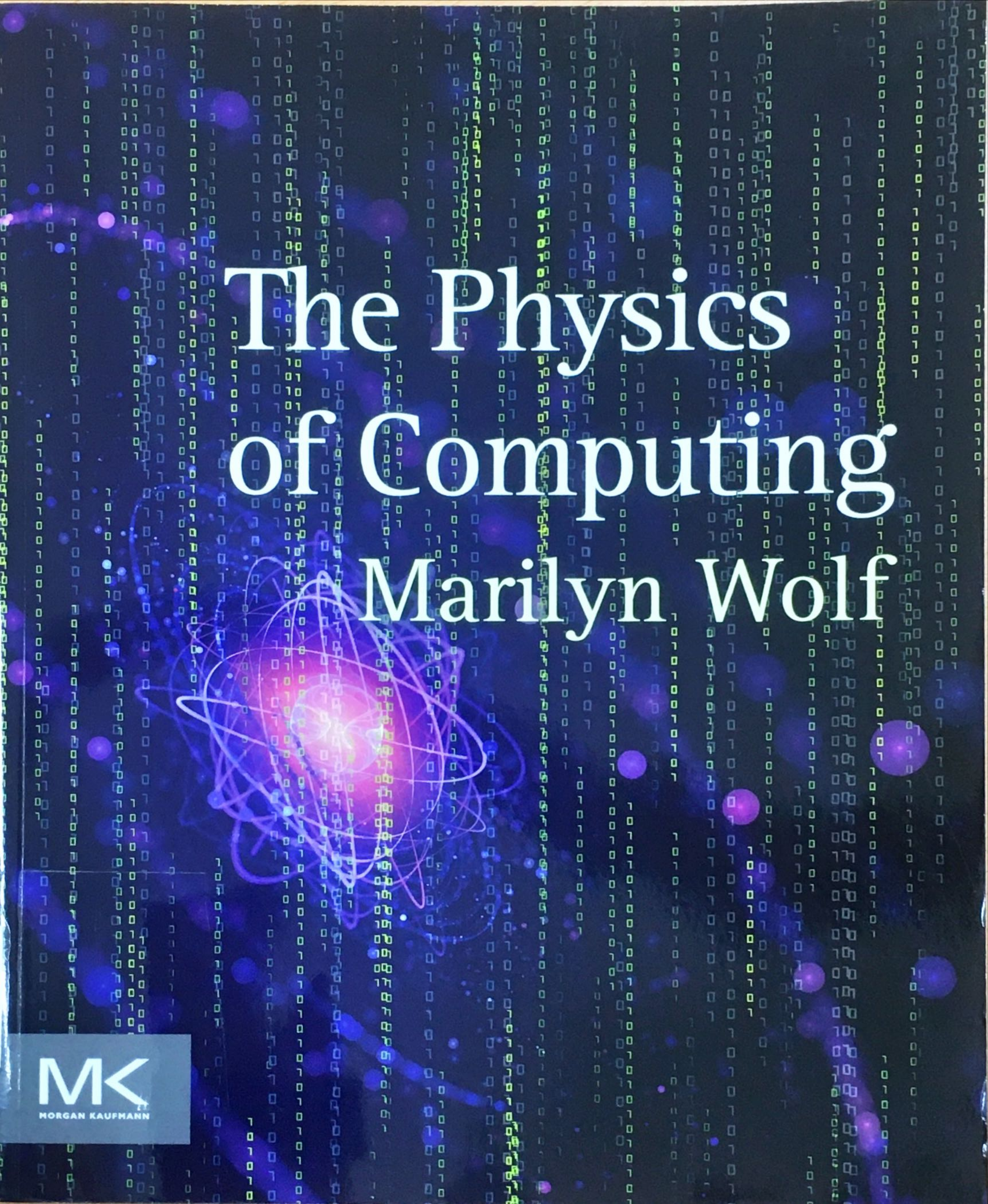
The Physics of Computing

Marilyn Wolf



MK
MORGAN KAUFMANN

The Physics of Computing

Marilyn Wolf

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Morgan Kaufmann is an imprint of Elsevier

MK
MORGAN KAUFMANN

Morgan Kaufmann is an imprint of Elsevier
50 Hampshire Street, 5th Floor, Cambridge, MA 02139, United States

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the Library of Congress

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

ISBN: 978-0-12-809381-8

Publisher: Todd Green
Acquisition Editor: Steve Merken
Editorial Project Manager: Nate McFadden
Production Project Manager: Mohanambal Natarajan
Cover Designer: Matthew Limbert

Typeset by TNQ Books and Journals

Contents

Preface ix

CHAPTER 1 Electronic Computers 1
- **1.1** Introduction 1
- **1.2** The long road to computers 1
  - **1.2.1** Mechanical computing devices 2
  - **1.2.2** Theories of computing 4
  - **1.2.3** Electronic computers 6
- **1.3** Computer system metrics 9
- **1.4** A tour of this book 10
- **1.5** Synthesis 11
- Questions 11

CHAPTER 2 Transistors and Integrated Circuits 13
- **2.1** Introduction 13
- **2.2** Electron devices and electronic circuits 13
  - **2.2.1** Early vacuum tube devices 13
  - **2.2.2** Vacuum tube triode 15
- **2.3** Physics of materials 18
  - **2.3.1** Metals 19
  - **2.3.2** Boltzmann's constant and temperature 22
  - **2.3.3** Semiconductors 24
- **2.4** Solid-state devices 29
  - **2.4.1** Semiconductor diode 30
  - **2.4.2** MOS capacitor 33
  - **2.4.3** Basic MOSFET operation 37
  - **2.4.4** Advanced MOSFET characteristics 45
- **2.5** Integrated circuits 48
  - **2.5.1** Moore's Law 49
  - **2.5.2** Manufacturing processes 51
  - **2.5.3** Lithography 55
  - **2.5.4** Yield 57
  - **2.5.5** Separation of concerns 58
- **2.6** Synthesis 59
- Questions 59

CHAPTER 3 Logic Gates 63
- **3.1** Introduction 63
- **3.2** The CMOS inverter 64

3.3 Static gate characteristics 67
3.4 Delay 72
3.4.1 Transistor models 73
3.4.2 RC models for delay 76
3.4.3 Drive and loads 83
3.5 Power and energy 85
3.6 Scaling theory 90
3.7 Reliability 94
3.8 Synthesis 96
Questions 96

CHAPTER 4 Sequential Machines 99
4.1 Introduction 99
4.2 Combinational logic 99
4.2.1 The event model 99
4.2.2 The network model 100
4.2.3 Gain and reliability 103
4.2.4 Gain and delay 105
4.2.5 Delay and power 108
4.2.6 Noise and reliability in logic and interconnect 109
4.2.7 Power supply and reliability 109
4.2.8 Noise and input/output coupling 115
4.3 Interconnect 116
4.3.1 Parasitic impedance 116
4.3.2 Transmission lines 118
4.3.3 Crosstalk 124
4.3.4 Wiring complexity and Rent's Rule 126
4.4 Sequential machines 128
4.4.1 Sequential models 128
4.4.2 Registers 131
4.4.3 Clocking 134
4.4.4 Metastability 140
4.5 Synthesis 144
Questions 145

CHAPTER 5 Processors and Systems 149
5.1 Introduction 149
5.2 System reliability 150
5.3 Processors 153
5.3.1 Microprocessor characteristics 153
5.3.2 Busses and interconnect 155
5.3.3 Global communication 159
5.3.4 Clocking 161

5.4 Memory 168
5.4.1 Memory structures 168
5.4.2 Memory system performance 172
5.4.3 DRAM systems 174
5.4.4 DRAM reliability 176
5.5 Mass storage 177
5.5.1 Magnetic disk drives 177
5.5.2 Flash memory 178
5.5.3 Storage and performance 181
5.6 System power consumption 182
5.6.1 Server systems 182
5.6.2 Mobile systems and batteries 185
5.6.3 Power management 188
5.7 Heat transfer 189
5.7.1 Heat transfer characteristics 190
5.7.2 Heat transfer modeling 192
5.7.3 Heat and reliability 198
5.7.4 Thermal management 200
5.8 Synthesis 201
Questions 201

CHAPTER 6 Input and Output 205
6.1 Introduction 205
6.2 Displays 205
6.3 Image sensors 211
6.4 Touch sensors 216
6.5 Microphones 217
6.6 Accelerometers and inertial sensors 218
6.7 Synthesis 220
Questions 220

CHAPTER 7 Emerging Technologies 221
7.1 Introduction 221
7.2 Carbon nanotubes 221
7.2.1 Nanotube transistors 222
7.3 Quantum computers 223
7.4 Synthesis 228

Appendix A: Useful Constants and Formulas 229
Appendix B: Circuits 233
Appendix C: Probability 241
Appendix D: Advanced Topics 243
References 253
Index 259

Preface

This book is an attempt to take a step back and provide a longer, more foundational view of computer engineering. While a lot of software design proceeded from the theoretical to the experimental—think about sorting, for example—computer system design has long taken an example-driven approach. Studying particular computer systems was useful in the early days when we were not quite sure what major concepts undergirded the design of these systems. But today we are much more capable of making generalized statements about many topics in computer engineering. Those foundational topics should form the core of our approach to computer engineering.

A foundational approach to digital systems is particularly important because it is so hard today to touch and see logic in operation. When I was a student, most systems were boards made of SSI and MSI logic. We had no choice but to deal with circuits and waveforms. Today, all that is hidden inside a chip; even boards have smaller wires and are harder to analyze. Many students today think that 1s and 0s float through chips—they have no conception of voltages and currents in computers.

This book does not ask students to design *anything*. Instead, we walk students through the principles that define the computer system design space. I like to think of the concepts in this course as knobs and meters on a control panel. Changing the settings of knobs change the values on the meters; the twist of one knob may change several meters. Consider, for example, the cascading effects of reducing MOS dielectric thickness: transistor transconductance changes, which in turn affects gate delay for the better and leakage current for the worse.

Engineers do not design systems in a vacuum. Instead, they design to a set of goals or requirements. The traditional goal for computer architects is performance, or more precisely throughput. But computers are actually designed with several other goals in mind, with energy/power and reliability paramount. Performance, energy, and reliability all have their roots in physical phenomenon. And they are inextricably linked by physics: improving one metric incurs costs in the other metrics. As in the rest of life, there is no free lunch in computer design.

Computer engineering is a relatively young field. Introductory courses have concentrated on building and taxonomy—build something, see the different ways to build something. As the field matures, it is time to start thinking about new ways to present the basics. The biologist E.O. Wilson said, "A field is initially defined by the questions it asks and eventually defined by the answers it provides." After 70 years of electronic computer engineering, it is time to start thinking of the field in terms of answers.

I have been thinking about computer design for a long time. Performance was for many years the paramount metric for computer design, although the ASIC world was always interested in area as a proxy for cost. I started to think more seriously about other fundamental limits, particularly power, for the class on pervasive information systems taught by Perry Cook and myself. A new course at Georgia Tech gave me the impetus to take this train of thought to its logical conclusion.

This course takes a broad view of both computer engineering and physics. Some of the most basic phenomena in computer architecture and even software design—the memory wall, the power wall, the race to dark—are due to fundamental physics. But the physics required to understand these major computational issues is much more than classical Shockley semiconductor theory. We need to understand thermodynamics, electrostatics, and good deal of circuit theory.

As I thought about this material, I came to regard Boltzmann's constant is a key concept. *k* pops up everywhere: the diode equation, Arrhenius's equation, temperature, and the list goes on. Boltzmann's constant links temperature and energy, so it should not be a surprise that it is so tightly woven into the topics of this book.

Some of the material in this book is unique to modern CMOS—leakage mechanisms. But some of the topics can be applied to a wide range of circuit and device technologies. Delay through logic networks, metastability, and the fundamentals of reliability are examples of foundational concepts that computer engineers should understand even if CMOS is swept off the map by some other technology.

Parts of this book will probably strike some readers as hopelessly simplified and compressed. I hope that those same readers will find other parts of the book dense and hard to follow. The only way to understand how computers work is to understand the relationships between an entire collection of topics that do not always seem at first glance to be related. We know those knobs on the design process are linked because a change to one parameter that we think will help our design often results in side effects that negate much of the benefit we sought. I worked very hard to find the simplest possible description for a lot of concepts. Hopefully they are presented in a way that gives a basic understanding. The reader who is interested in a deeper understanding of one of these topics can pursue each one in more depth. But the singular goal of this book is to provide a unified description of the fundamental physical principles of computing machines.

Both computer engineers and electrical engineers are potential audiences for this book. These two groups come to the material with very different backgrounds. Computer engineers often have limited experience in circuit design. While they may have some knowledge of ideas like Kirchhoff's laws, they usually are not extremely comfortable with circuit analysis. Electrical engineers may not always have a detailed understanding of computer architecture. One of my challenges in writing this book was to provide enough background for each audience without overloading them.

I have mentioned a number of historical discoveries and inventions for several reasons. First, going through the sequence of ideas that led to today's design practices helps both to remind us that there is often more than one way to do things and to highlight the true advantages of the methods that were settled upon by history. Second, semiconductor physics and computer engineering have produced some of the most important inventions of the 20th century. These inventions will be used for centuries to come. We should not let ourselves become so accustomed to these ideas that we lose all capacity to marvel at them and at their inventors.

Richard Feynman's *Lectures on Computing* was an early inspiration for this class, but much of that book is devoted to quantum computing. That book does not consider many topics in traditional computing. For example, metastability is a fundamental physical concept in computer system design that Feynman does not touch. We salute him for his early recognition of the physical nature of computing. We also thank him for the *Lectures on Physics*, which provided a clear, concise way to think about many of the basic physical phenomena that underlie computing.

Several tips of the hat are in order. My friend and colleague Saibal Mukhopodhyay proposed the *Physical Foundations of Computer Engineering* course and provided many concrete suggestions, particularly on reliability and leakage. My thanks go out to him for his patience and insight. Dave Coelho graciously provided information on his power distribution system. Kees Vissers invented the welder current comparison. Alec Ishii advised me on clock distribution. Kevin Cao suggested how to make best use of the Predictive Technology Models. Bruce Jacob gave insight into DRAM. Srini Devadas's suggestions led to the advanced topics appendix. Tom Conte provided the core memory and Pentium Pro as well as many discussions on the present state and future of computing. Thanks to the reviewers for their helpful comments. And many thanks to my editor Nate McFadden for guiding this book through the long development and production process. Any faults in this book can be traced entirely and solely to me.

Marilyn Wolf
Atlanta, GA

CHAPTER 1

Electronic Computers

1.1 Introduction

We take computers for granted. But what is a computer? The computers we use every day can be summarized as **EBT**: *electronic*, *binary*, *Turing machines*. This combination of technologies has allowed us to conquer problems as diverse as weather forecasting, self-driving vehicles, and shopping.

But the path that led us to today's computers is long, and the EBT approach to computing is surprisingly recent. Some of the history helps us understand why we build computers the way we do. Machines that could compute—calculate with numbers and perform other multistep tasks—were originally conceived of and built with the only devices available, namely mechanical devices. These mechanical computers had severe limitations that were overcome only with the creation of electronic devices.

We will start with a brief introduction to the history of computing machines. We will then discuss the theory of computing that was developed based on this early experience and which underpins our modern ideas of computers. We then discuss the metrics we use to evaluate computer designs and the trade-offs between these goals that physics imposes on us. We will wrap up with a survey of the remainder of this book.

1.2 The long road to computers

Mechanical computation was inspired in large part by the need to control machines. People could not react quickly and reliably enough to maintain the proper operation of the equipment that appeared during the Industrial Revolution. Early mechanical computing devices were analog—they operated on continuous values. But machines that manipulated discrete values arrived surprisingly early. Eventually, theory was developed to describe these types of machines. One of those theories, the *Turing machine*, is the blueprint for the way in which we design computers today.

The Physics of Computing. http://dx.doi.org/10.1016/B978-0-12-809381-8.00001-8

1.2.1 Mechanical computing devices

Mechanical devices are much older than electrical devices. The subtlety required to make machines with complex behaviors should not be minimized. Those early inventors lacked hardened metals with which to make both the machines themselves and the tools required to shape the parts. They also lacked machine tools that we take for granted today. Even relatively simple machines needed to be controlled without human intervention. The invention of the steam engine spurred demand for automatic control of machines.

Control

The **governor** is a prime example of an early analog computer built from mechanical components. James Watt, the inventor of the first practical steam engine, designed a governor to control the speed of the steam engine, and similar devices had been used before. He built his governor in 1788. Fig. 1.1 shows the operation of Watt's governor. The central shaft is rotated and is connected to the steam engine's crankshaft. Two shafts are hinged at the top, with a weight at the end of each. A set of levers connect the weighted shafts to the engine's throttle—the higher the weights, the more closed the throttle. As the central shaft (and the engine's crankshaft) rotates at higher speed, the weights lift themselves up due to centrifugal force, causing the engine to be throttled down. This mechanism implements **negative feedback**, a fundamental principle in control.

Memory

The Jacquard loom demonstrates an important concept, built with mechanical elements, which would be fundamental to our modern conception of computers: **discrete memory**. The loom, which was created in 1801, was invented to automate the weaving of complex patterns in fabrics. It created a huge mass market for sophisticated fabrics that previously had been available only to the very wealthy. As Fig. 1.2 shows, a fabric consists of threads in both *x*, known as the weft thread, and the warp thread in *y*. The loom's shuttle moves the warp thread back and forth. Lifting the weft threads with small hooks known as bolus hooks selectively causes the warp thread to be either above or below the weft, selecting when it will be visible. The more complex the pattern, the more complex the patterns of thread liftings; a complex pattern could lift hundreds of threads in a different pattern on each row. The Jacquard loom stored that complex pattern in punched card: a hole in the card allowed the bolus hook to

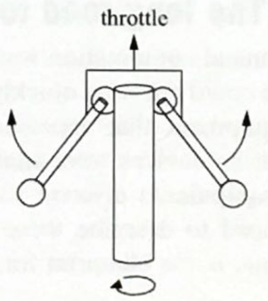


FIGURE 1.1

A mechanical governor.

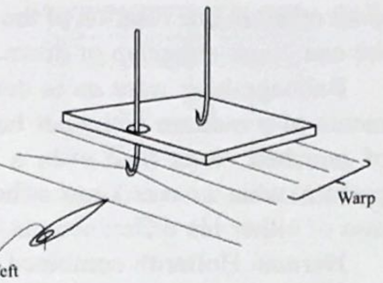


FIGURE 1.2
Operation of a Jacquard loom.

descend and grab its weft thread; no hole meant that the weft was not lifted at that point. Not only could the card represent complex two-dimensional patterns, but the entire fabric's pattern could be changed simply by replacing the card.

Babbage machines

Charles Babbage started work in 1822 on a **difference machine** to calculate numbers, specifically to compute polynomial functions. His work was motivated by the need to tables of functions that were more accurate than those compiled by hand calculation, an error-prone process. At the heart of this machine's operation is the use of gears to count; this principle was also used in the mechanical calculators that were widely used well into the second half of the 20th century. The basic principle behind mechanical counting is simple and elegant, as shown in Fig. 1.3. We use a shaft whose position represents the value of a digit in the counting sum. We can put a radial mark on the end of each shaft, then a dial around the shaft with a position for each digit; the mark points the value of the shaft's digit. To build a two-digit counter, we connect the two shafts with gears. In this figure, the small gear has 5 teeth and the large gear has 15 teeth, giving a gearing ratio of 3:1. Three full rotations of the small gear produce one complete rotation of the large gear. The large gear remembers the number of teeth turned against it by the small gear up to a limit of 15. If we want to count in base 10, we gear the shafts at a 10:1 ratio: 10 rotations of the ones-digit

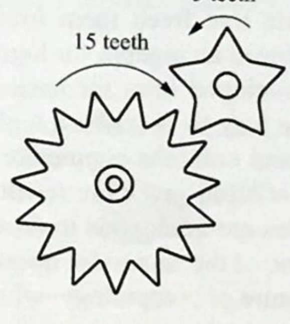


FIGURE 1.3
Using gears to count.