid / Tony Givargis

EMBEDDED SYSTEM DESIGN : A UNIFIED HARDWARE / SOFTWARE INTRODUCTION

Third Edition

Copyright © 2002 by John Wiley & Sons, Inc. All rights reserved.

Authorized reprint by Wiley India (P.) Ltd., 4435/7, Ansari Road, Daryaganj, New Delhi 110 002.

All rights reserved. AUTHORIZED REPRINT OF THE EDITION PUBLISHED BY JOHN WILEY & SONS INC., U.K. No part of this book may be reproduced in any form without the written permission of the publisher.

Limits of Liability/Disclaimer of Warranty: The publisher and the author make no representation or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Website is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Website may provide or recommendations it may make. Further, readers should be aware that Internet Websites listed in this work may have changed or disappeared between when this work was written and when it is read.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books. For more information about Wiley products, visit our website at www.wiley.com.

Reprint : 2009

Printed at : Print India Press, Delhi

ISBN : 978-81-265-0837-2

Contents

Preface	vii
Purpose	vii
Coverage	vii
How to Use This Book	viii
Laboratory	ix
Additional Materials	x
Acknowledgments	x
About the Authors	xi
CHAPTER 1: Introduction	1
1.1 Embedded Systems Overview	1
1.2 Design Challenge — Optimizing Design Metrics	4
Common Design Metrics	4
The Time-to-Market Design Metric	6
The NRE and Unit Cost Design Metrics	7
The Performance Design Metric	8
1.3 Processor Technology	9
General-Purpose Processors — Software	9
Single-Purpose Processors — Hardware	10
Application-Specific Processors	12
1.4 IC Technology	13
Full-Custom/VLSI	13
Semicustom ASIC (Gate Array and Standard Cell)	13
PLD	14
Trends	14
1.5 Design Technology	16
Compilation/Synthesis	17
Libraries/IP	18

Test/Verification	18
More Productivity Improvers	18
Trends	19
1.6 Trade-offs	19
Design Productivity Gap	22
1.7 Summary and Book Outline	24
1.8 References and Further Reading	25
1.9 Exercises	25
CHAPTER 2: Custom Single-Purpose Processors: Hardware	29
2.1 Introduction	29
2.2 Combinational Logic	30
Transistors and Logic Gates	30
Basic Combinational Logic Design	32
RT-Level Combinational Components	33
2.3 Sequential Logic	34
Flip-Flops	34
RT-Level Sequential Components	35
Sequential Logic Design	36
2.4 Custom Single-Purpose Processor Design	38
2.5 RT-Level Custom Single-Purpose Processor Design	44
2.6 Optimizing Custom Single-Purpose Processors	47
Optimizing the Original Program	47
Optimizing the FSMD	48
Optimizing the Datapath	50
Optimizing the FSM	50
2.7 Summary	51
2.8 References and Further Reading	51
2.9 Exercises	52
CHAPTER 3: General-Purpose Processors: Software	55
3.1 Introduction	55
3.2 Basic Architecture	56
Datapath	56
Control Unit	57
Memory	58
3.3 Operation	59
Instruction Execution	59
Pipelining	60

Superscalar and VLIW Architectures	61
3.4 Programmer's View	61
Instruction Set	62
Program and Data Memory Space	64
Registers	64
I/O	65
Interrupts	65
Example: Assembly-Language Programming of Device Drivers	66
Operating System	67
3.5 Development Environment	69
Design Flow and Tools	69
Example: Instruction-Set Simulator for a Simple Processor	71
Testing and Debugging	71
3.6 Application-Specific Instruction-Set Processors (ASIPs)	74
Microcontrollers	74
Digital Signal Processors (DSP)	75
Less-General ASIP Environments	75
3.7 Selecting a Microprocessor	75
3.8 General-Purpose Processor Design	77
3.9 Summary	80
3.10 References and Further Reading	80
3.11 Exercises	81
CHAPTER 4: Standard Single-Purpose Processors: Peripherals	83
4.1 Introduction	83
4.2 Timers, Counters, and Watchdog Timers	84
Timers and Counters	84
Example: Reaction Timer	87
Watchdog Timers	88
Example: ATM Timeout Using a Watchdog Timer	89
4.3 UART	90
4.4 Pulse Width Modulators	92
Overview	92
Example: Controlling a DC Motor Using a PWM	94
4.5 LCD Controllers	95
Overview	95
Example: LCD Initialization	97
4.6 Keypad Controllers	97

4.7	Stepper Motor Controllers	98
	Overview	98
	Example: Using a Stepper Motor Driver	99
	Example: Controlling a Stepper Motor Directly	101
4.8	Analog-to-Digital Converters	102
	Example: Successive Approximation	103
4.9	Real-Time Clocks	105
4.10	Summary	106
4.11	References and Further Reading	106
4.12	Exercises	107
CHAPTER 5: Memory		109
5.1	Introduction	109
5.2	Memory Write Ability and Storage Permanence	111
	Write Ability	111
	Storage Permanence	112
	Trade-offs	112
5.3	Common Memory Types	112
	Introduction to “Read-Only” Memory — ROM	112
	Mask-Programmed ROM	114
	OTP ROM — One-Time Programmable ROM	114
	EPROM — Erasable Programmable ROM	115
	EEPROM — Electrically Erasable Programmable ROM	116
	Flash Memory	117
	Introduction to Read-Write Memory — RAM	118
	SRAM — Static RAM	119
	DRAM — Dynamic RAM	120
	PSRAM — Pseudo-Static RAM	120
	NVRAM — Nonvolatile RAM	120
	Example: HM6264 and 27C256 RAM/ROM Devices	120
	Example: TC55V2325FF-100 Memory Device	122
5.4	Composing Memory	123
5.5	Memory Hierarchy and Cache	125
	Cache Mapping Techniques	126
	Cache-Replacement Policy	128
	Cache Write Techniques	128
	Cache Impact on System Performance	128
5.6	Advanced RAM	130

The Basic DRAM	130
Fast Page Mode DRAM (FPM DRAM)	131
Extended Data Out DRAM (EDO DRAM)	132
Synchronous (S) and Enhanced Synchronous (ES) DRAM	132
Rambus DRAM (RDRAM)	133
DRAM Integration Problem	133
Memory Management Unit (MMU)	134
5.7 Summary	134
5.8 References and Further Reading	135
5.9 Exercises	135
CHAPTER 6: Interfacing	137
6.1 Introduction	137
6.2 Communication Basics	138
Basic Terminology	138
Basic Protocol Concepts	140
Example: The ISA Bus Protocol — Memory Access	143
6.3 Microprocessor Interfacing: I/O Addressing	144
Port and Bus-Based I/O	144
Memory-Mapped I/O and Standard I/O	145
Example: The ISA Bus Protocol — Standard I/O	147
Example: A Basic Memory Protocol	147
Example: A Complex Memory Protocol	148
6.4 Microprocessor Interfacing: Interrupts	148
6.5 Microprocessor Interfacing: Direct Memory Access	153
Example: DMA I/O and the ISA Bus Protocol	158
6.6 Arbitration	159
Priority Arbiter	160
Daisy-Chain Arbitration	160
Network-Oriented Arbitration Methods	162
Example: Vectored Interrupt Using an Interrupt table	162
6.7 Multilevel Bus Architectures	164
6.8 Advanced Communication Principles	166
Parallel Communication	166
Serial Communication	166
Wireless Communication	167
Layering	168
Error Detection and Correction	168

6.9	Serial Protocols	169
	I ² C	169
	CAN	171
	FireWire	172
	USB	172
6.10	Parallel Protocols	173
	PCI Bus	173
	ARM Bus	173
6.11	Wireless Protocols	174
	IrDA	174
	Bluetooth	174
	IEEE 802.11	174
6.12	Summary	176
6.13	References and Further Reading	176
6.14	Exercises	176
CHAPTER 7: Digital Camera Example		179
7.1	Introduction	179
7.2	Introduction to a Simple Digital Camera	179
	User's Perspective	180
	Designer's Perspective	180
7.3	Requirements Specification	185
	Nonfunctional Requirements	186
	Informal Functional Specification	187
	Refined Functional Specification	187
7.4	Design	194
	Implementation 1: Microcontroller Alone	195
	Implementation 2: Microcontroller and CCDPP	195
	Implementation 3: Microcontroller and CCDPP/Fixed-Point DCT	200
	Implementation 4: Microcontroller and CCDPP/DCT	203
7.5	Summary	205
7.6	References and Further Reading	205
7.7	Exercises	205
CHAPTER 8: State Machine and Concurrent Process Models		207
8.1	Introduction	207
8.2	Models vs. Languages, Text vs. Graphics	209
	Models vs. Languages	209
	Textual Languages vs. Graphical Languages	210



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

Temporal and Spatial Thinking	295
11.3 Verification: Hardware/Software Co-Simulation	296
Formal Verification and Simulation	296
Simulation Speed	298
Hardware-Software Co-Simulation	299
Emulators	301
11.4 Reuse: Intellectual Property Cores	301
Hard, soft and firm cores	302
New Challenges Posed by Cores to Processor Providers	302
New Challenges Posed by Cores to Processor Users	303
11.5 Design Process Models	304
11.6 Summary	306
11.7 Book Summary	307
11.8 References and Further Reading	307
11.9 Exercises	308
APPENDIX A: Online Resources	311
A.1 Introduction	311
A.2 Summary of the ESD Web Page	312
A.3 Lab Resources	312
Chapter 2	312
Chapter 3	314
Chapter 4	314
Chapter 5	315
Chapter 6	315
Chapter 7	315
A.4 About the Book Cover	315
Outdoors	315
Indoors	316
Index	319

CHAPTER 1: *Introduction*



- 1.1 Embedded Systems Overview
 - 1.2 Design Challenge – Optimizing Design Metrics
 - 1.3 Processor Technology
 - 1.4 IC Technology
 - 1.5 Design Technology
 - 1.6 Tradeoffs
 - 1.7 Summary and Book Outline
 - 1.8 References and Further Reading
 - 1.9 Exercises
-

1.1 Embedded Systems Overview

Computing systems are everywhere. It is probably no surprise that millions of computing systems are built every year, destined for desktop computers like personal computers, laptops, workstations, mainframes, and servers. What may be surprising is that billions of computing systems are built every year for a very different purpose: They are embedded within larger electronic devices, repeatedly carrying out a particular function, often going completely unrecognized by the device's user. Creating a precise definition of such embedded computing systems, or simply *embedded systems*, is not an easy task. We might try the following definition: an embedded system is nearly any computing system other than a desktop computer. That definition isn't perfect, but it may be as close as we'll get. We can better understand such systems by examining common examples and common characteristics. Such examination will reveal major challenges facing designers of embedded systems.

Embedded systems are found in a variety of common electronic devices, such as consumer electronics (cell phones, pagers, digital cameras, camcorders, videocassette recorders, portable video games, calculators, and personal digital assistants), home appliances (microwave ovens, answering machines, thermostats, home security systems, washing machines, and lighting systems), office automation (fax machines, copiers, printers, and scanners), business equipment (cash registers, curbside check-in, alarm systems, card readers,

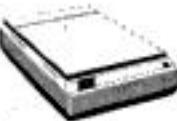
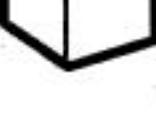
Anti-lock brakes	Modems	
Auto-focus cameras	MPEG decoders	
Automatic teller machines	Network cards	
Automatic toll systems	Network switches/routers	
Automatic transmission	On-board navigation	
Avionic systems	Pagers	
Battery chargers	Photocopiers	
Camcorders	Point-of-sale systems	
Cell phones	Portable video games	
Cell phone base stations	Printers	
Cordless phones	Satellite phones	
Cruise control	Scanners	
Curbside check-in systems	Smart ovens/dishwashers	
Digital cameras	Speech recognizers	
Disk drives	Stereo systems	
Electronic card readers	Teleconferencing systems	
Electronic instruments	Televisions	
Electronic toys/games	Temperature controllers	
Factory control	Theft tracking systems	
Fax machines	TV set-top boxes	
Fingerprint identifiers	VCR's, DVD players	
Home security systems	Video game consoles	
Life-support systems	Video phones	
Medical testing systems	Washers and dryers	

Figure 1.1: A short list of embedded systems.

product scanners, and automated teller machines), and automobiles (transmission control, cruise control, fuel injection, antilock brakes, and active suspension). Figure 1.1 is a short list of embedded system examples; a more complete list would require many pages. One might say that nearly any device that runs on electricity either already has or will soon have a computing system embedded within it. Although embedded computers typically cost far less than desktop computers, their quantities are huge. For example, in 1999 a typical American household may have had one desktop computer, but each one had between 35 and 50 embedded computers, with that number expected to rise to nearly 300 by 2004. Furthermore, the average 1998 car had 50 embedded computers costing several hundred dollars in all, with an annual cost growth rate of 17%. Several billion embedded microprocessor units were sold annually in recent years, compared to a few hundred million desktop microprocessor units.

Embedded systems have several common characteristics that distinguish such systems from other computing systems:

1. *Single-functioned*: An embedded system usually executes a specific program repeatedly. For example, a pager is always a pager. In contrast, a desktop system executes a variety of programs, like spreadsheets, word processors, and video games, with new programs added frequently. Of course, there are exceptions. One case is where an embedded system's program is updated with a newer program version. For example, some cell phones can be updated in such a manner. A second case is where

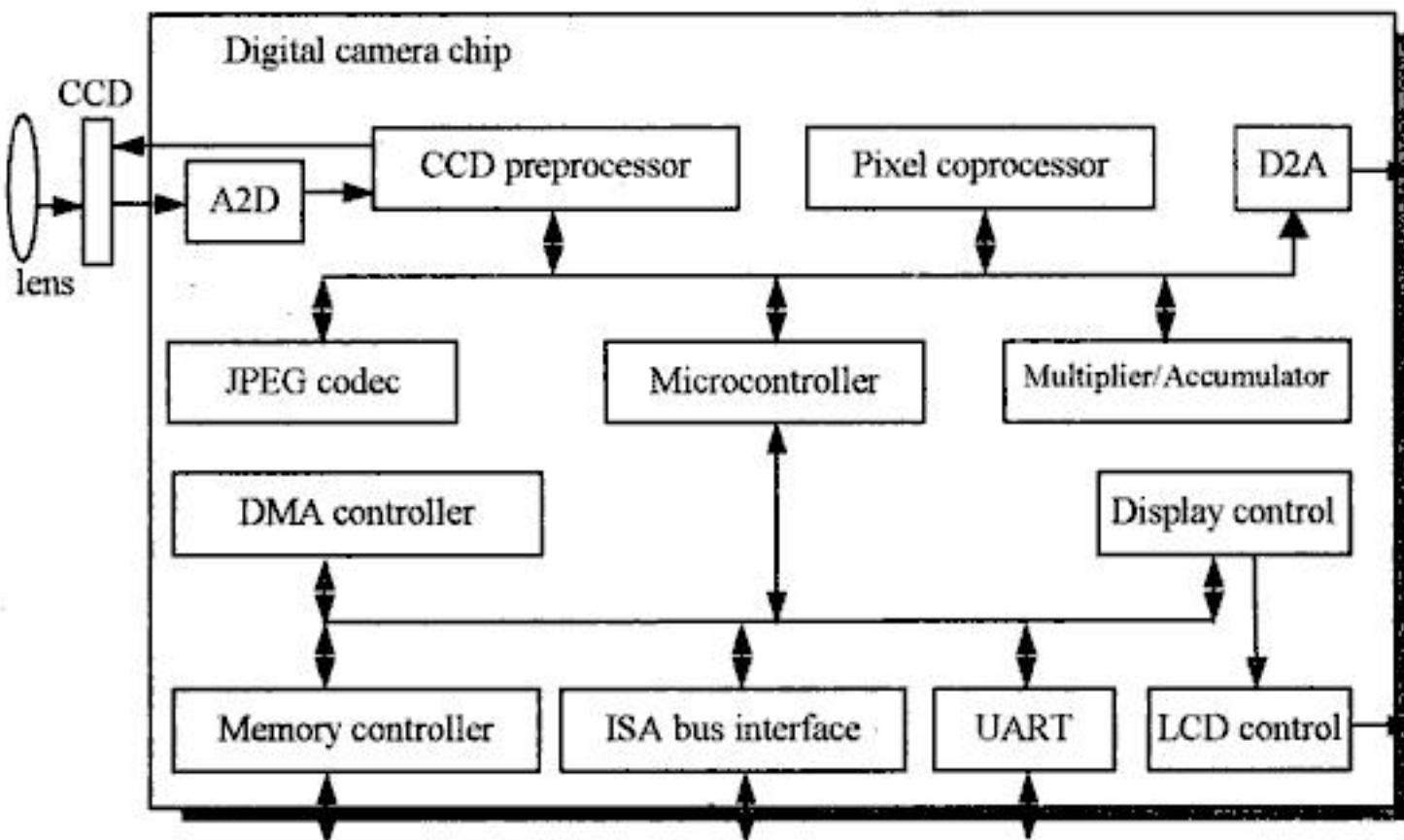
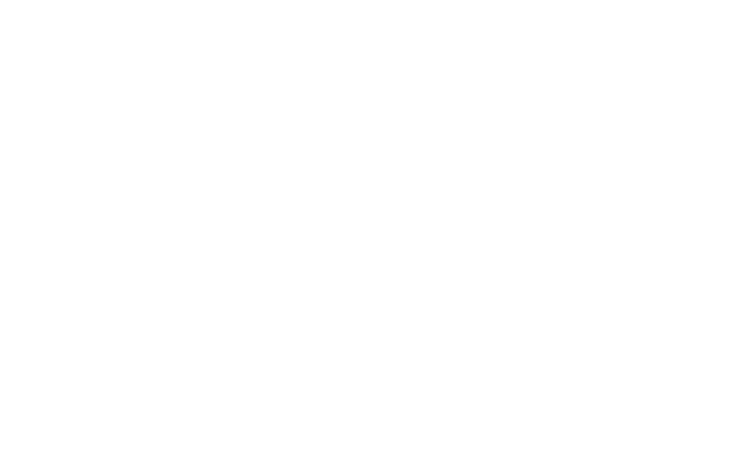


Figure 1.2: An embedded system example — a digital camera.

several programs are swapped in and out of a system due to size limitations. For example, some missiles run one program while in cruise mode, then load a second program for locking onto a target. Nevertheless, we can see that even these exceptions represent systems with a specific function.

2. *Tightly constrained*: All computing systems have constraints on design metrics, but those on embedded systems can be especially tight. A design metric is a measure of an implementation's features, such as cost, size, performance, and power. Embedded systems often must cost just a few dollars, must be sized to fit on a single chip, must perform fast enough to process data in real time, and must consume minimum power to extend battery life or prevent the necessity of a cooling fan.
3. *Reactive and real time*: Many embedded systems must continually react to changes in the system's environment and must compute certain results in real time without delay. For example, a car's cruise controller continually monitors and reacts to speed and brake sensors. It must compute acceleration or deceleration amounts repeatedly within a limited time; a delayed computation could result in a failure to maintain control of the car. In contrast, a desktop system typically focuses on computations, with relatively infrequent (from the computer's perspective) reactions to input devices. In addition, a delay in those computations, while perhaps inconvenient to the computer user, typically does not result in a system failure.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

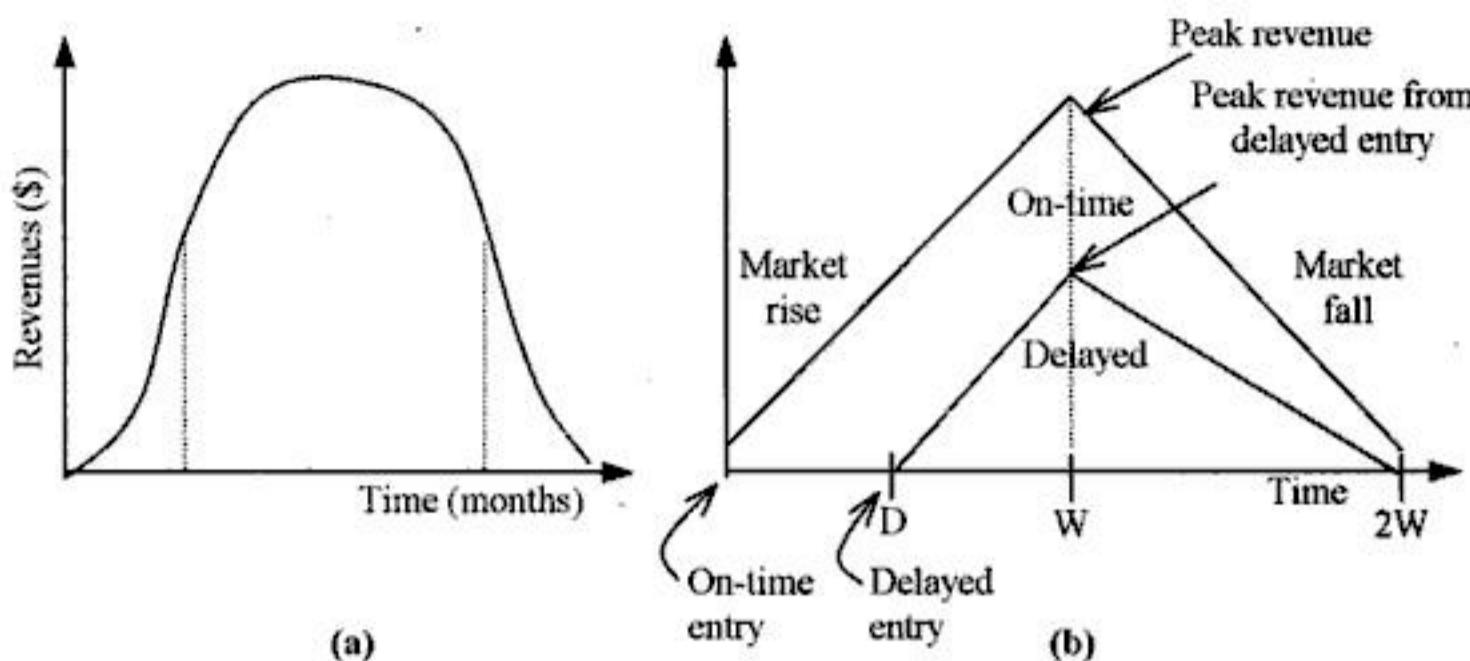


Figure 1.4: Time-to-market: (a) market window, (b) simplified revenue model for computing revenue loss from delayed entry.

pins pop out. To best meet this optimization challenge, the designer must be comfortable with a variety of hardware and software implementation technologies, and must be able to migrate from one technology to another, in order to find the best implementation for a given application and constraints. Thus, a designer cannot simply be a hardware expert or a software expert, as is commonly the case today; the designer must have expertise in both areas.

The Time-to-Market Design Metric

Most of these metrics are heavily constrained in an embedded system. The time-to-market constraint has become especially demanding in recent years. Introducing an embedded system to the marketplace early can make a big difference in the system's profitability, since market windows for products are becoming quite short, with such windows often measured in months. For example, Figure 1.4(a) shows a sample market window during which time a product would have highest sales. Missing this window, which means that the product begins being sold further to the right on the time scale, can mean significant loss in sales. In some cases, each day that a product is delayed from introduction to the market can translate to a one-million-dollar loss. The average time-to-market constraint has been reported as having shrunk to only 8 months!

Adding to the difficulty of meeting the time-to-market constraint is the fact that embedded system complexities are growing due to increasing IC capacities, as we will see later in this chapter. Such rapid growth in IC capacity translates into pressure on designers to add more functionality to a system. Thus, designers today are being asked to do more in less time.

Let's investigate the loss of revenue that can occur due to delayed entry of a product in the market. We'll use a simplified model of revenue that is shown in Figure 1.4(b). This model assumes the peak of the market occurs at the halfway point, denoted as W , of the product life, and that the peak is the same even for a delayed entry. The revenue for an

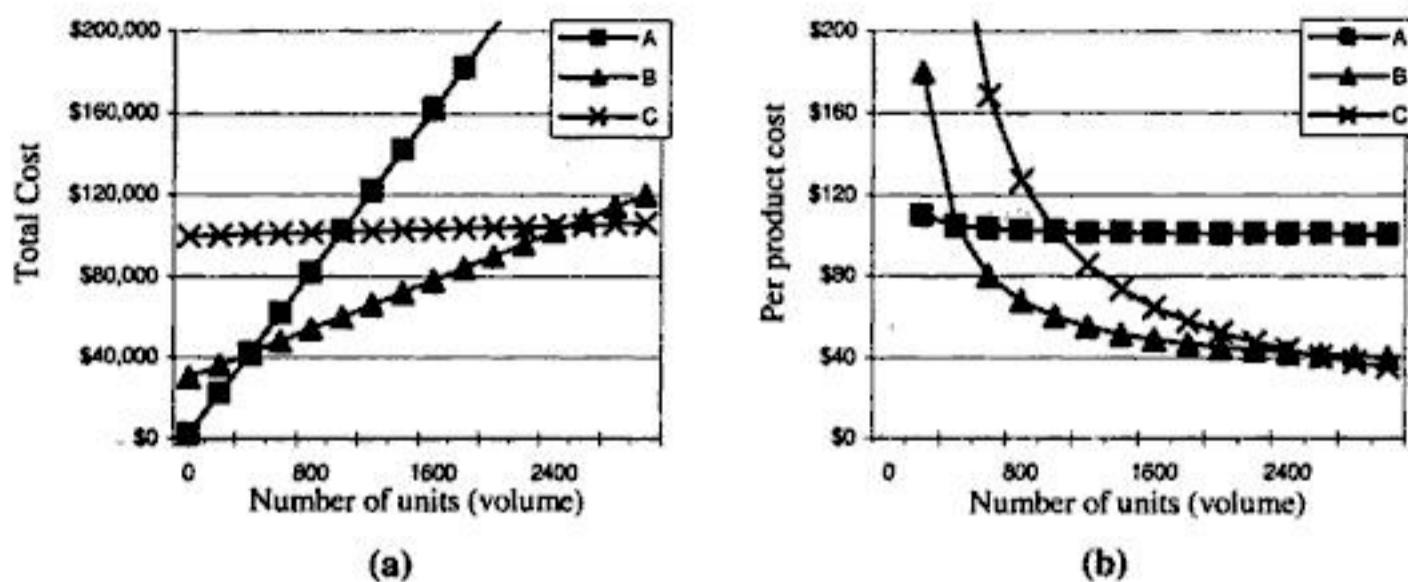


Figure 1.5: Costs for technologies A, B, and C as a function of volume: (a) total cost, (b) per-product cost.

on-time market entry is the area of the triangle labeled *On-time*, and the revenue for a delayed entry product is the area of the triangle labeled *Delayed*. The revenue loss for a delayed entry is just the difference of these two triangles' areas. Let's derive an equation for percentage revenue loss, which equals $((\text{On-time} - \text{Delayed}) / \text{On-time}) * 100\%$. For simplicity, we'll assume the market rise angle is 45 degrees, meaning the height of the triangle is W , and we leave as an exercise the derivation of the same equation for any angle. The area of the *On-time* triangle, computed as $\frac{1}{2} * \text{base} * \text{height}$, is thus $\frac{1}{2} * 2W * W$, or W^2 . The area of the *Delayed* triangle is $\frac{1}{2}(W - D + W) * (W - D)$. After algebraic simplification, we obtain the following equation for percentage revenue loss:

$$\text{percentage revenue loss} = (D(3W - D) / 2W^2) * 100\%$$

Consider a product whose lifetime is 52 weeks, so $W = 26$. According to the preceding equation, a delay of just $D = 4$ weeks results in a revenue loss of 22%, and a delay of $D = 10$ weeks results in a loss of 50%. Some studies claim that reaching market late has a larger negative effect on revenues than development cost overruns or even a product price that is too high.

The NRE and Unit Cost Design Metrics

As another exercise, let's consider NRE cost and unit cost in more detail. Suppose three technologies are available for use in a particular product. Assume that implementing the product using technology A would result in an NRE cost of \$2,000 and unit cost of \$100, that technology B would have an NRE cost of \$30,000 and unit cost of \$30, and that technology C would have an NRE cost of \$100,000 and unit cost of \$2. Ignoring all other design metrics, like time-to-market, the best technology choice will depend on the number of units we plan to produce. We illustrate this concept with the plot of Figure 1.5(a). For each of the three technologies, we plot total cost versus the number of units produced, where:

$$\text{total cost} = \text{NRE cost} + \text{unit cost} * \# \text{ of units}$$

We see from the plot that, of the three technologies, technology A yields the lowest total cost for low volumes, namely for volumes between 1 and 400. Technology B yields the lowest total cost for volumes between 400 and 2500. Technology C yields the lowest cost for volumes above 2500.

Figure 1.5(b) illustrates how larger volumes allow us to amortize NRE costs such that lower per-product costs result. The figure plots per-product cost versus volume, where:

$$\text{per-product cost} = \text{total cost} / \# \text{ of units} = \text{NRE cost} / \# \text{ of units} + \text{unit cost}$$

For example, for technology C and a volume of 200,000, the contribution to the per-product cost due to NRE cost is $\$100,000 / 200,000$, or $\$0.50$. So the per-product cost would be $\$0.50 + \$2 = \$2.50$. The larger the volume, the lower the per-product cost, since the NRE cost can be distributed over more products. The per-product cost for each technology approaches that technology's unit cost for very large volumes. So for very large volumes, numbering in the hundreds of thousands, we can approach a per-product cost of just $\$2$ — quite a bit less than the per-product cost of over $\$100$ for small volumes.

Clearly, one must consider the revenue impact of both time-to-market and per-product cost, as well as all the other relevant design metrics when evaluating different technologies.

The Performance Design Metric

Performance of a system is a measure of how long the system takes to execute our desired tasks. Performance is perhaps the most widely used design metric in marketing an embedded system, and also one of the most abused. Many metrics are commonly used in reporting system performance, such as clock frequency or instructions per second. However, what we really care about is how long the system takes to execute our application. For example, in terms of performance, we care about how long a digital camera takes to process an image. The camera's clock frequency or instructions per second are not the key issues — one camera may actually process images faster but have a lower clock frequency than another camera.

With that said, there are several measures of performance. For simplicity, suppose we have a single task that will be repeated over and over, such as processing an image in a digital camera. The two main measures of performance are:

- *Latency, or response time:* The time between the start of the task's execution and the end. For example, processing an image may take 0.25 second.
- *Throughput:* The number of tasks that can be processed per unit time. For example, a camera may be able to process 4 images per second.

However, note that throughput is not always just the number of tasks times latency. A system may be able to do better than this by using parallelism, either by starting one task before finishing the next one or by processing each task concurrently. A digital camera, for example, might be able to capture and compress the next image, while still storing the previous image to memory. Thus, our camera may have a latency of 0.25 second but a throughput of 8 images per second.

In embedded systems, performance at a very detailed level is also often of concern. In particular, two signal changes may have to be generated or measured within some number of nanoseconds.

Speedup is a common method of comparing the performance of two systems. The speedup of system A over system B is determined simply as:

$$\text{speedup of A over B} = \text{performance of A} / \text{performance of B}.$$

Performance could be measured either as latency or as throughput, depending on what is of interest. Suppose the speedup of camera A over camera B is 2. Then we also can say that A is 2 times faster than B and B is 2 times slower than A.

1.3 Processor Technology

We can define *technology* as a manner of accomplishing a task, especially using technical processes, methods, or knowledge. This book takes the perspective that three types of technologies are central to embedded system design: processor technologies, IC technologies, and design technologies. We describe all three briefly in this chapter and provide further details in subsequent chapters.

Processor technology relates to the architecture of the computation engine used to implement a system's desired functionality. Although the term *processor* is usually associated with programmable software processors, we can think of many other, nonprogrammable, digital systems as being processors also. Each such processor differs in its specialization towards a particular function (e.g., image compression), thus manifesting design metrics different than other processors. We illustrate this concept graphically in Figure 1.6. The application requires a specific embedded functionality, symbolized as a cross, such as the summing of the items in an array, as shown in Figure 1.6(a). Several types of processors can implement this functionality, each of which we now describe. We often use a collection of such processors to optimize a system's design metrics, as in our digital camera example.

General-Purpose Processors — Software

The designer of a *general-purpose processor*, or *microprocessor*, builds a programmable device that is suitable for a variety of applications to maximize the number of devices sold. One feature of such a processor is a program memory — the designer of such a processor does not know what program will run on the processor, so the program cannot be built into the digital circuit. Another feature is a general datapath — the datapath must be general enough to handle a variety of computations, so such a datapath typically has a large register file and one or more general-purpose arithmetic-logic units (ALUs). An embedded system designer, however, need not be concerned about the design of a general-purpose processor. An embedded system designer simply uses a general-purpose processor, by programming the processor's memory to carry out the required functionality. Many people refer to this part of an implementation as the "software" portion.

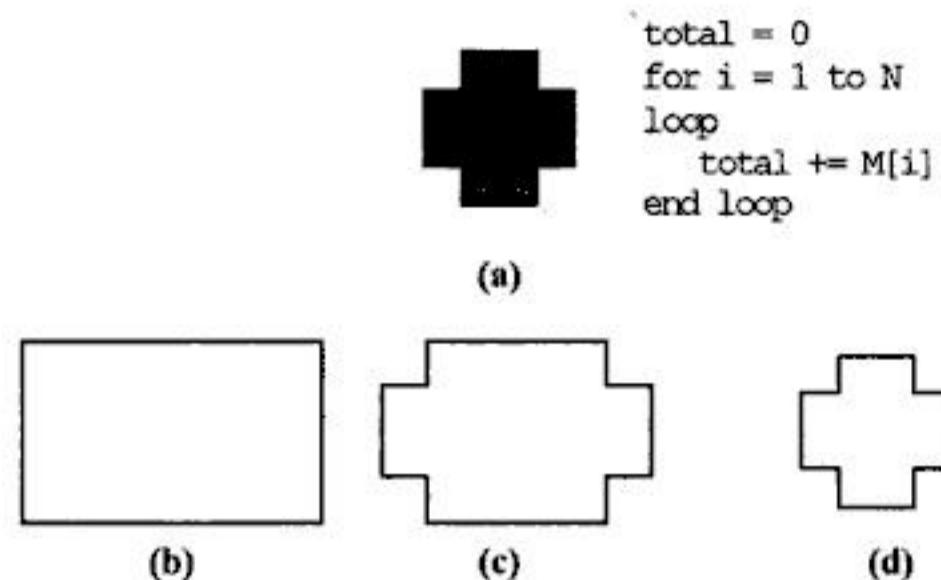


Figure 1.6: Processors vary in their customization for the problem at hand: (a) desired functionality, (b) general-purpose processor, (c) application-specific processor, (d) single-purpose processor.

Using a general-purpose processor in an embedded system may result in several design metric benefits. Time-to-market and NRE costs are low because the designer must only write a program but not do any digital design. Flexibility is high because changing functionality requires changing only the program. Unit cost may be low in small quantities compared with designing our own processor, since the general-purpose processor manufacturer sells large quantities to other customers and hence distributes the NRE cost over many units. Performance may be fast for computation-intensive applications, if using a fast processor, due to advanced architecture features and leading-edge IC technology.

However, there are also some design-metric drawbacks. Unit cost may be relatively high for large quantities, since in large quantities we could design our own processor and amortize our NRE costs such that our unit cost is lower. Performance may be slow for certain applications. Size and power may be large due to unnecessary processor hardware.

For example, we can use a general-purpose processor to carry out our array-summing functionality from the earlier example. Figure 1.6(b) illustrates that a general-purpose processor covers the desired functionality but not necessarily efficiently. Figure 1.7(a) shows a simple architecture of a general-purpose processor implementing the array-summing functionality. The functionality is stored in a program memory. The controller fetches the current instruction, as indicated by the program counter (PC), into the instruction register (IR). It then configures the datapath for this instruction and executes the instruction. It then determines the next instruction address, sets the PC to this address, and fetches again.

Single-Purpose Processors — Hardware

A *single-purpose processor* is a digital circuit designed to execute exactly one program. For example, consider the digital camera example of Figure 1.2. All of the components other than the microcontroller are single-purpose processors. The JPEG codec, for example, executes a single program that compresses and decompresses video frames. An embedded system



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

autoincrementing register, a path that allows us to add a register with a memory location in one instruction, fewer registers, and a simpler controller.

1.4 IC Technology

Every processor must eventually be implemented on an integrated circuit (IC). IC technology involves the manner in which we map a digital (gate-level) implementation onto an IC. An IC, often called a “chip,” is a semiconductor device consisting of a set of connected transistors and other devices. A number of different processes exist to build semiconductors, the most popular of which is complementary metal oxide semiconductor (CMOS).

IC technologies differ by how customized the IC is for a particular design. IC technology is independent from processor technology; any type of processor can be mapped to any type of IC technology, as illustrated in Figure 1.14.

To understand the differences among IC technologies, we must first recognize that semiconductors consist of numerous layers as illustrated in Figure 1.8. The bottom layers form the transistors. The middle layers form logic components. The top layers connect these components with wires. One way to create these layers is by depositing photo-sensitive chemicals on the chip surface and then shining light through masks to change regions of the chemicals. Thus, the task of building the layers is actually one of designing appropriate masks. A set of masks is often called a *layout*. The narrowest line that we can create on a chip is called the *feature size*, which today is well below one micrometer (submicron). For each IC technology, all layers must eventually be built to get a working IC; the question is who builds each layer and when.

Full-Custom/VLSI

In a full-custom IC technology, we optimize all layers for a particular embedded system’s digital implementation. Such optimization includes placing the transistors to minimize interconnection lengths, sizing the transistors to optimize signal transmissions and routing wires among the transistors. Once we complete all the masks, we send the mask specifications to a fabrication plant that builds the actual ICs. Full-custom IC design, often referred to as very large scale integration (VLSI) design, has a very high NRE cost and long turnaround times, typically many months before the IC becomes available, but can yield excellent performance with small size and power. It is usually used only in high-volume or extremely performance-critical applications.

Semicustom ASIC (Gate Array and Standard Cell)

In an application-specific IC (ASIC) technology, the lower layers are fully or partially built, leaving us to finish the upper layers. In a gate-array ASIC technology, the masks for the transistor and gate levels are already built (i.e., the IC already consists of arrays of gates). The remaining task is to connect these gates to achieve our particular implementation. In a standard-cell ASIC technology, logic-level cells, such as an AND gate or an



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

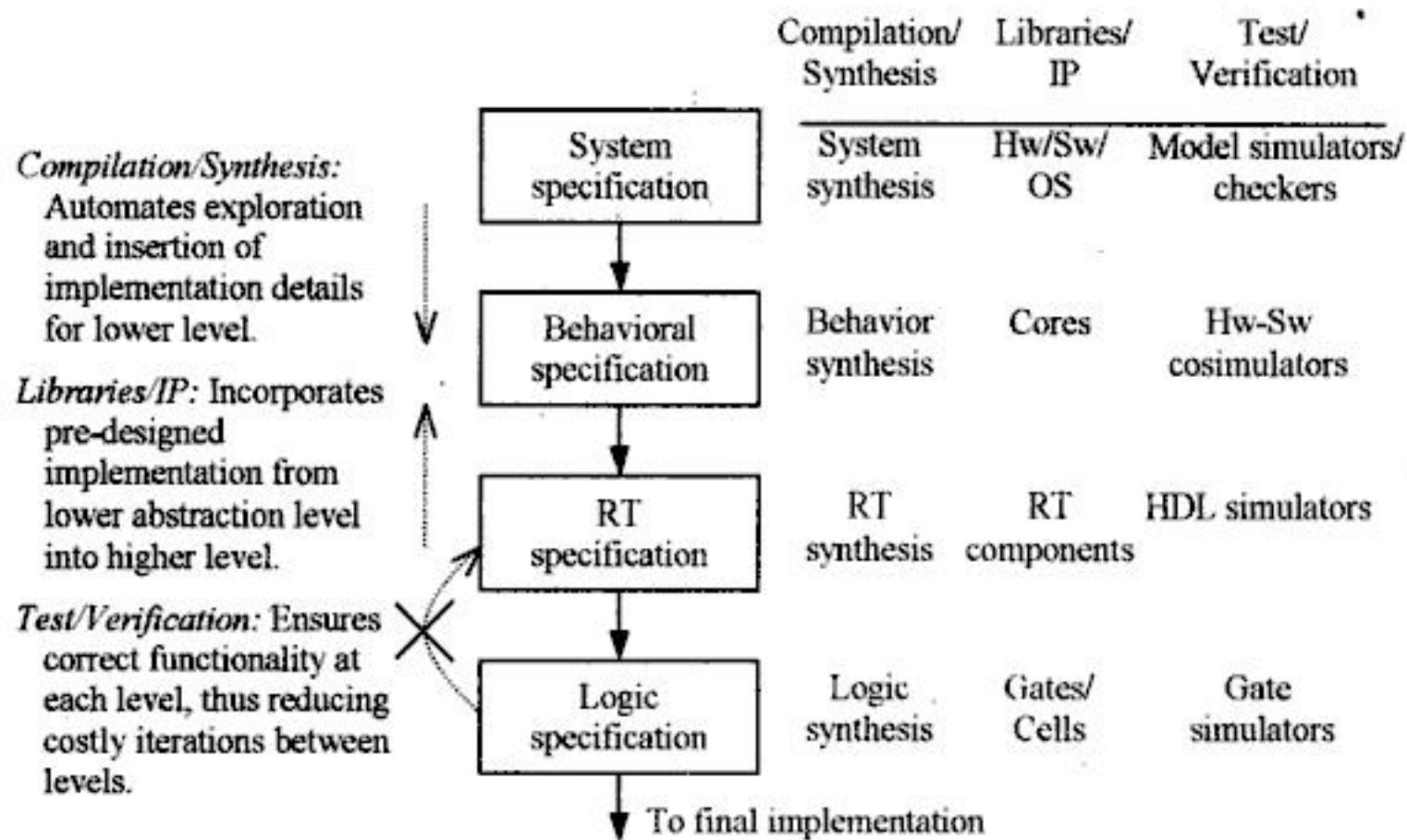


Figure 1.11: Ideal top-down design process, and productivity improvers.

There are three main approaches to improving the design process for increased productivity, which we label as compilation/synthesis, libraries/IP, and test/verification. Several other approaches also exist. We will discuss all of these approaches. Each approach can be applied at any of the four abstraction levels.

Compilation/Synthesis

Compilation/synthesis lets a designer specify desired functionality in an abstract manner and automatically generates lower-level implementation details. Describing a system at high abstraction levels can improve productivity by reducing the amount of details, often by an order of magnitude, that a designer must specify.

A logic synthesis tool converts Boolean expressions into a connection of logic gates, called a netlist. A register-transfer (RT) synthesis tool converts finite-state machines and register transfers into a datapath of RT components and a controller of Boolean equations. A behavioral synthesis tool converts a sequential program into finite-state machines and register transfers. Likewise, a software compiler converts a sequential program to assembly code, which is essentially register-transfer code. Finally, a system synthesis tool converts an abstract system specification into a set of sequential programs on general- and single-purpose processors.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

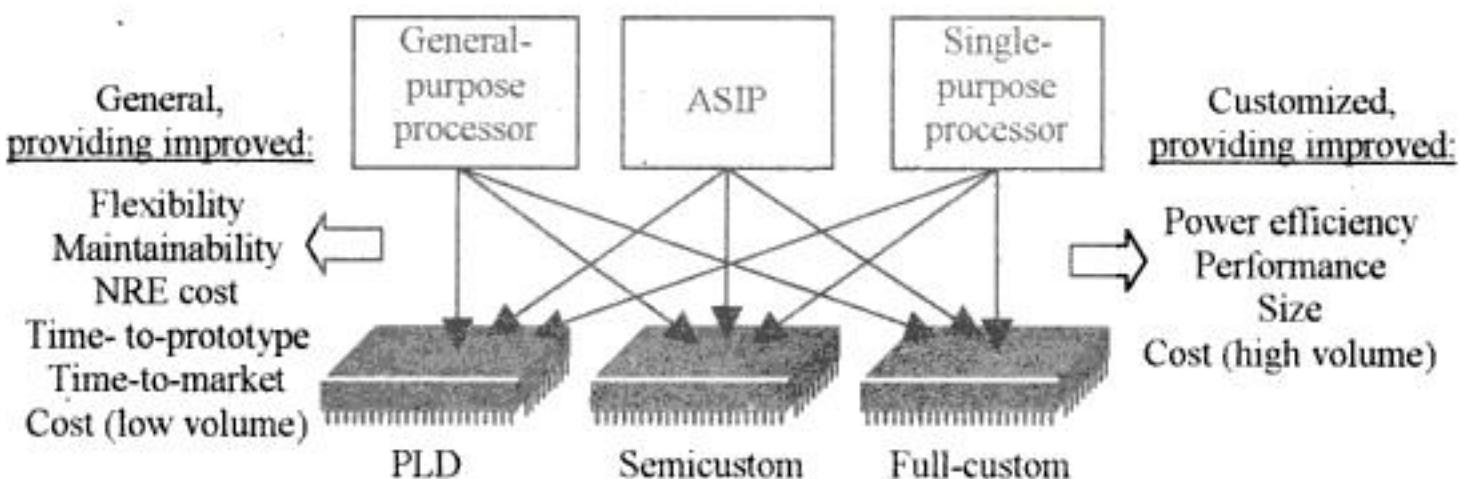


Figure 1.14: The independence of processor and IC technologies: Any processor technology can be mapped to any IC technology.

of abstraction, as illustrated in Figure 1.13. Thus, the starting point for either hardware or software is sequential programs, enhancing the view that system functionality can be implemented in hardware, software, or some combination thereof, leading to the following important point:

The choice of hardware versus software for a particular function is simply a trade-off among various design metrics, like performance, power, size, NRE cost, and especially flexibility; there is no fundamental difference between what hardware or software can implement.

Hardware/software codesign is the field that emphasizes a unified view of hardware and software, and develops synthesis tools and simulators that enable the co-development of systems using both hardware and software.

In general, we can view the basic design trade-off as general versus customized implementation, with respect to either processor technology or IC technology, as illustrated in Figure 1.14. The more general, programmable technologies on the left of the figure provide greater flexibility (a design can be reprogrammed relatively easily), reduced NRE cost (designing using those technologies is generally cheaper), faster time-to-prototype and time-to-market (since designing takes less time), and lower cost in low volumes (since the IC manufacturer distributes its IC NRE cost over large quantities of ICs). On the other hand, more customized technologies provide for better power efficiency, faster performance, reduced size, and lower cost in high volumes.

Recall that each of the three processor technologies can be implemented in any of the three IC technologies. For example, a general-purpose processor can be implemented on a PLD, semicustom, or full-custom IC. In fact, a company marketing a product, such as a set-top box or even a general-purpose processor, might first market a semicustom implementation to reach the market early, and then later introduce a full-custom implementation. They might also first map the processor to an older but more reliable technology, like 0.2 micron, and then later map it to a newer technology, like 0.08 micron. These two evolutions of mappings to a large extent explain why a general-purpose processor's



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

The chapter is mostly a review of the features of such processors; we assume the reader already has familiarity with programming such processors using structured languages. Chapter 4 covers standard single-purpose processors, describing a number of common peripherals used in embedded systems. Chapter 5 describes memories, which are components necessary to store data for processors. Chapter 6 describes buses, components necessary to communicate data among processors and memories, beginning with basic interfacing concepts, and introducing more advanced concepts and describing common buses. Chapter 7 provides an example of using processor technology to build an embedded system, a digital camera, illustrating the trade-offs of several different implementations.

Chapter 8 introduces some advanced techniques for programming embedded systems, including state machine models, and concurrent process models. It also introduces real-time systems. Chapter 9 discusses the very common class of embedded systems known as control systems, and introduces some design techniques used for such systems.

Chapter 10 describes the three main IC technologies with which we can implement the processor-based designs we learn to create in the earlier chapters. Finally, Chapter 11 summarizes key tools and advances in design technology and emphasizes the need for a new breed of engineers for embedded systems proficient with both software and hardware design.

1.8 References and Further Reading

- Brooks Jr., F.P., *The Mythical Man-Month*, anniversary edition. Reading, MA: Addison-Wesley, 1995. Original edition published in 1975.
- *EE Times*, Oct 11, 1999. Embedded Systems section.
- Midyear forecast – CEO Perspectives, *EE Times*, May 27, 1998, Issue 1009.
- Semiconductor Industry Association. *International Technology Roadmap for Semiconductors: 1999 edition*. Austin, TX: International SEMATECH, 1999.
- Debardeleben, J., Madisetti, V.K., and Gadiant, A.J., Incorporating Cost Modeling into Embedded System Design, *IEEE Design and Test of Computers*, July 1997, pp. 24-35. Includes discussion of revenue model.

1.9 Exercises

- 1.1 What is an embedded system? Why is it so hard to define?
- 1.2 List and define the three main characteristics of embedded systems that distinguish such systems from other computing systems.
- 1.3 What is a design metric?
- 1.4 List a pair of design metrics that may compete with one another, providing an intuitive explanation of the reason behind the competition.
- 1.5 What is a “market window” and why is it so important for products to reach the market early in this window?



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

CHAPTER 2: *Custom Single-Purpose Processors: Hardware*



- 2.1 Introduction
- 2.2 Combinational Logic
- 2.3 Sequential Logic
- 2.4 Custom Single-Purpose Processor Design
- 2.5 RT-Level Custom Single-Purpose Processor Design
- 2.6 Optimizing Custom Single-Purpose Processors
- 2.7 Summary
- 2.8 References and Further Reading
- 2.9 Exercises

2.1 Introduction

A *processor* is a digital circuit designed to perform computation tasks. A processor consists of a datapath capable of storing and manipulating data and a controller capable of moving data through the datapath. A general-purpose processor is designed such that it can carry out a wide variety of computation tasks, which are described by a set of programmer-provided instructions. In contrast, a single-purpose processor is designed specifically to carry out a particular computation task. While some tasks are so common that we can purchase standard single-purpose processors to implement those tasks, others are unique to a particular embedded system. Such custom tasks may be best implemented using custom single-purpose processors that we design ourselves.

An embedded system designer may obtain several benefits by choosing to use a custom single-purpose processor rather than a general-purpose processor to implement a computation task.

First, performance may be faster, due to fewer clock cycles resulting from a customized datapath, and due to shorter clock cycles resulting from simpler functional units, fewer multiplexors, or simpler controller logic. Second, size may be smaller, due to a simpler



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

inputs. Such a circuit has no memory of past inputs. We can use a simple technique to design a combinational circuit from our basic logic gates, as illustrated in Figure 2.4. We start with a problem description, which describes the outputs in terms of the inputs, as in Figure 2.4(a). We translate that description to a truth table, with all possible combinations of input values on the left and desired output values for each combination on the right, as in Figure 2.4(b). For each output column, we can derive an output equation, with one equation term per row, as in Figure 2.4(c). We can then translate these equations to a circuit diagram. However, we usually want to minimize the logic gates in the circuit. We can minimize the output equations by algebraically manipulating the equations. Alternatively, we can use Karnaugh maps, as shown in Figure 2.4(d). Once we've obtained the desired output equations, we can draw the circuit diagram, as shown in Figure 2.4(e).

RT-Level Combinational Components

Although we can design all combinational circuits in this manner, large circuits would be very complex to design. For example, a circuit with 16 inputs would have 2^{16} , or 64K, rows in its truth table. One way to reduce the complexity is to use combinational components that are more powerful than logic gates. Figure 2.5 shows several such combinational components, often called *register-transfer*, or RT, level components. We now describe each briefly.

A *multiplexor*, sometimes called a *selector*, allows only one of its data inputs I_m to pass through to the output O . Thus, a multiplexor acts much like a railroad switch, allowing only one of multiple input tracks to connect to a single output track. If there are m data inputs, then there are $\log_2(m)$ select lines S . We call this an m -by-1 multiplexor, meaning m data inputs, and 1 data output. The binary value of S determines which data input passes through; 00...00 means I_0 passes through, 00...01 means I_1 passes through, 00...10 means I_2 passes through, and so on. For example, an 8×1 multiplexor has eight data inputs and thus three select lines. If those three select lines have values of 110, then I_6 will pass through to the output. So if I_6 were 1, then the output would be 1; if I_6 were 0, then the output would be 0. We commonly use a more complex device called an n -bit multiplexor, in which each data input as well as the output consist of n lines. Suppose the previous example used a 4-bit 8×1 multiplexor. Thus, if I_6 were 1110, then the output would be 1110. Note that n is independent of the number of select lines.

Another combinational component is a decoder. A *decoder* converts its binary input I into a one-hot output O . "One-hot" means that exactly one of the output lines can be 1 at a given time. Thus, if there are n outputs, then there must be $\log_2(n)$ inputs. We call this a $\log_2(n) \times n$ decoder. For example, a 3×8 decoder has three inputs and eight outputs. If the input were 000, then the output O_0 would be 1 and all other outputs would be 0. If the input were 001, then the output O_1 would be 1, and so on. A common feature on a decoder is an extra input called *enable*. When enable is 0, all outputs are 0. When enable is 1, the decoder functions as before.

An *adder* adds two n -bit binary inputs A and B , generating an n -bit output sum along with an output carry. For example, a 4-bit adder would have a 4-bit A input, a 4-bit B input, a 4-bit *sum* output, and a 1-bit *carry* output. If A were 1010 and B were 1001, then *sum* would be 0011 and *carry* would be 1. An adder often comes with a carry input also, so that such adders can be cascaded to create larger adders.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

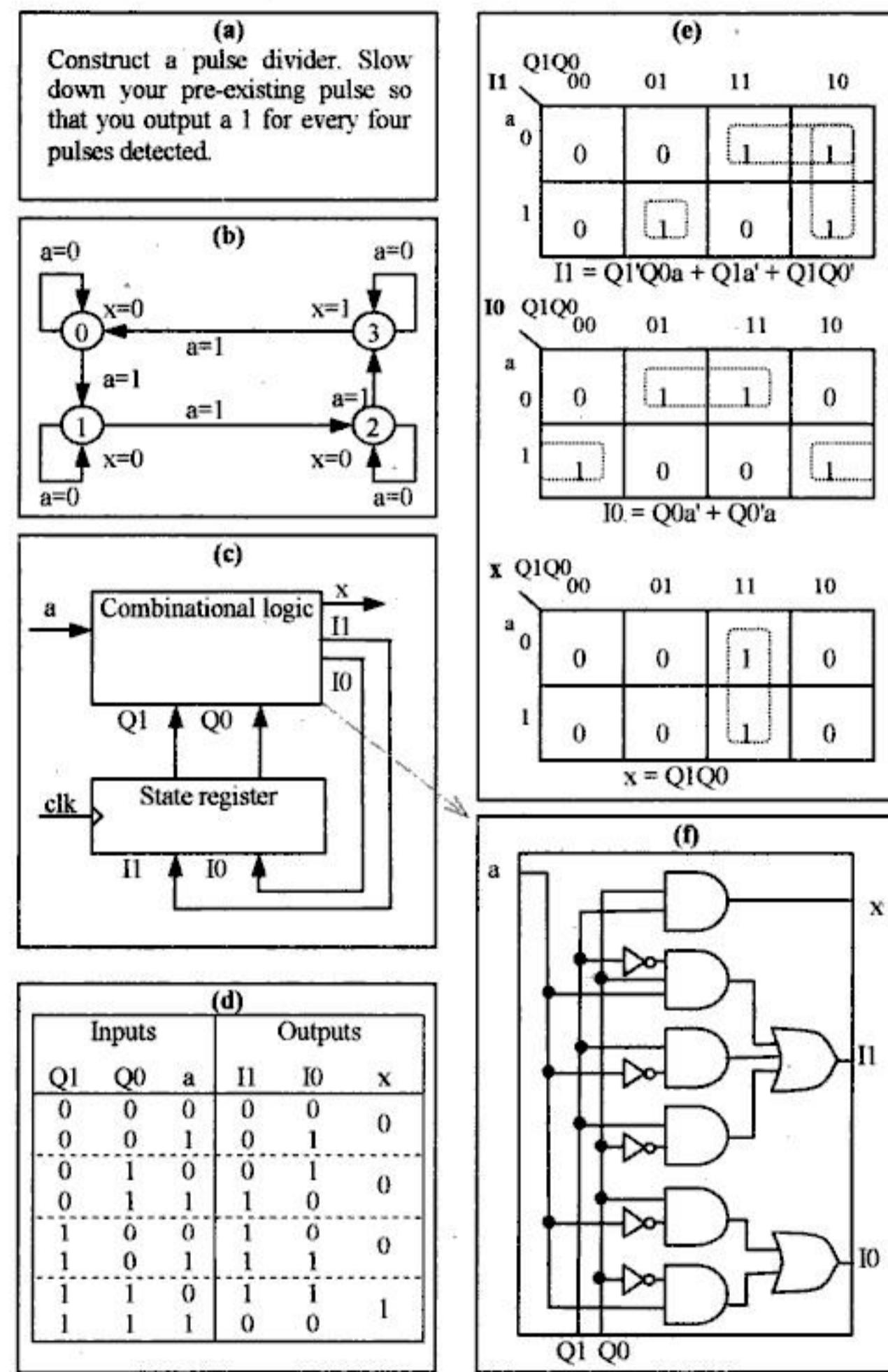


Figure 2.7: Sequential logic design: (a) problem description, (b) state diagram, (c) implementation model, (d) state table (Moore-type), (e) minimized output equations, (f) combinational logic.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

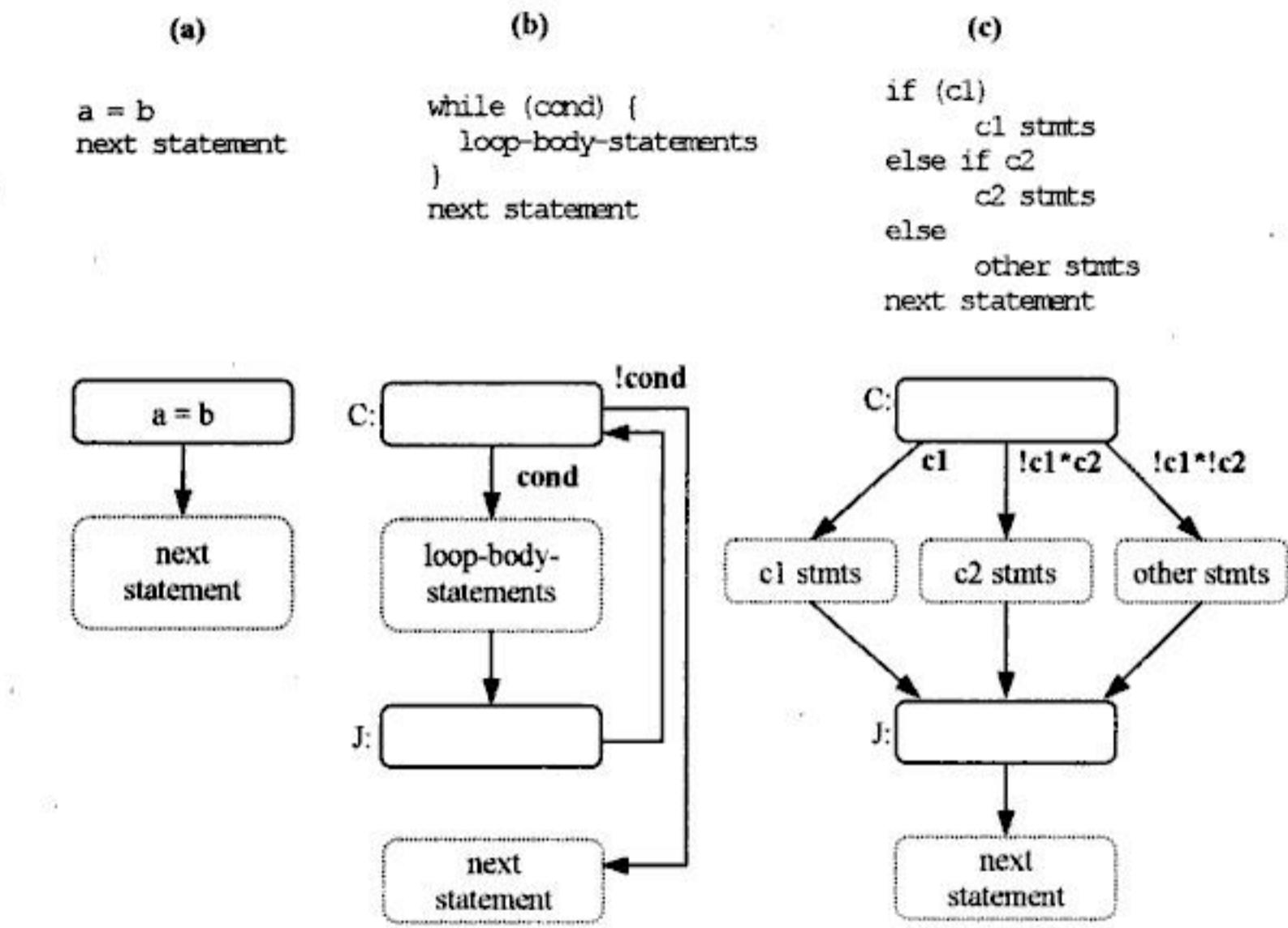


Figure 2.10: Templates for creating a state diagram from program statements: (a) assignment, (b) loop, (c) branch.

and a controller part, as shown in Figure 2.11. The datapath part should consist of an interconnection of combinational and sequential components. The controller part should consist of a pure FSM (i.e., one containing only Boolean actions and conditions.)

We construct the datapath through a four-step process:

1. First, we create a register for any declared variable. In the example, the variables are x and y . We treat an output port as an implicit variable, so we create a register d and connect it to the output port. We also draw the input and output ports. Figure 2.11(b) shows these three registers as light-gray rectangles.
2. Second, we create a functional unit for each arithmetic operation in the state diagram. In the example, there are two subtractions, one comparison for less than, and one comparison for inequality, yielding two subtractors and two comparators, shown as white rectangles in Figure 2.11(b).
3. Third, we connect the ports, registers, and functional units. For each write to a variable in the state diagram, we draw a connection from the write's source to the variable's register. A source may be an input port, a functional unit, or another register. For each arithmetic and logical operation, we connect sources to an input of the operation's corresponding functional unit. When more than one source is



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

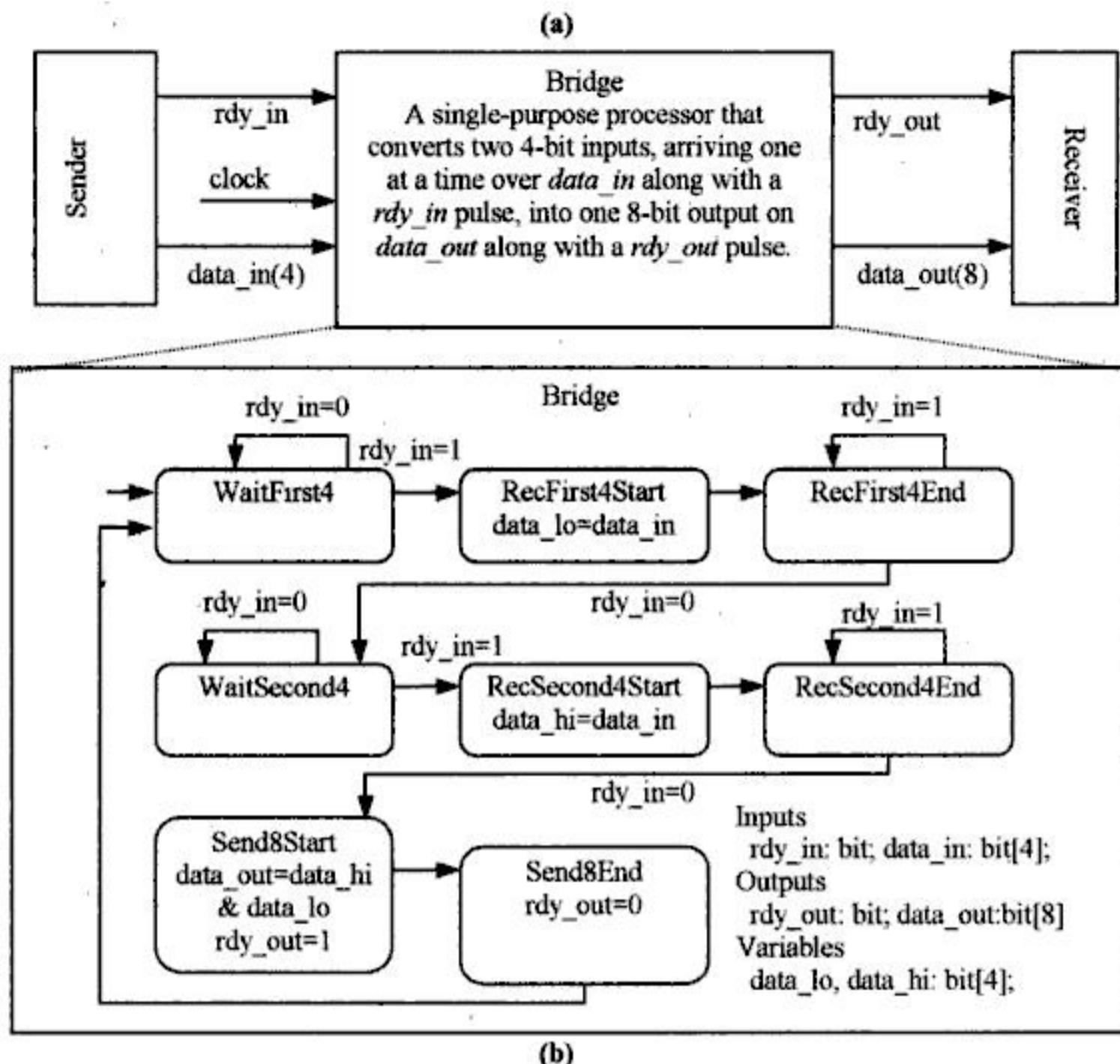


Figure 2.13: RT-level custom single-purpose processor design example: (a) problem specification, (b) FSMD.

Different designers might attack this problem at different levels of abstraction. One designer might start thinking in terms of registers, multiplexors, and flip-flops. Another might try to describe the bridge as a sequential program. But perhaps the most natural level is to describe the bridge as an FSMD, as shown in Figure 2.13(b). We begin by creating a state *WaitFirst4* that waits for the first 4 bits, whose presence on *data_in* will be indicated by a pulse on *rd़y_in*. Once the pulse is detected, we transition to a state *RecFirst4Start* that saves the contents of *data_in* in a variable called *data_lo*. We then wait for the pulse on *rd़y_in* to end, and then wait for the other 4 bits, indicated by a second pulse on *rd़y_in*. We save the contents of *data_in* in a variable called *data_hi*. After waiting for the second pulse on *rd़y_in* to end, we write the full 8 bits of data to the output *data_out*, and we pulse *rd़y_out*. We



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

- one clock cycle per instruction (advanced processors use parallelism to meet or exceed one cycle per instruction). (d) Compare the estimated gates with 200,000 gates, a typical number of gates for a modern 32-bit processor.
- 2.19 Design a single-purpose processor that outputs Fibonacci numbers up to n places. Start with a function computing the desired result, translate it into a state diagram, and sketch a probable datapath.
- 2.20 Design a circuit that does the matrix multiplication of matrices A and B . Matrix A is 3×2 and matrix B is 2×3 . The multiplication works as follows:

$$\begin{array}{c} A \\ \left[\begin{array}{cc} a & b \\ c & d \\ e & f \end{array} \right] \end{array} \bullet \begin{array}{c} B \\ \left[\begin{array}{ccc} g & h & i \\ j & k & l \end{array} \right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{ccc} a^*g + b^*j & a^*h + b^*k & a^*i + b^*l \\ c^*g + d^*j & c^*h + d^*k & c^*i + d^*l \\ e^*g + f^*j & e^*h + f^*k & e^*i + f^*l \end{array} \right] \end{array}$$

- 2.21 An algorithm for matrix multiplication, assuming that we have one adder and one multiplier, follows. (a) Convert the matrix multiplication algorithm into a state diagram using the template provided in Figure 2.10. (b) Rewrite the matrix multiplication algorithm given the assumption that we have three adders and six multipliers. (c) If each multiplication takes two cycles to compute and each addition takes one cycle compute, how many cycles does it take to complete the matrix multiplication given one adder and one multiplier? Three adders and six multipliers? Nine adders and 18 multipliers? (d) If each an adder requires 10 transistors to implement and each multiplier requires 100 transistors to implement, what is the total number of transistor needed to implement the matrix multiplication circuit using one adder and one multiplier? Three adders and six multipliers? Nine adders and 18 multipliers? (e) Plot your results from parts (c) and (d) into a graph with latency along the x -axis and size along the y -axis.

```

main() {
    int A[3][2] = { {1, 2}, {3, 4}, {5, 6} };
    int B[2][3] = { {7, 8, 9}, {10, 11, 12} };
    int C[3][3];
    int i, j, k;

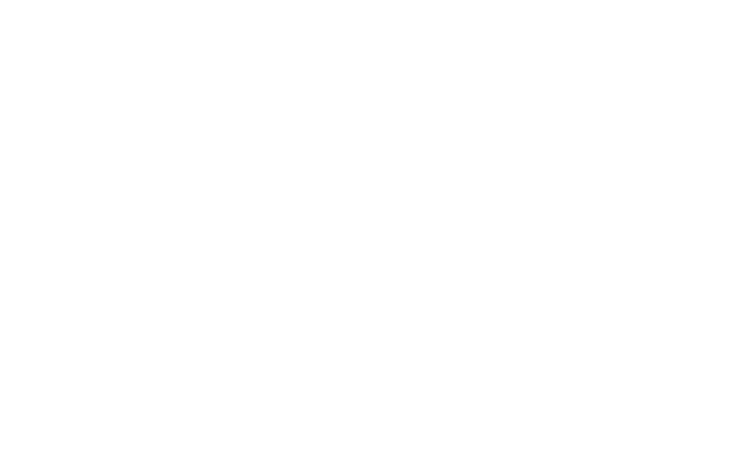
    for (i=0; i < 3; i++) {
        for ( j=0; j < 3; j++) {
            C[i][j]=0;
            for (k=0; k < 2; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}

```

- 2.22 A subway has an embedded system controlling the turnstile, which releases when two tokens are deposited. (a) Draw the FSMD state diagram for this system. (b) Separate the FSMD into an FSM+D. (c) Derive the FSM logic using truth tables and K-maps to minimize logic. (d) Draw your FSM and datapath connections.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



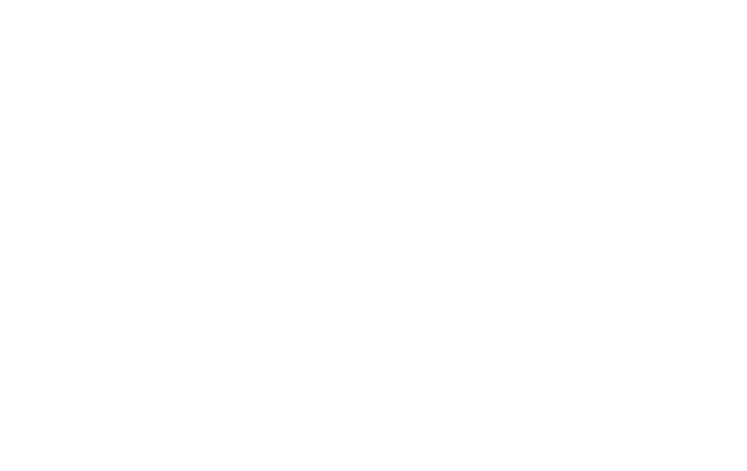
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



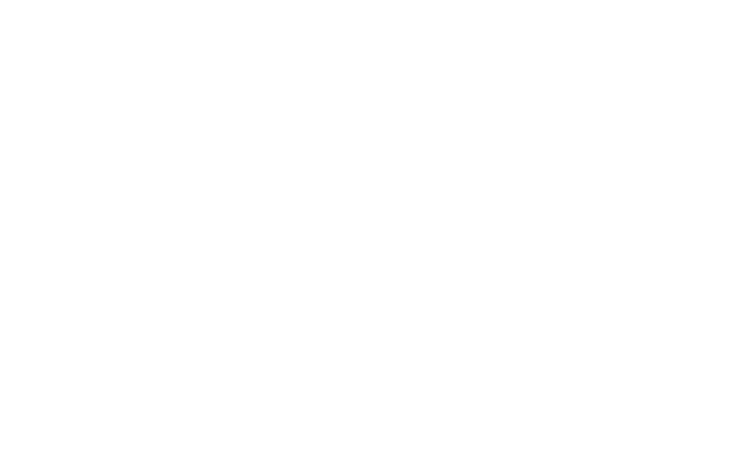
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



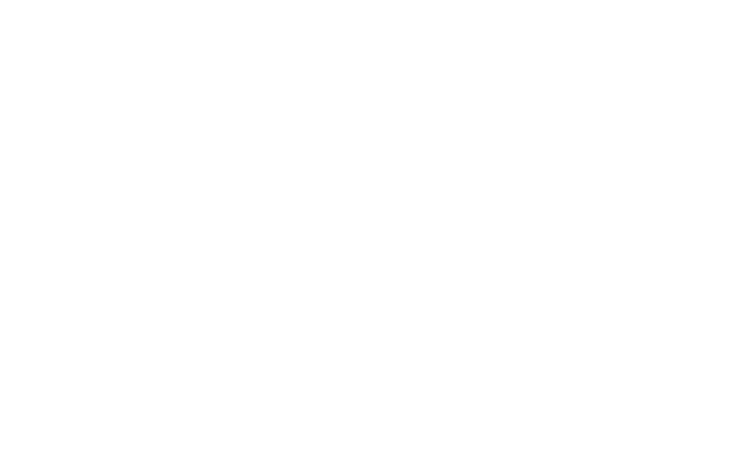
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



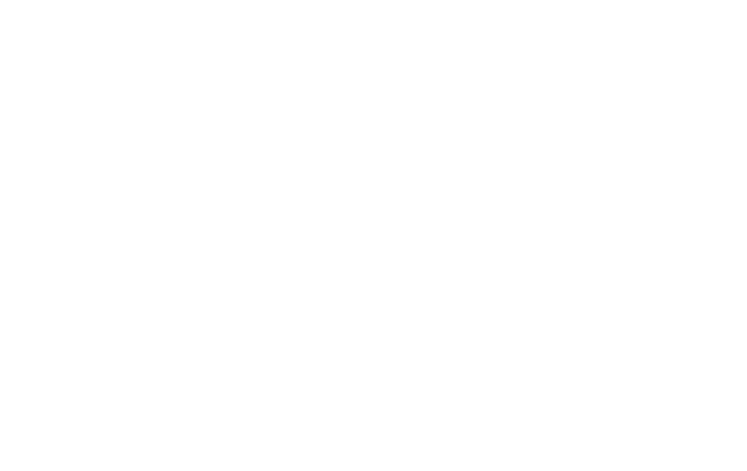
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



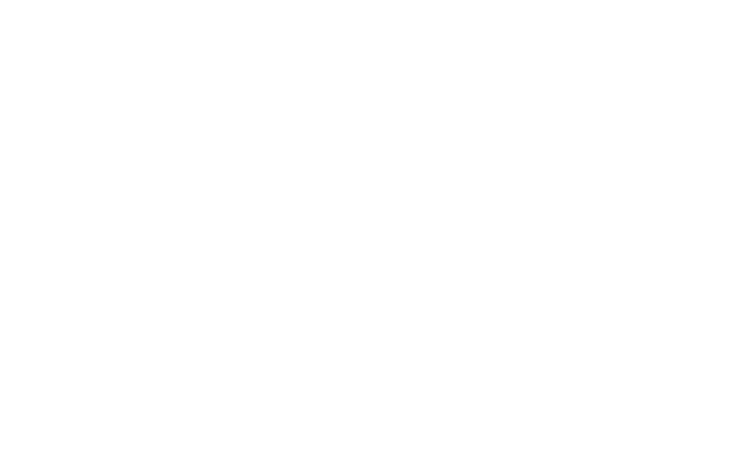
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

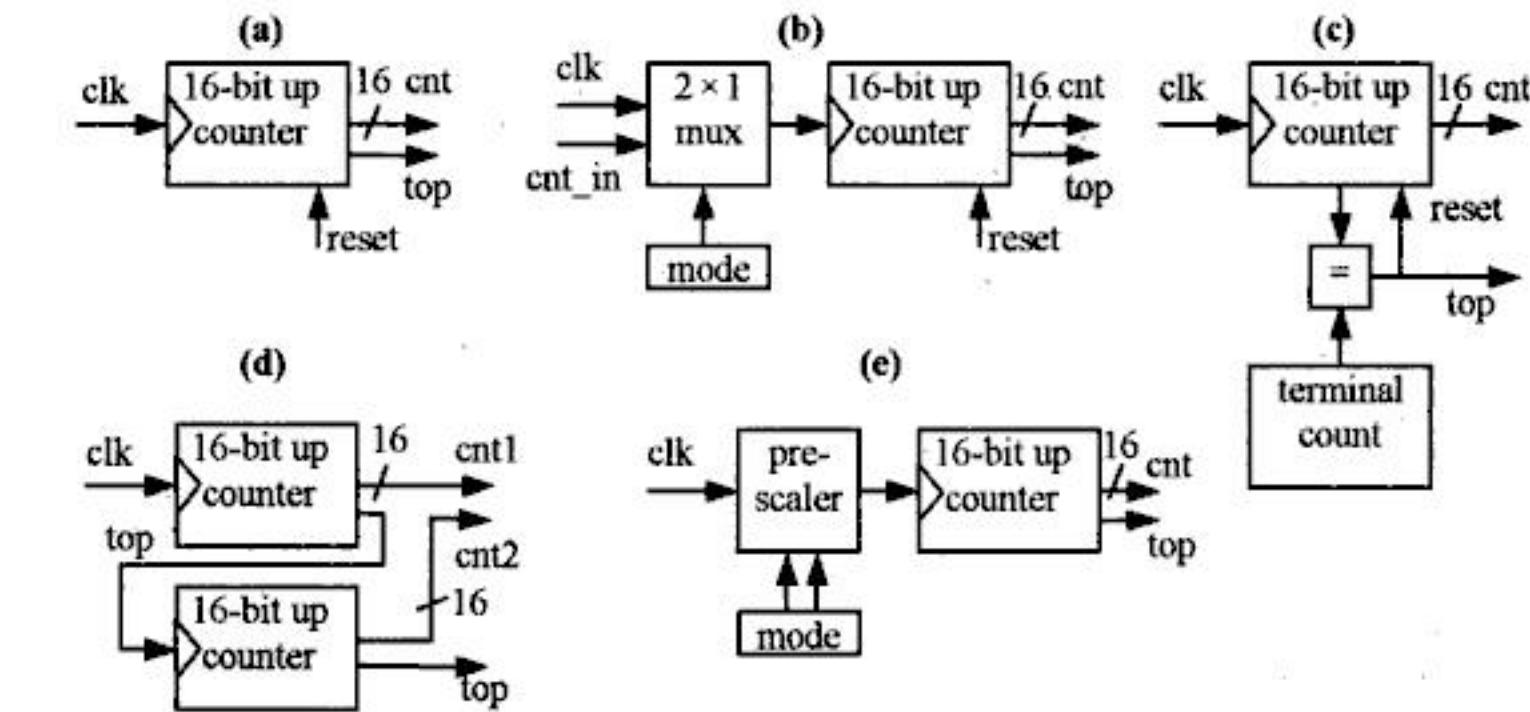


Figure 4.1: Timer structures: (a) a basic timer, (b) a timer/counter, (c) a timer with a terminal count, (d) a 16/32-bit timer, (e) a timer with a prescaler.

Figure 4.1(a) provides the structure of a very simple timer. This timer has an internal 16-bit up counter, which increments its value on each clock pulse. Thus, the output value *cnt* represents the number of pulses since the counter was last reset to zero. To interpret this number as a time interval, we must know the frequency or period of the clock signal *clk*. For example, suppose we wish to measure the time that passes between two button presses. In this case, we could reset the timer on the occurrence of the first press, and then read the timer output on the second press. Suppose the frequency of *clk* were 100 MHz, meaning the period would be $1 / (100 \text{ MHz}) = 10 \text{ nanoseconds}$, and that $cnt = 20,000$ at the time of the second button press. We would then compute the time that passed between the first and second button presses as $20,000 * 10 \text{ nanoseconds} = 200 \text{ microseconds}$. We note that since this timer's counter can count from 0 to 65,535, this particular timer has a measurement range of 0 to $65,535 * 10 \text{ nanoseconds} = 655.35 \text{ microseconds}$, with a resolution of 10 nanoseconds. We define a timer's *range* as the maximum time interval the timer can measure. A timer's *resolution* is the minimum interval it can measure.

The timer in Figure 4.1(a) has an additional output *top* that indicates when the top value of its range has been reached, also known as an overflow occurring, in which case the timer rolls over to 0. When we use a timer in conjunction with a general-purpose processor, and we expect time intervals to exceed the timer range, we typically connect the *top* signal to an interrupt pin on the processor. We create a corresponding interrupt service routine that counts the number of times the routine is called, thus effectively extending the range we can measure. Many microcontrollers that include built-in timers will have special interrupts just for its timers, with those interrupts distinct from external interrupts.

Figure 4.1(b) provides the structure of a more advanced timer that can also be configured as a counter. A *mode* register holds a bit, which the user sets, that uses a 2×1 multiplexor to



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



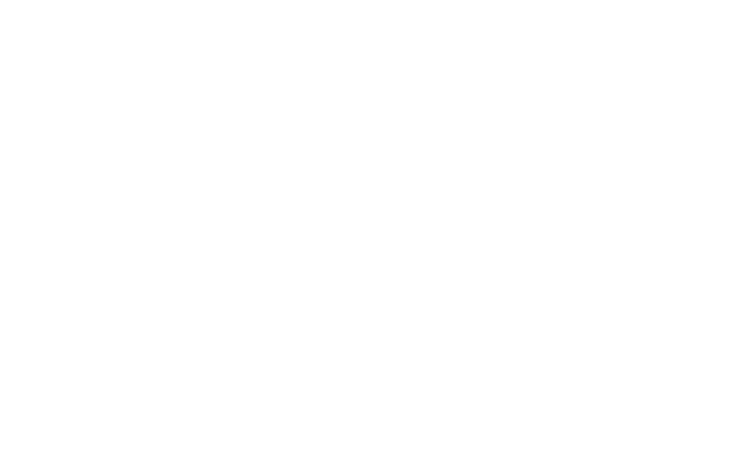
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



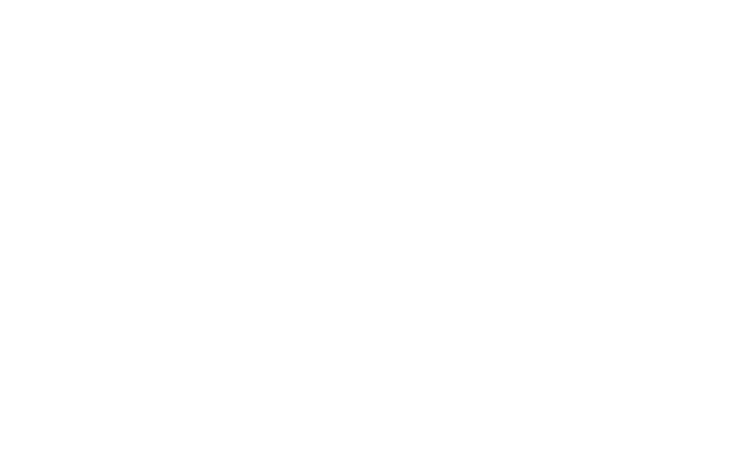
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



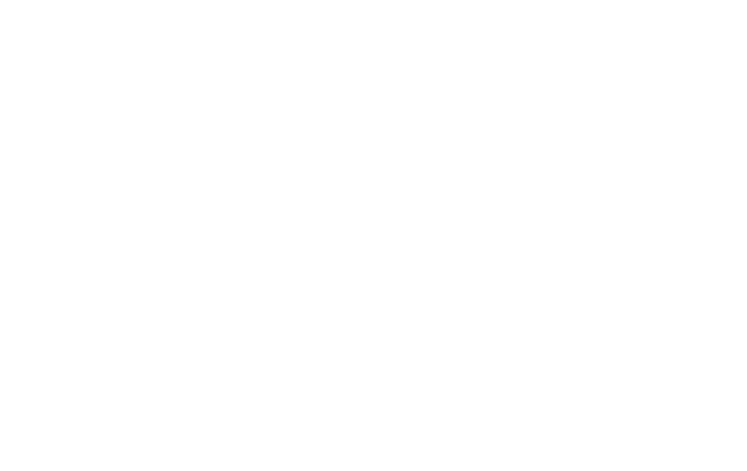
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



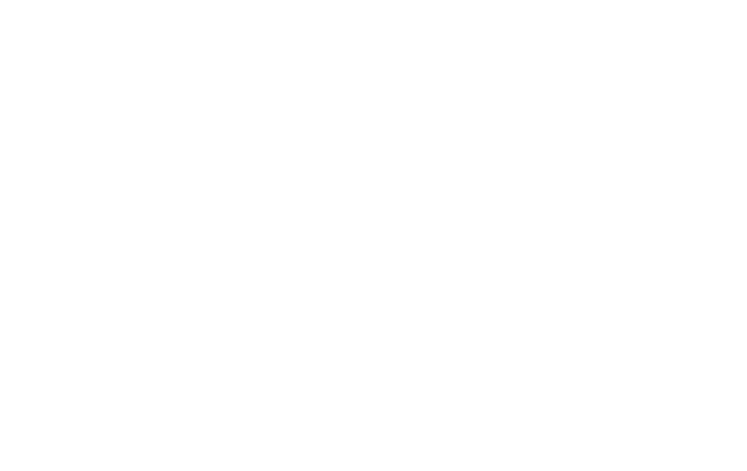
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



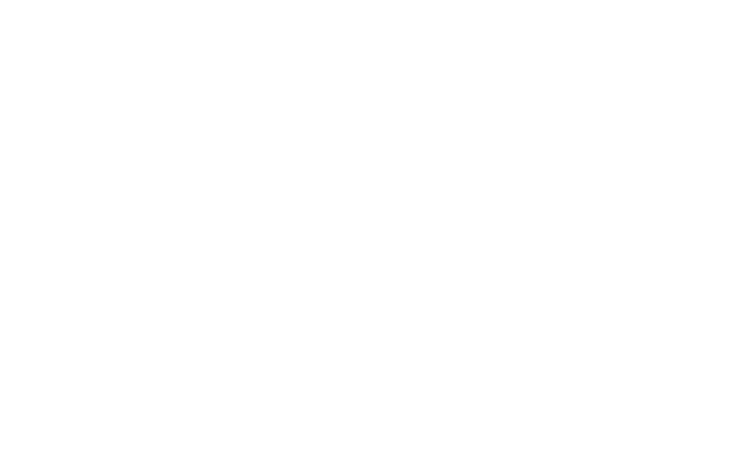
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

EPROMs can only be erased in their entirety. EEPROMs are typically more expensive than EPROMs, but far more convenient to use. EEPROMs are often called E²s, pronounced "E-squareds."

Because EEPROMs can be erased and programmed electronically, we can build the circuit providing the higher-than-normal voltage levels for such electronic erasing and programming right into the embedded system in which the EEPROM is being used. Thus, we can treat this as a memory that can be both read and written — a write to a particular word would consist of erasing that word followed by programming that word. Thus, an EEPROM is in-system programmable. We can use it to store data that an embedded system should save after power is shut off. For example, EEPROM is typically used in telephones that can store commonly dialed phone numbers in memory for speed-dialing. If you unplug the phone, thus shutting off power, and then plug it back in, the numbers will still be in memory. EEPROMs can typically hold data for 10 years and can be erased and programmed tens of thousands of times before losing their ability to store data.

In-system programming of EEPROMs has become so common that many EEPROMs come with a built-in memory controller. A *memory controller* hides internal memory-access details from the memory user, and provides a simple memory interface to the user. In this case, the memory controller would contain the circuitry and single-purpose processor necessary for erasing the word at the user-specified address, and then programming the user-specified data into that word.

While read accesses may require only tens of nanoseconds, writes may require tens of microseconds or more, because of the necessary erasing and programming. Thus, EEPROMs with built-in memory controllers will typically latch the address and data, so that the writing processor can move on to other tasks. Furthermore, such an EEPROM would have an extra "busy" pin to indicate to the processor that the EEPROM is busy writing, meaning that a processor wanting to write to the EEPROM must check the value of this busy pin before attempting to write. Some EEPROMs support read accesses even while the memory is busy writing.

A common use of EEPROM is to serve as the program memory for a microprocessor. In this case, we may want to ensure that the memory cannot be in-system programmed. Thus, EEPROM typically comes with a pin that can be used to disable programming.

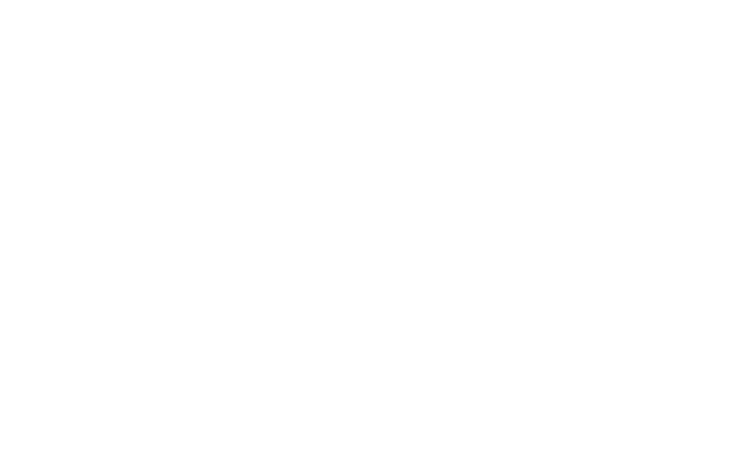
EEPROMs are more writable than EPROMs, as illustrated in Figure 5.2, since EEPROMs can be programmed in-system, and they are easier to erase. EEPROM is where the distinction between ROM and RAM begins to blur, since EEPROMs are in-system programmable and thus writable directly by a processor. Thus, the term "read-only-memory" for EEPROM is really a misnomer, since the processor can in fact write to an EEPROM. Such writes are slow compared to reads and are limited in number, but nevertheless, EEPROMs can and are commonly written by a processor during normal system operation.

Flash Memory

Flash memory is an extension of EEPROM that was developed in the late 1980s. While also using the floating-gate principle of EEPROM, flash memory is designed such that large blocks of memory can be erased all at once, rather than just one word at a time as in



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

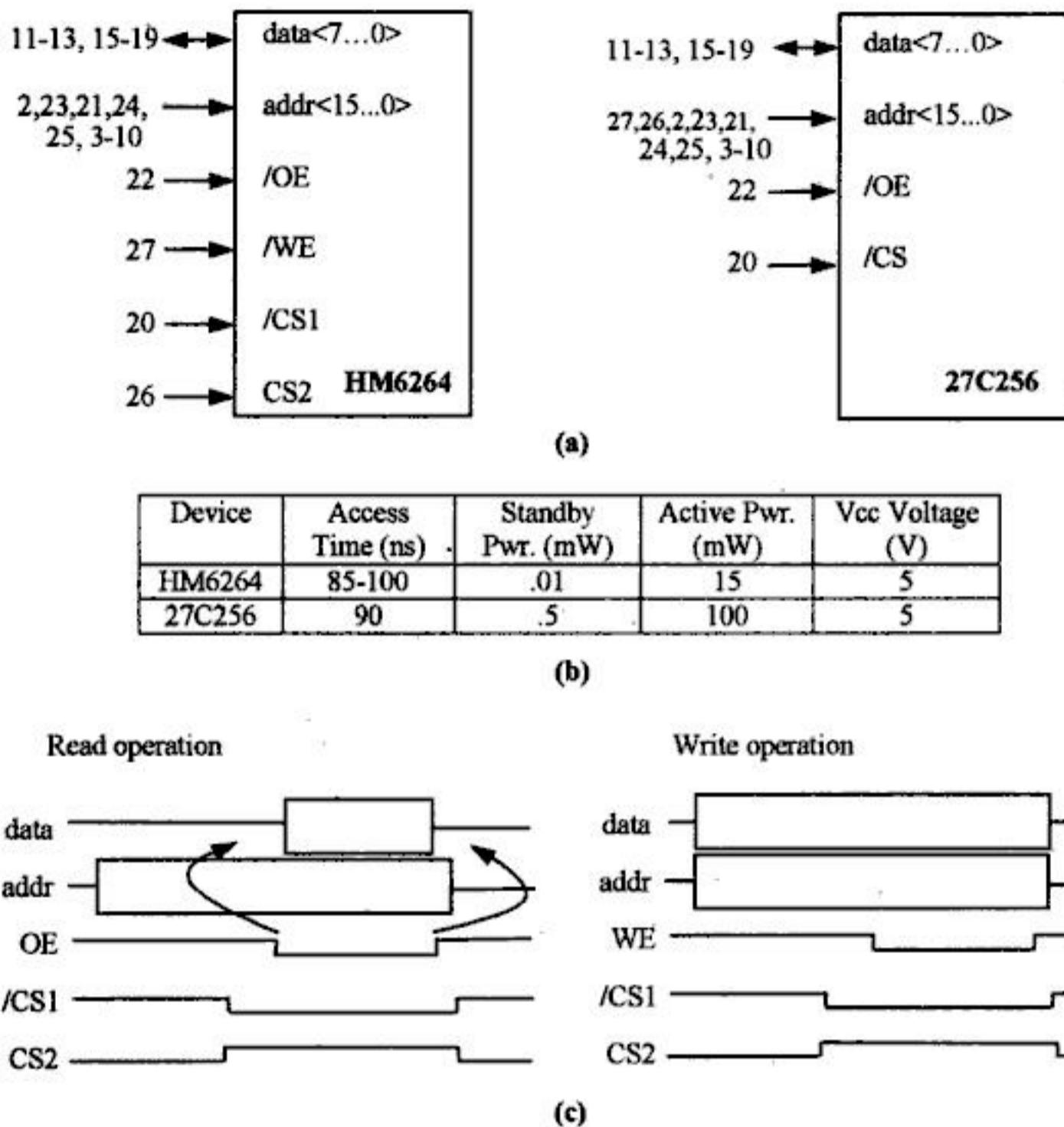


Figure 5.8: HM6264 and 27C256 RAM/ROM devices: (a) block diagram, (b) characteristics, (c) timing diagrams.

Subsequent digits give the memory capacity in kilobits. Both these devices are available in 4, 8, 16, 32, and 64 kilobytes, so the part numbers 62 or 27 would be followed by the number of kilobits, which may be 32, 64, 128, 256, or 512. Figure 5.8(b) summarizes some of the characteristics of these devices.

Memory access to and from these devices is performed through an 8-bit parallel protocol. Placing a memory address on the address-bus and asserting the read signal output enable (*OE*) performs a read operation. Placing some data and a memory address on the data and address busses and asserting the write signal write enable (*WE*) performs a write operation. The read and write timing is given in Figure 5.8(c).



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



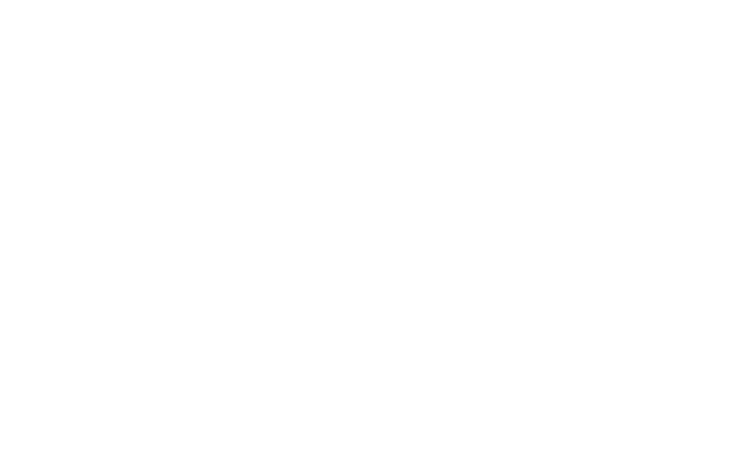
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



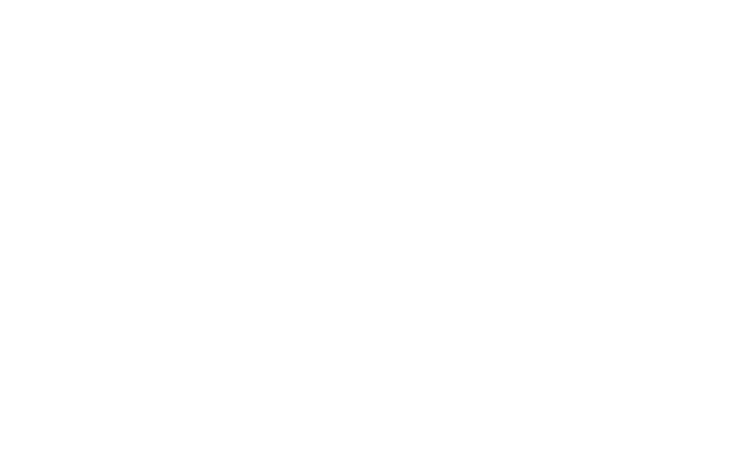
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



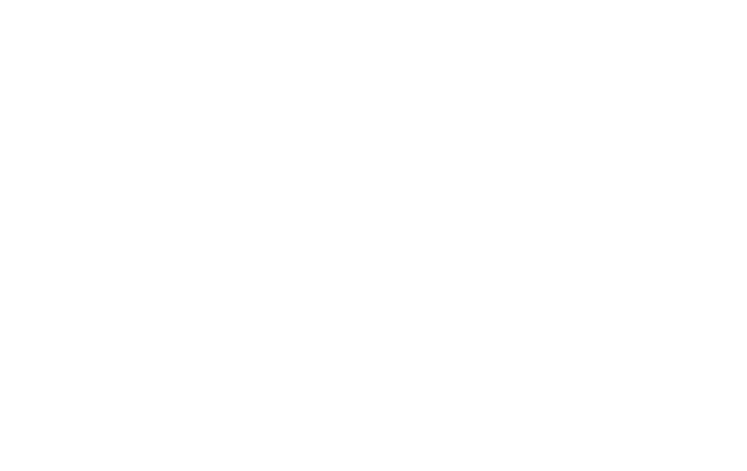
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



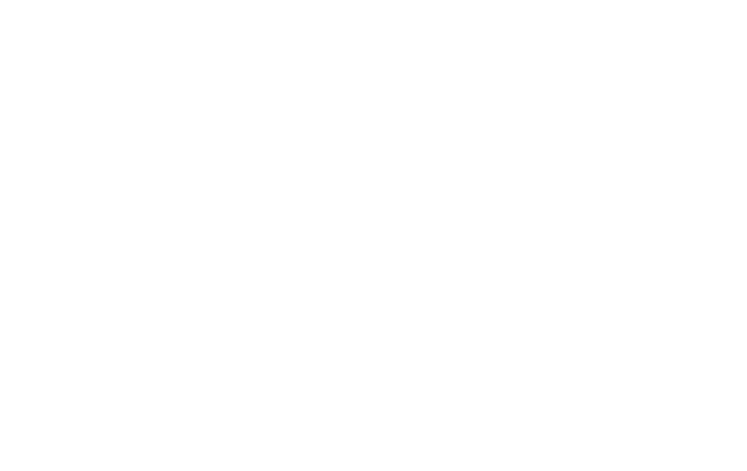
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



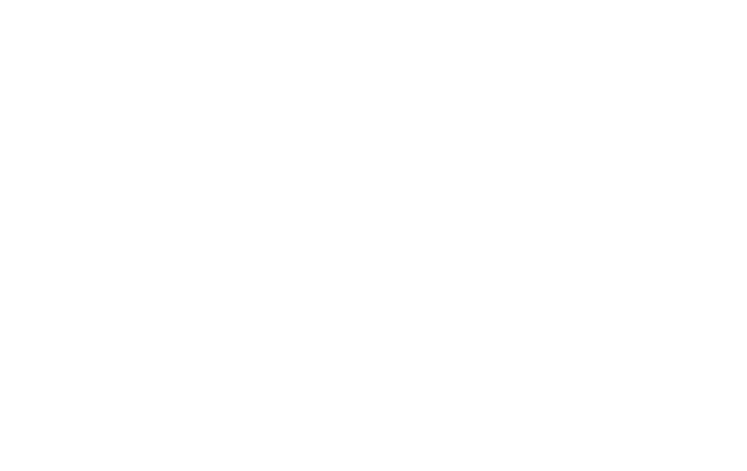
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



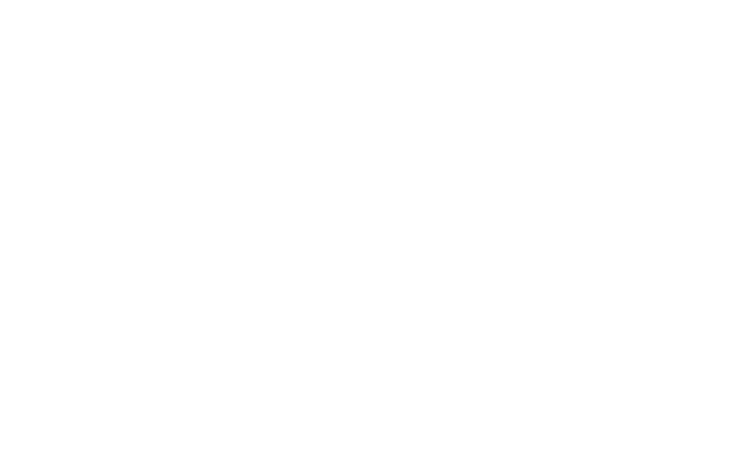
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



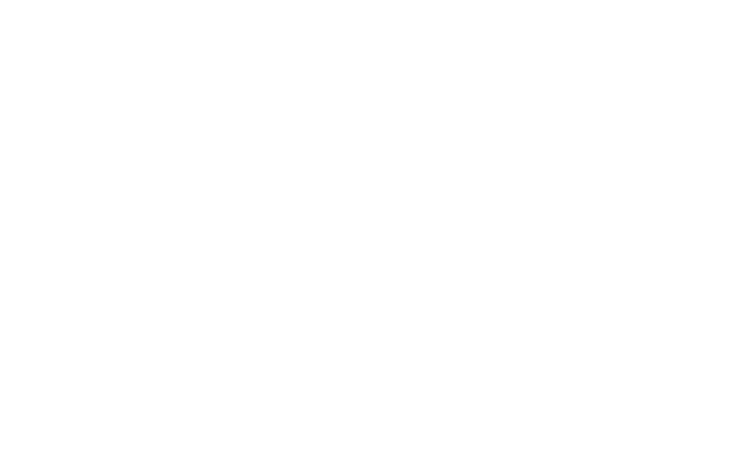
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



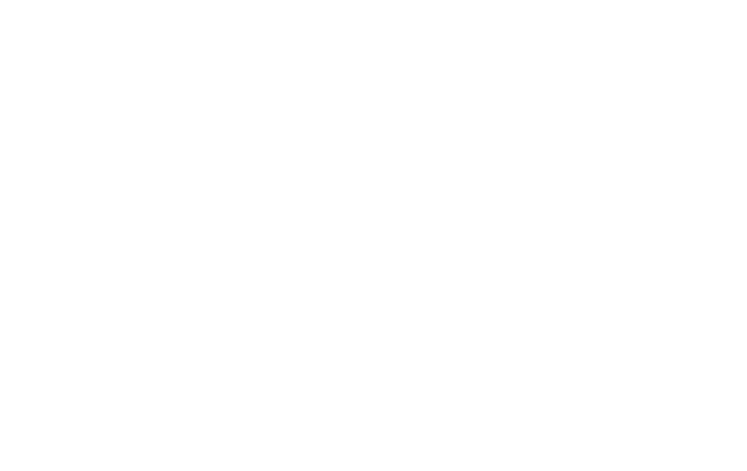
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



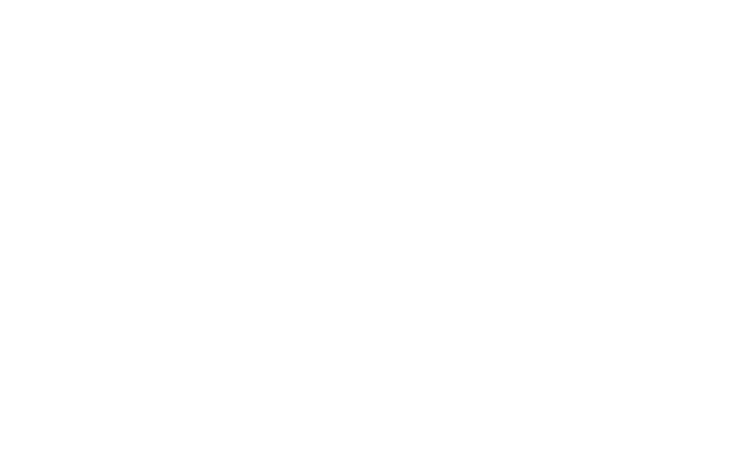
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



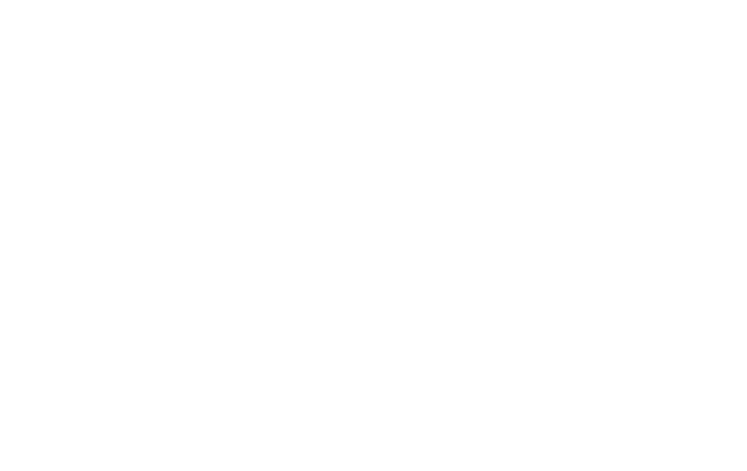
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

- | | |
|---------------------------|-------------------------------------|
| 11. Portable MP3 player | 29. TV-based Web access box |
| 12. Digital camera | 30. House temperature control |
| 13. Electronic book | 31. Home alarm system |
| 14. Trash compactor | 32. Point-of-sale system |
| 15. Hearing aid | 33. Video-game console |
| 16. Dishwasher | 34. TV remote control |
| 17. Electronic clock | 35. Electronic keyboard/synthesizer |
| 18. Video camera | 36. Fax machine |
| 19. Electronic wristwatch | 37. Scanner |
| 20. Pager | 38. Wireless networking |
| 21. Cell phone | 39. Telephone modem |
| 22. CD player | 40. Cable modem |
| 23. DVD player | 41. Printer |
| 24. Smart speakers | 42. Portable video game |
| 25. Stereo receiver | 43. Personal digital assistant |
| 26. TV set-top box | 44. Portable digital picture viewer |
| 27. Television | 45. Phone with answering machine |
| 28. VCR | |

Index

A

accelerator, 11
actuator, 247
adder, [33](#)
address space, 57
aliasing, 264
allocation, 50, 294
ALU, 34, 56
analog-to-digital converter, 102
application-specific IC. *See* ASIC
application-specific instruction-set processor. *See* ASIP
arbitration, 160
ASIC, [13](#), 276
ASIP, 12, 74
assembler, 71
assembly-language programming, 61
asynchronous, 36

B

battery-backed RAM, 120
baud rate, 91
behavioral description, 285
behavioral synthesis, 294
benchmark, 76
binding, 50
bit, 57
Bluetooth, 174
bridge, 165
buffering, 153
bus, 138

bus-based I/O, 145
busy-waiting, 232
byte, 57

C

cache, 125
cache block, 59, 126
cache hit, 59, 126
cache line, 126
cache memory, 59
cache miss, 59, 126
cache replacement policy, 128
CAD, 43
CAN bus, 171
CCD, 180
cell-based array, 277
checksum, 169
clock cycle, 57
closed-loop control, 248
CMOS transistor, 30, 270
codesign, [21](#), 295
combinational logic, [33](#)
communicating process model, 208
comparator, 34
compiler, 71
computation model, 209
concurrent process model, 223
condition variable, 233
consumer-producer problem, 228
control system, 245
control unit, 57
control-dominated system, 208

controllability, 297

controller, 39, 42

coprocessor, 11

core, 302

correctness, 5

co-simulation, 18

counter, 36, 84

CPLD, 278

CPU, 56

critical path, 57

critical section, 229

cross compiler, 71

cruise-control system, 248

D

daisy-chain arbitration, 160

data-dominated system, 208

dataflow model, 208, 241

datapath, 9, 38, 41, 56

DC motor, 94

D-cache, 59

DCT, 182, 200

deadline, 240

deadline monotonic priority assignment, 240

deadlock, 231

debugger, 73

decoder, 33

design metric, 4

design process model, 304

design productivity gap, 22

design technology, 16

development processor, 69

device programmer, 73

Dhrystone benchmark, 76

Dhrystone MIPS, 77

digital camera, 4, 179, 315

digital signal processor, 12, 75

digital-to-analog converter, 102

direct addressing, 63

direct memory access. *See* DMA

direct-mapped cache, 126

DMA, 154

DRAM, 120, 130, 134

driver, 64, 66

DSP. *See* digital signal processor

duty cycle, 92

dynamic RAM. *See* DRAM

E

EEMBC, 77

EEPROM, 116

embedded processors, 74

embedded system, 1

emulator, 73, 301

EPROM, 115

ESDRAM, 133

extended data out DRAM, 132

extended FSM, 213

extended parallel I/O, 145

F

fast page mode DRAM, 131

feature size, 13

field programmable gate array. *See* FPGA

FIFO scheduler, 239

finite-state machine. *See* FSM

finite-state machine with data. *See* FSMD

FireWire bus, 172

fixed interrupt, 149

fixed priority arbitration, 160

fixed-point arithmetic, 201

flash memory, 117

flexibility, 5

flip-flop, 35

formal verification, 296

FPGA, 14, 279, 312

frameworks, 19

FSM, 36, 211

FSMD, 39, 44, 47, 48, 213

full-custom IC, 13

fully-associative cache, 126

G

gate array, 13, 276

gates, 30

- GCD, 48
 general-purpose processor, [9](#), [29](#), 55, 77
 greatest common divisor, 39
- H**
 handshake protocol, 141
 hardware/software codesign. *See* codesign
 hardware-software co-simulator, 300
 Harvard architecture, 58
 HCFSM, 217
 high-level synthesis, 294
 Huffman encoding, 183
- I**
 I²C bus, 169, 315
 IC, [13](#), [21](#)
 IC technology, [13](#)
 I-cache, 59
 IEEE 802.11, 174
 immediate addressing, 63
 implicant, 287
 implicit addressing, 63
 indexed addressing, 63
 indirect addressing, 63
 Industry Standard Architecture. *See* ISA bus
 infrared, 167
 inherent addressing, 63
 instruction-set simulator, 71, 300
 in-system programmable memory, 112
 integrated circuit. *See* IC
 integrated development environment (IDE), 70
 Intel 8051, 147, 195, 312
 intellectual property. *See* IP
 interrupt, 65, 149
 interrupt address table, 150
 interrupt address vector, 149
 interrupt controller, 160
 interrupt service routine. *See* ISR
 interrupt-driven I/O, 149
 IP, 18, 302
 IrDA, 174
- ISA bus, 143, 147, 158, 315
 ISR, 65, 68, 149, 152, 163
- J**
 JPEG, 4, 181
- K**
 Karnaugh map, [33](#), 288
 keypad, 97
- L**
 languages, 19
 latency, [8](#)
 layout, 272
 LCD, 95
 linker, 71
 liquid crystal display. *See* LCD
 logic gates, 30
 logic synthesis, 287
- M**
 machine-language programming, 61
 maintainability, 5
 market window, [6](#)
 mask, 67
 maskable interrupt, 151
 mask-programmed ROM, 114
 Mealy-type FSM, 212
 memory hierarchy, 125
 memory programmer, 111
 memory-mapped I/O, 145
 message passing, 231
 microcontroller, 12, 74, 312
 microprocessor. *See* general-purpose processor
 minterm, 287
 MIPS, 77
 MMU, 134
 monitor, 235
 Moore's Law, 15, 110
 Moore-type FSM, 212
 multilevel logic minimization, 290
 multiplexor, [33](#)



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

This Wiley Student Edition is part of a continuing program of paperbound textbooks especially designed for students in developing countries at a reduced price.

THIS BOOK IS FOR SALE ONLY IN THE COUNTRY TO WHICH IT IS FIRST CONSIGNED BY WILEY INDIA PVT. LTD. AND MAY NOT BE RE-EXPORTED.

FOR SALE ONLY IN :

INDIA, BANGLADESH, NEPAL, PAKISTAN, SRI LANKA & BHUTAN



Wiley India Pvt. Ltd.
4435/7, Ansari Road, Daryaganj,
New Delhi-110 002.
Tel: 91-11-43630000
Fax: 91-11-23275895
E-mail: csupport@wileyindia.com
Website: www.wileyindia.com

ISBN 978-81-265-0837-2



9 788126 508372