WILEY-
INDIA
EDITION

EMBEDDED SYSTEM DESIGN

A Unified Hardware / Software Introduction

Frank Vahid / Tony Givargis

EMBEDDED SYSTEM DESIGN : A UNIFIED HARDWARE / SOFTWARE INTRODUCTION
Third Edition

Reprint : 2009

Printed at : Print India Press, Delhi

ISBN : 978-81-265-0837-2

Contents

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 1: *Introduction*

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

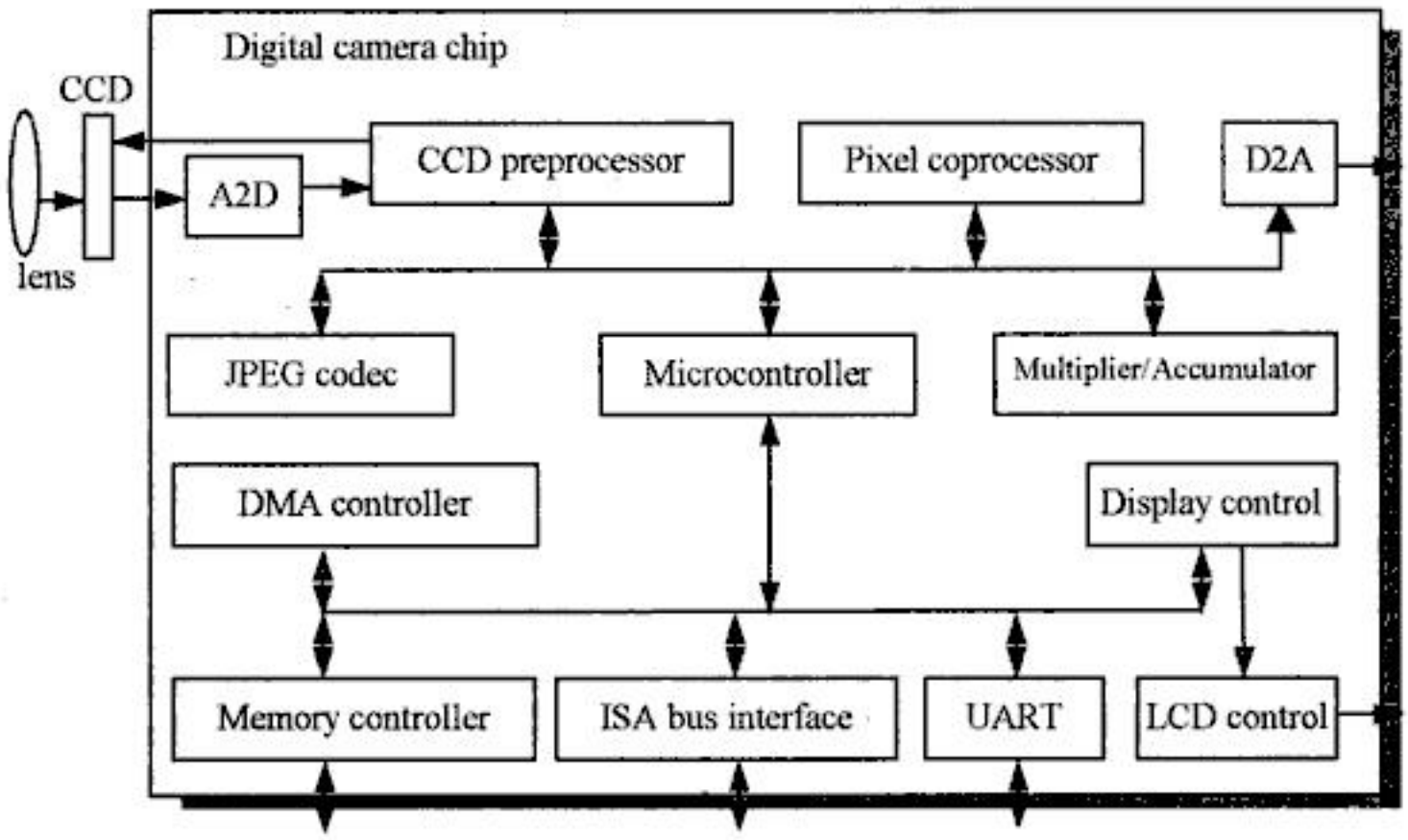


Figure 1.2: An embedded system example — a digital camera.

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.

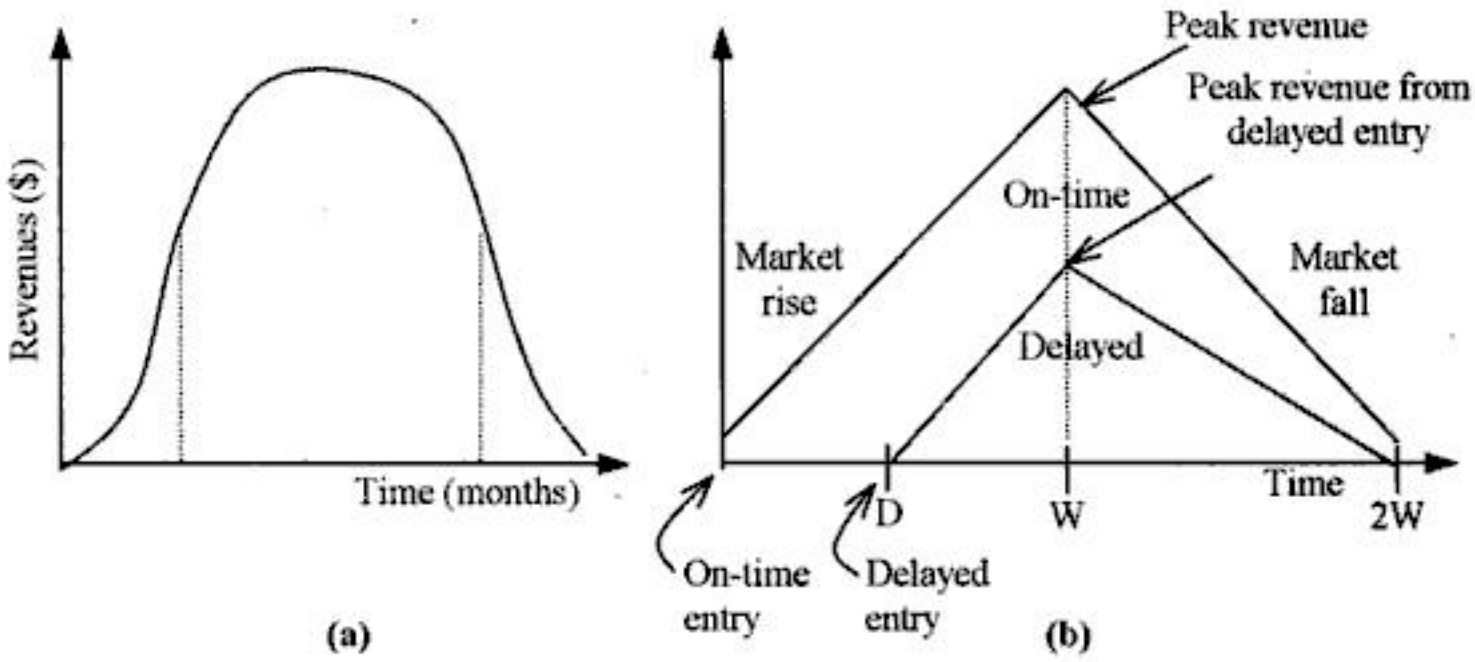


Figure 1.4: Time-to-market: (a) market window, (b) simplified revenue model for computing revenue loss from delayed entry.

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

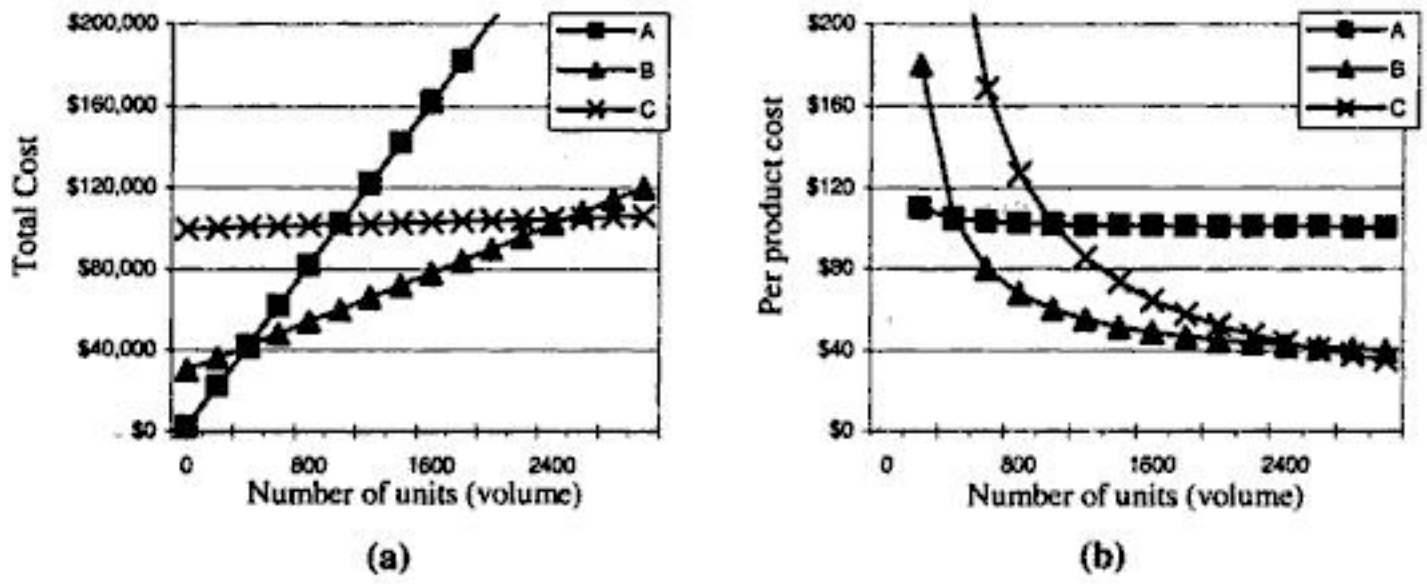


Figure 1.5: Costs for technologies A, B, and C as a function of volume: (a) total cost, (b) per-product cost.

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 ((*On-time* – *Delayed*) / *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 ½ * *base* * *height*, is thus ½ * $2W$ * W, or W^2. The area of the *Delayed* triangle is ½$(W - D + W)$ * $(W - D)$. 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:

speedup of A over B = performance of A / 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.

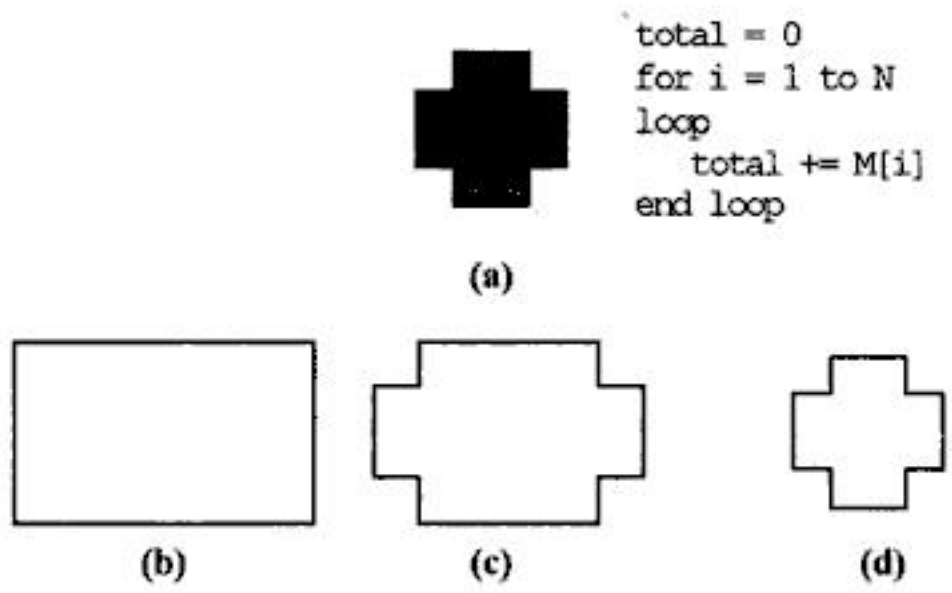


Figure 1.6: Processors vary in their customization for the problem at hand: (a) desired functionality, (b) general-purpose processor, (c) application-specific processor, (d) single-purpose processor.

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.

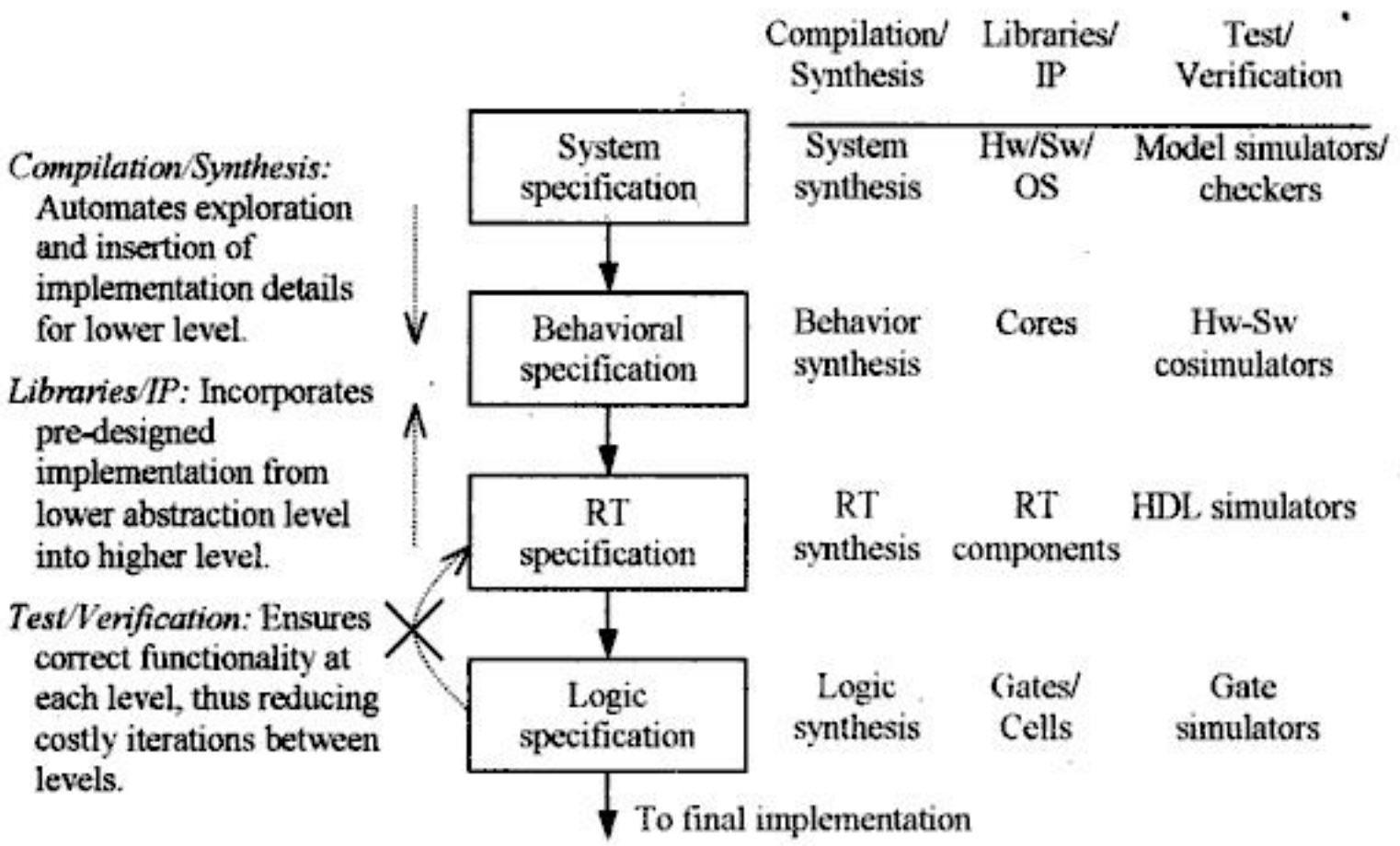


Figure 1.11: Ideal top-down design process, and productivity improvers.

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.

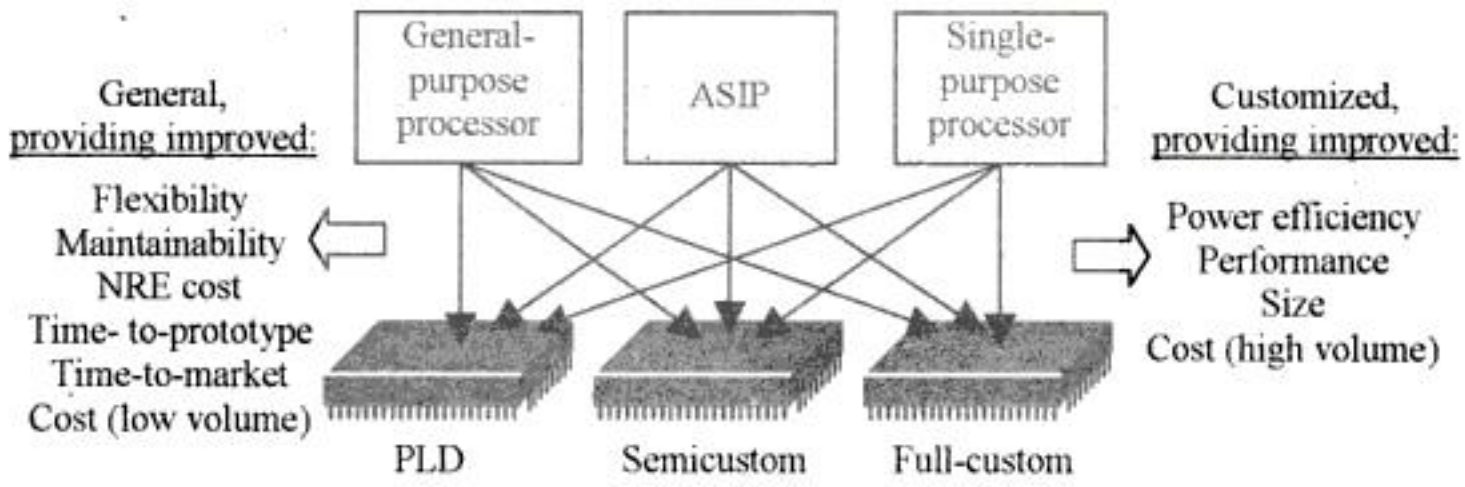


Figure 1.14: The independence of processor and IC technologies: Any processor technology can be mapped to any IC technology.

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.
- Debardelaben, J., Madisetti, V.K., and Gadient, 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

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 *In* 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.

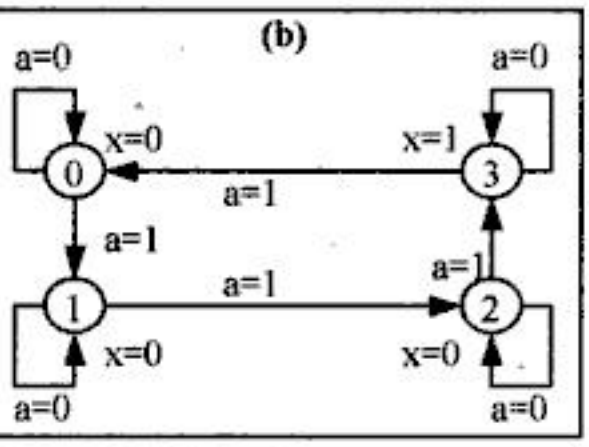
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.

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

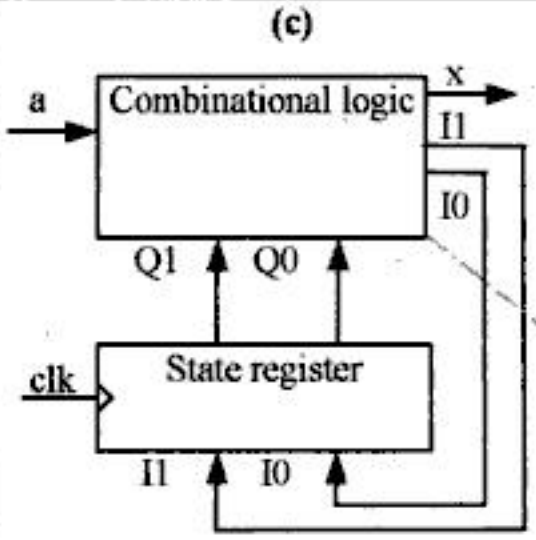
(a)

Construct a pulse divider. Slow down your pre-existing pulse so that you output a 1 for every four pulses detected.

(b)



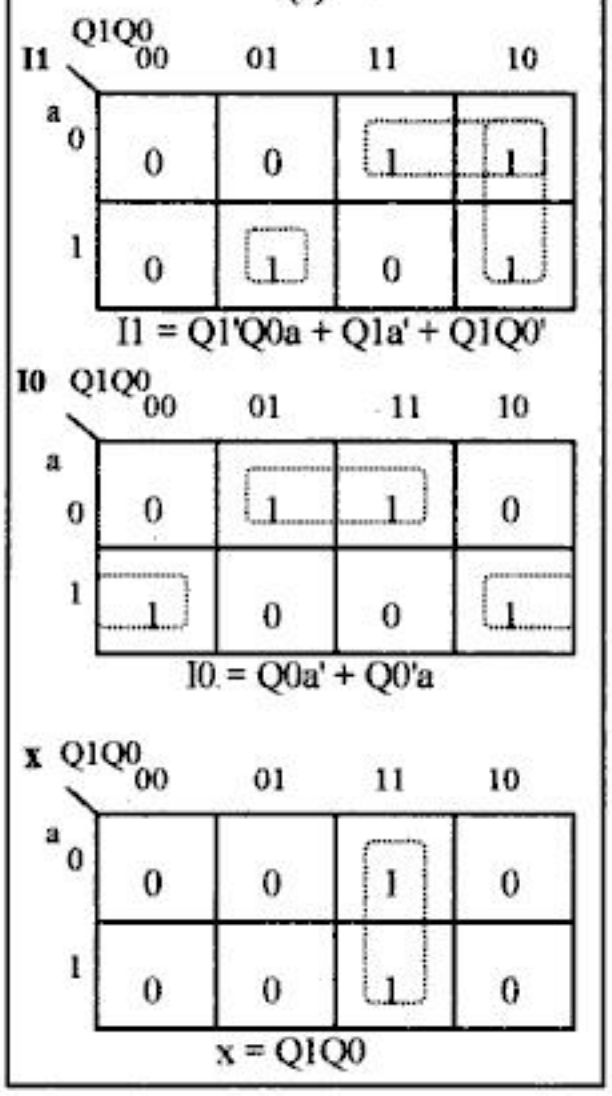
(c)



(d)

Inputs			Outputs		
Q1	Q0	a	I1	I0	x
0	0	0	0	0	0
0	0	1	0	1	
0	1	0	0	1	0
0	1	1	1	0	
1	0	0	1	0	0
1	0	1	1	1	
1	1	0	1	1	1
1	1	1	0	0	

(e)



(f)

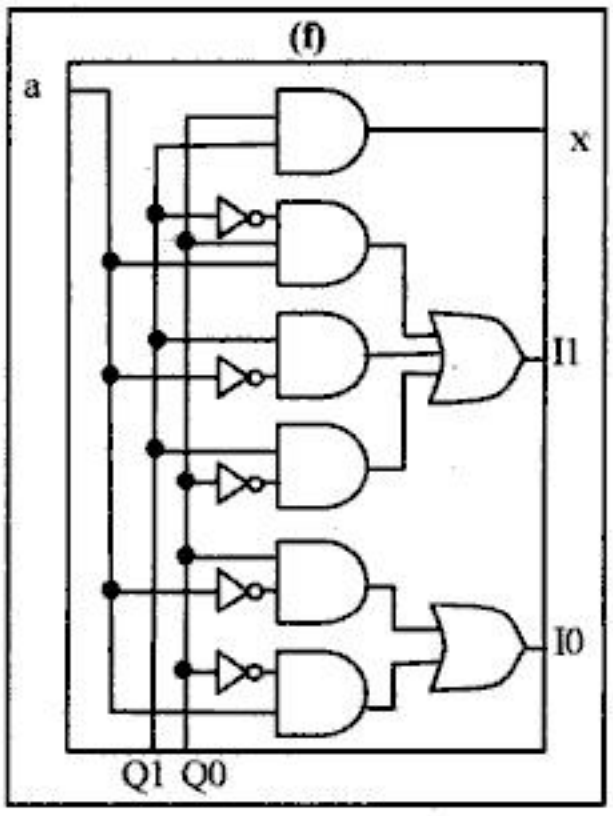


Figure 2.7: Sequential logic design: (a) problem description, (b) state diagram, (c) implementation model, (d) state table (Moore-type), (e) minimized output equations, (f) combinational logic.

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.

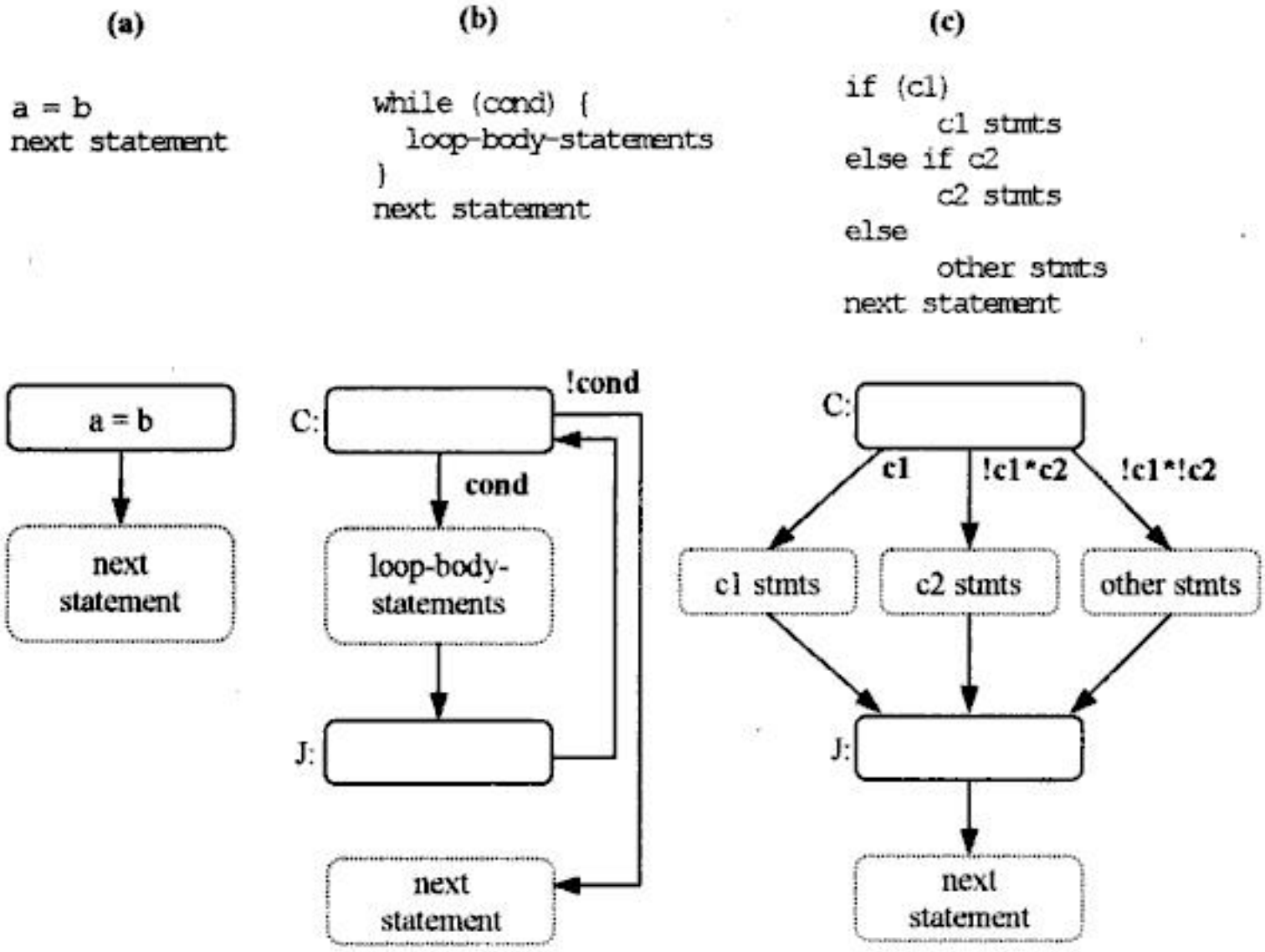


Figure 2.10: Templates for creating a state diagram from program statements: (a) assignment, (b) loop, (c) branch.

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.

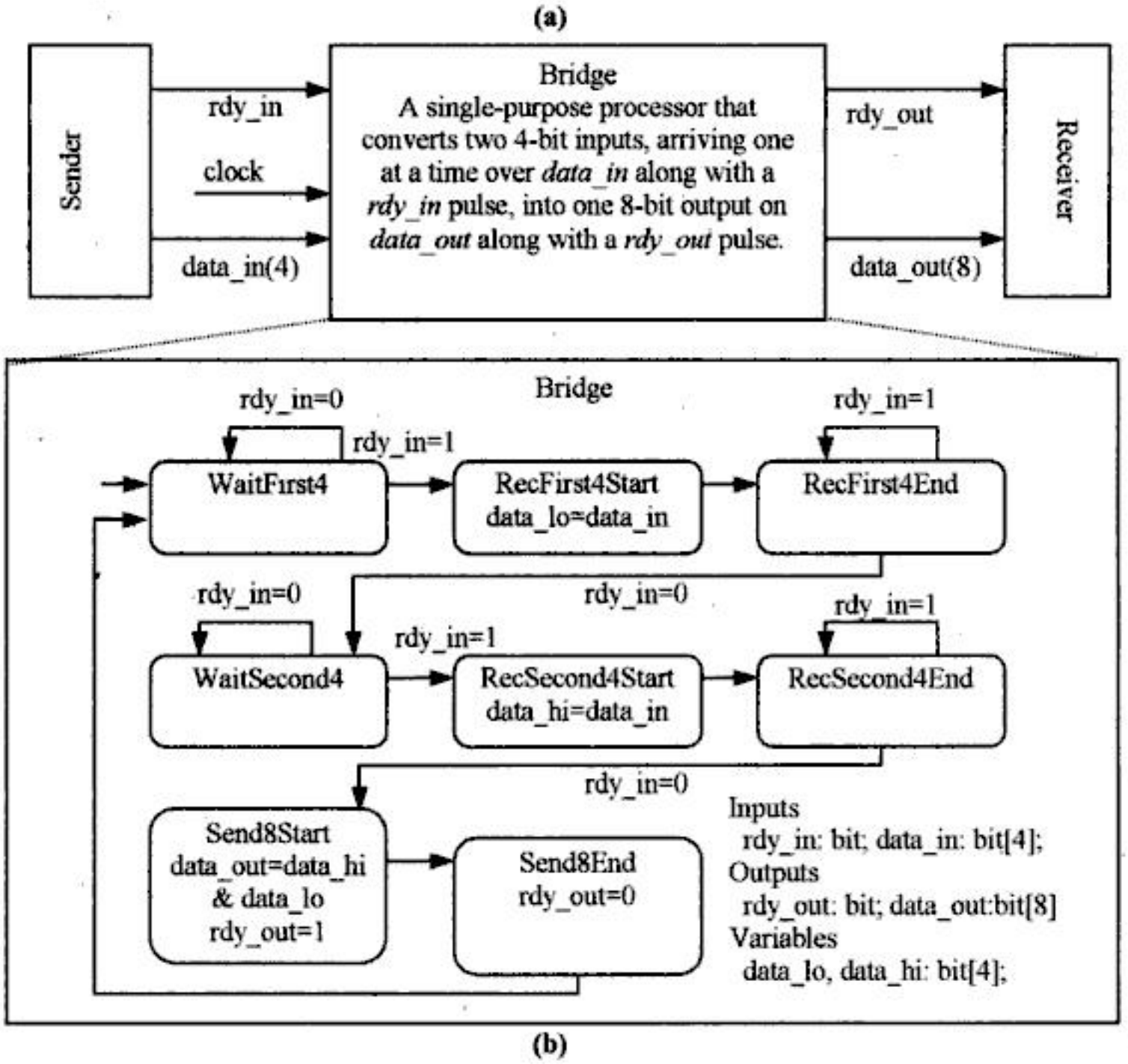


Figure 2.13: RT-level custom single-purpose processor design example: (a) problem specification, (b) FSMD.

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 *rdy_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 *rdy_in* to end, and then wait for the other 4 bits, indicated by a second pulse on *rdy_in*. We save the contents of *data_in* in a variable called *data_hi*. After waiting for the second pulse on *rdy_in* to end, we write the full 8 bits of data to the output *data_out*, and we pulse *rdy_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:

$$\underset{A}{\begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix}} \bullet \underset{B}{\begin{bmatrix} g & h & i \\ j & k & l \end{bmatrix}} = \underset{C}{\begin{bmatrix} a*g + b*j & a*h + b*k & a*i + b*l \\ c*g + d*j & c*h + d*k & c*i + d*l \\ e*g + f*j & e*h + f*k & e*i + f*l \end{bmatrix}}$$

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.

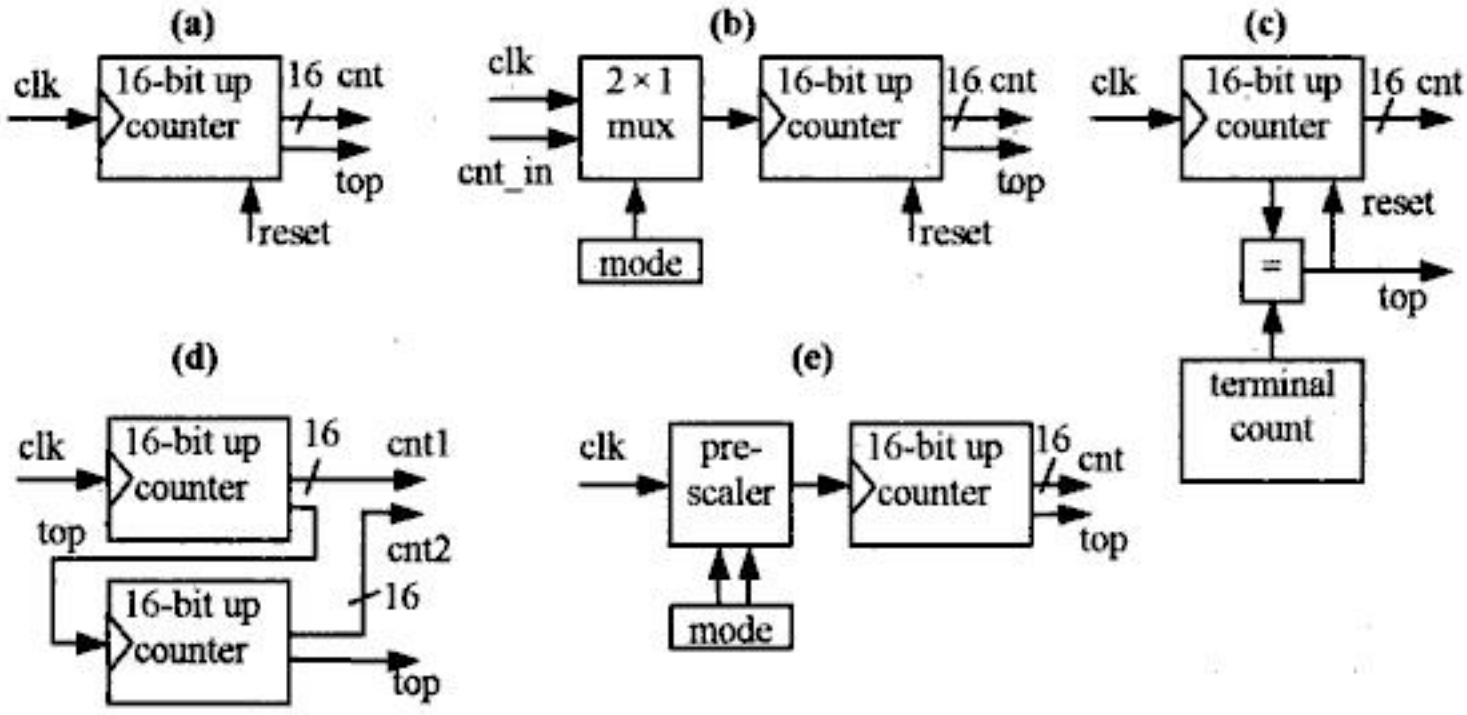


Figure 4.1: Timer structures: (a) a basic timer, (b) a timer/counter, (c) a timer with a terminal count, (d) a 16/32-bit timer, (e) a timer with a prescaler.

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 MHz) = 10 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 nanoseconds = 200 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 nanoseconds = 655,35 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 E^2s, 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.

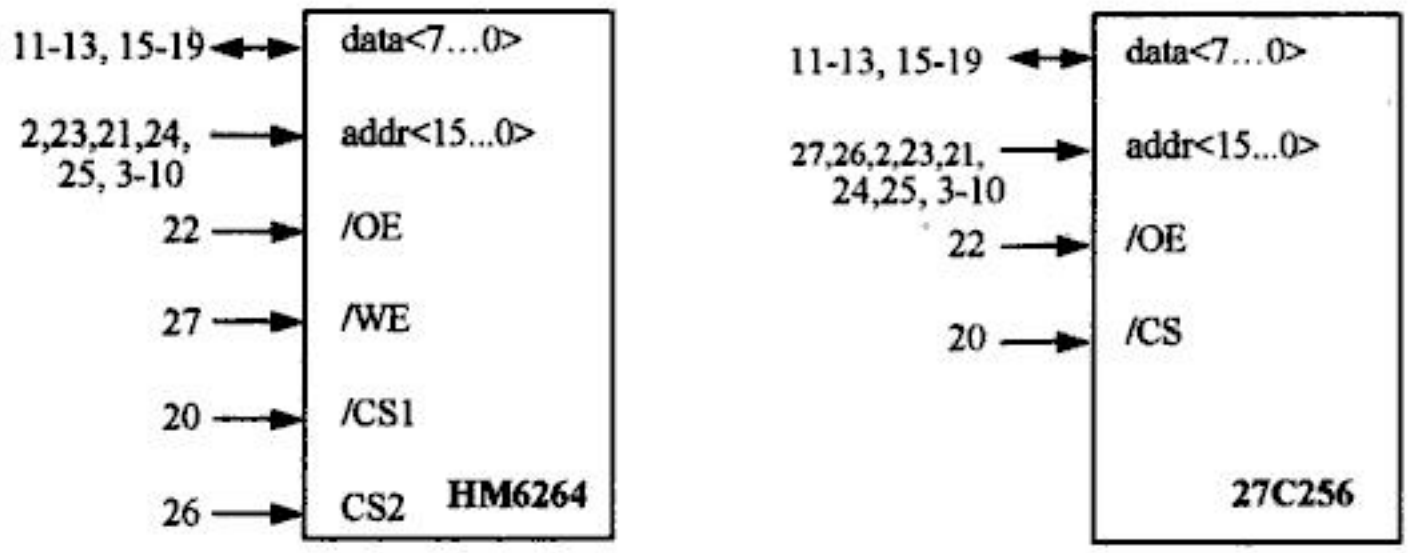
Flash Memory

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.

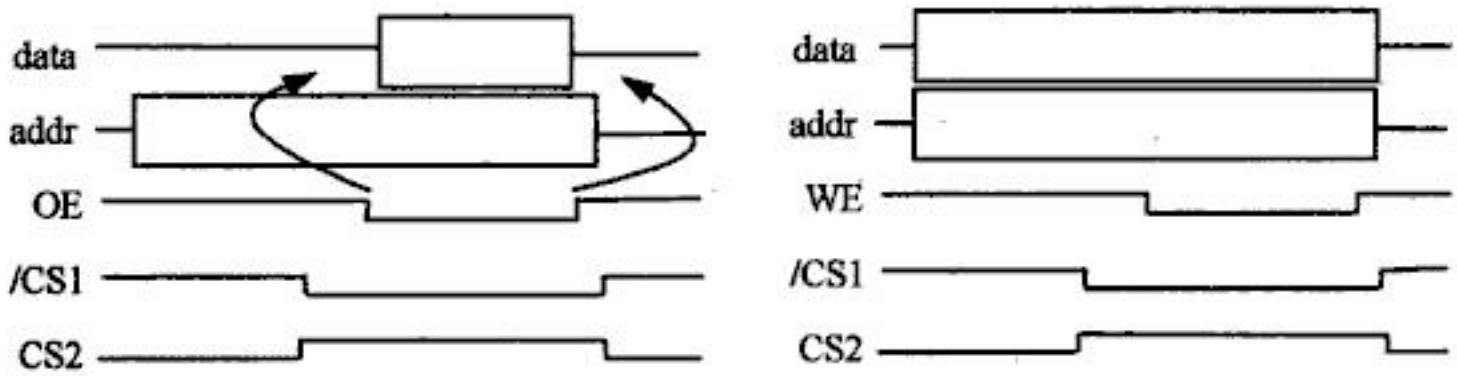
You have either reached a page that is unavailable for viewing or reached your viewing limit for this book.



(a)

Device	Access Time (ns)	Standby Pwr. (mW)	Active Pwr. (mW)	Vcc Voltage (V)
HM6264	85-100	.01	15	5
27C256	90	.5	100	5

(b)



(c)

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

Index

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.

WSE

WILEY

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