HZ BOOKS
华章教育

ELSEVIER

经 典 原 版 书 库

嵌入式计算系统设计原理

（美）Marilyn Wolf
佐治亚理工学院 著

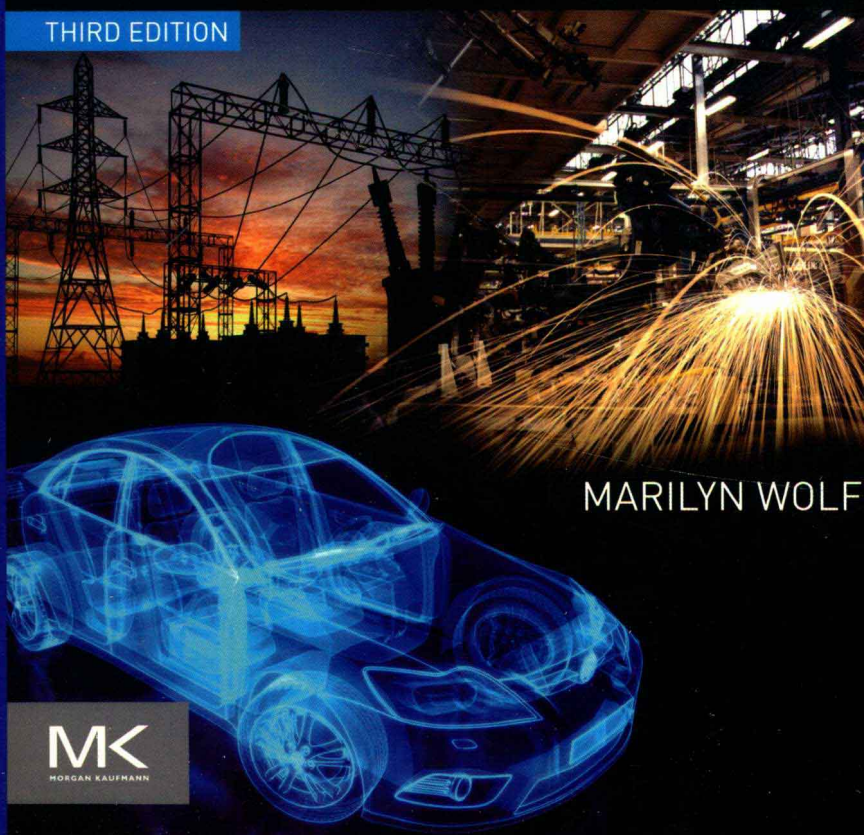
（英文版·第3版）

Principles of Embedded Computing System Design (Third Edition)

COMPUTERS AS COMPONENTS

PRINCIPLES OF EMBEDDED COMPUTING SYSTEM DESIGN

THIRD EDITION



MARILYN WOLF

机械工业出版社
China Machine Press

MK
MORGAN KAUFMANN

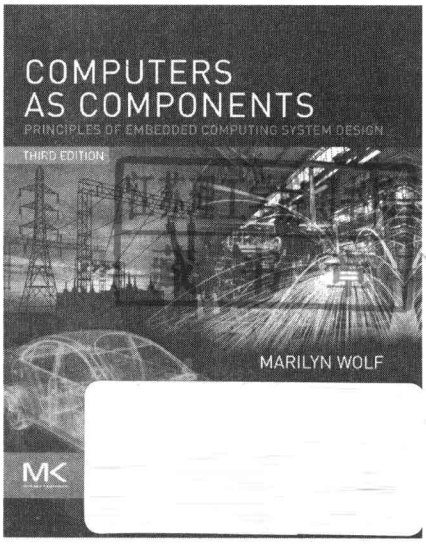
经 典 原 版 书 库

嵌入式计算系统设计原理

（英文版·第3版）

Computers as Components

Principles of Embedded Computing System Design (Third Edition)



（美）Marilyn Wolf 著
佐治亚理工学院

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

嵌入式计算系统设计原理（英文版·第3版）/（美）沃尔夫（Wolf, M.）著. —北京：机械工业出版社，2013.1
（经典原版书库）
书名原文：Computers as Components: Principles of Embedded Computing System Design, Third Edition
ISBN 978-7-111-41228-1
I. 嵌… II. 沃… III. 微型计算机－系统设计－英文 IV. TP36
中国版本图书馆CIP数据核字（2013）第012569号

版权所有·侵权必究
封底无防伪标均为盗版
本书法律顾问 北京市展达律师事务所

本书版权登记号：图字：01-2012-7895

Computers as Components: Principles of Embedded Computing System Design, Third Edition
Marilyn Wolf
ISBN: 978-0-12-388436-7
Copyright © 2012 by Elsevier Inc. All rights reserved.
Authorized English language reprint edition published by the Proprietor.
Copyright © 2013 by Elsevier (Singapore) Pte Ltd. All rights reserved.
Elsevier (Singapore) Pte Ltd.
3 Killiney Road
#08-01 Winsland House I
Singapore 239519
Tel: (65) 6349-0200
Fax: (65) 6733-1817
First Published 2013
Printed in China by China Machine Press under special arrangement with Elsevier (Singapore) Pte Ltd. This edition is authorized for sale in China only, excluding Hong Kong SAR, Macau SAR and Taiwan.
Unauthorized export of this edition is a violation of the Copyright Act. Violation of this Law is subject to Civil and Criminal Penalties.

本书英文影印版由Elsevier (Singapore) Pte Ltd. 授权机械工业出版社在中国大陆境内独家发行。本版仅限在中国境内（不包括香港特别行政区、澳门特别行政区及台湾地区）出版及标价销售。未经许可之出口，视为违反著作权法，将受法律之制裁。
本书封底贴有Elsevier防伪标签，无标签者不得销售。

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）
责任编辑：迟振春
北京诚信伟业印刷有限公司印刷
2013年2月第1版第1次印刷
186mm × 240mm · 32.5印张
标准书号：ISBN 978-7-111-41228-1
定价：79.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S. Tanenbaum，Bjarne Stroustrup，Brain W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com

电子邮件：hzjsj@hzbook.com

联系电话：（010）88379604

联系地址：北京市西城区百万庄南街 1 号

邮政编码：100037

华章教育

华章科技图书出版中心

To Dad, for everything he taught me

试读结束，需要全本PDF请购买 www.ertongbook.com

Foreword to the First Edition

Digital system design has entered a new era. At a time when the design of microprocessors has shifted into a classical optimization exercise, the design of embedded computing systems in which microprocessors are merely components has become a wide-open frontier. Wireless systems, wearable systems, networked systems, smart appliances, industrial process systems, advanced automotive systems, and biologically interfaced systems provide a few examples from across this new frontier.

Driven by advances in sensors, transducers, microelectronics, processor performance, operating systems, communications technology, user interfaces, and packaging technology on the one hand, and by a deeper understanding of human needs and market possibilities on the other, a vast new range of systems and applications is opening up. It is now up to the architects and designers of embedded systems to make these possibilities a reality.

However, embedded system design is practiced as a craft at the present time. Although knowledge about the component hardware and software subsystems is clear, there are no system design methodologies in common use for orchestrating the overall design process, and embedded system design is still run in an ad hoc manner in most projects.

Some of the challenges in embedded system design come from changes in underlying technology and the subtleties of how it can all be correctly mingled and integrated. Other challenges come from new and often unfamiliar types of system requirements. Then too, improvements in infrastructure and technology for communication and collaboration have opened up unprecedented possibilities for fast design response to market needs. However, effective design methodologies and associated design tools haven't been available for rapid follow-up of these opportunities.

At the beginning of the VLSI era, transistors and wires were the fundamental components, and the rapid design of computers on a chip was the dream. Today the CPU and various specialized processors and subsystems are merely basic components, and the rapid, effective design of very complex embedded systems is the dream. Not only are system specifications now much more complex, but they must also meet real-time deadlines, consume little power, effectively support complex real-time user interfaces, be very cost-competitive, and be designed to be upgradable.

Wayne Wolf has created the first textbook to systematically deal with this array of new system design requirements and challenges. He presents formalisms and a methodology for embedded system design that can be employed by the new type of "tall-thin" system architect who really understands the foundations of system design across a very wide range of its component technologies.

Moving from the basics of each technology dimension, Wolf presents formalisms for specifying and modeling system structures and behaviors and then clarifies these ideas through a series of design examples. He explores the complexities involved and how to systematically deal with them. You will emerge with a sense of clarity about

the nature of the design challenges ahead and with knowledge of key methods and tools for tackling those challenges.

As the first textbook on embedded system design, this book will prove invaluable as a means for acquiring knowledge in this important and newly emerging field. It will also serve as a reference in actual design practice and will be a trusted companion in the design adventures ahead. I recommend it to you highly.

Lynn Conway
Professor Emerita, Electrical Engineering and Computer Science, University of Michigan

Preface to the First Edition

Microprocessors have long been a part of our lives. However, microprocessors have become powerful enough to take on truly sophisticated functions only in the past few years. The result of this explosion in microprocessor power, driven by Moore's Law, is the emergence of embedded computing as a discipline. In the early days of microprocessors, when all the components were relatively small and simple, it was necessary and desirable to concentrate on individual instructions and logic gates. Today, when systems contain tens of millions of transistors and tens of thousands of lines of high-level language code, we must use design techniques that help us deal with complexity.

This book tries to capture some of the basic principles and techniques of this new discipline of embedded computing. Some of the challenges of embedded computing are well known in the desktop computing world. For example, getting the highest performance out of pipelined, cached architectures often requires careful analysis of program traces. Similarly, the techniques developed in software engineering for specifying complex systems have become important with the growing complexity of embedded systems. Another example is the design of systems with multiple processes. The requirements on a desktop general-purpose operating system and a real-time operating system are very different; the real-time techniques developed over the past 30 years for larger real-time systems are now finding common use in microprocessor-based embedded systems.

Other challenges are new to embedded computing. One good example is power consumption. While power consumption has not been a major consideration in traditional computer systems, it is an essential concern for battery-operated embedded computers and is important in many situations in which power supply capacity is limited by weight, cost, or noise. Another challenge is deadline-driven programming. Embedded computers often impose hard deadlines on completion times for programs; this type of constraint is rare in the desktop world. As embedded processors become faster, caches and other CPU elements also make execution times less predictable. However, by careful analysis and clever programming, we can design embedded programs that have predictable execution times even in the face of unpredictable system components such as caches.

Luckily, there are many tools for dealing with the challenges presented by complex embedded systems: high-level languages, program performance analysis tools, processes and real-time operating systems, and more. But understanding how all these tools work together is itself a complex task. This book takes a bottom-up approach to understanding embedded system design techniques. By first understanding the fundamentals of microprocessor hardware and software, we can build powerful abstractions that help us create complex systems.

A note to embedded system professionals

This book is not a manual for understanding a particular microprocessor. Why should the techniques presented here be of interest to you? There are two reasons. First, techniques such as high-level language programming and real-time operating systems are very important in making large, complex embedded systems that actually work. The industry is littered with failed system designs that didn't work because their designers tried to hack their way out of problems rather than stepping back and taking a wider view of the problem. Second, the components used to build embedded systems are constantly changing, but the principles remain constant. Once you understand the basic principles involved in creating complex embedded systems, you can quickly learn a new microprocessor (or even programming language) and apply the same fundamental principles to your new components.

A note to teachers

The traditional microprocessor system design class originated in the 1970s when microprocessors were exotic yet relatively limited. That traditional class emphasizes breadboarding hardware and software to build a complete system. As a result, it concentrates on the characteristics of a particular microprocessor, including its instruction set, bus interface, and so on.

This book takes a more abstract approach to embedded systems. While I have taken every opportunity to discuss real components and applications, this book is fundamentally not a microprocessor data book. As a result, its approach may seem initially unfamiliar. Rather than concentrating on particulars, the book tries to study more generic examples to come up with more generally applicable principles. However, I think that this approach is both fundamentally easier to teach and in the long run more useful to students. It is easier because one can rely less on complex lab setups and spend more time on pencil-and-paper exercises, simulations, and programming exercises. It is more useful to the students because their eventual work in this area will almost certainly use different components and facilities than those used at your school. Once students learn fundamentals, it is much easier for them to learn the details of new components.

Hands-on experience is essential in gaining physical intuition about embedded systems. Some hardware building experience is very valuable; I believe that every student should know the smell of burning plastic integrated circuit packages. But I urge you to avoid the tyranny of hardware building. If you spend too much time building a hardware platform, you will not have enough time to write interesting programs for it. And as a practical matter, most classes don't have the time to let students build sophisticated hardware platforms with high-performance I/O devices and possibly multiple processors. A lot can be learned about hardware by measuring and evaluating an existing hardware platform. The experience of programming complex embedded systems will teach students quite a bit about hardware as well—debugging interrupt-driven code is an experience that few students are likely to forget.

A home page for the book (*www.mkp.com/embed*) includes overheads, instructor's manual, lab materials, links to related Web sites, and a link to a password-protected ftp site that contains solutions to the exercises.

Acknowledgments

I owe a word of thanks to many people who helped me in the preparation of this book. Several people gave me advice about various aspects of the book: Steve Johnson (Indiana University) about specification, Louise Trevillyan and Mark Charney (both IBM Research) on program tracing, Margaret Martonosi (Princeton University) on cache miss equations, Randy Harr (Synopsys) on low power, Phil Koopman (Carnegie Mellon University) on distributed systems, Joerg Henkel (NEC C&C Labs) on low-power computing and accelerators, Lui Sha (University of Illinois) on real-time operating systems, John Rayfield (ARM) on the ARM architecture, David Levine (Analog Devices) on compilers and SHARC, and Con Korikis (Analog Devices) on the SHARC. Many people acted as reviewers at various stages: David Harris (Harvey Mudd College); Jan Rabaey (University of California at Berkeley); David Nagle (Carnegie Mellon University); Randy Harr (Synopsys); Rajesh Gupta, Nikil Dutt, Frederic Doucet, and Vivek Sinha (University of California at Irvine); Ronald D. Williams (University of Virginia); Steve Sapiro (SC Associates); Paul Chow (University of Toronto); Bernd G. Wenzel (Eurostep); Steve Johnson (Indiana University); H. Alan Mantooth (University of Arkansas); Margarida Jacome (University of Texas at Austin); John Rayfield (ARM); David Levine (Analog Devices); Ardsher Ahmed (University of Massachusetts/Dartmouth University); and Vijay Madisetti (Georgia Institute of Technology). I also owe a big word of thanks to my editor, Denise Penrose. Denise put in a great deal of effort finding and talking to potential users of this book to help us understand what readers wanted to learn. This book owes a great deal to her insight and persistence. Cheri Palmer and her production team did an excellent job on an impossibly tight schedule. The mistakes and miscues are, of course, all mine.

Preface to the Second Edition

Embedded computing is more important today than it was in 2000, when the first edition of this book appeared. Embedded processors are in even more products, ranging from toys to airplanes. Systems-on-chips now use up to hundreds of CPUs. The cell phone is on its way to becoming the new standard computing platform. As my column in *IEEE Computer* in September 2006 indicated, there are at least a half-million embedded systems programmers in the world today, probably closer to 800,000.

In this edition I've tried to both update and to revamp. One major change is that the book now uses the TI C55x DSP. I seriously rewrote the discussion of real-time scheduling. I have tried to expand on performance analysis as a theme at as many levels of abstraction as possible. Given the importance of multiprocessors in even the most mundane embedded systems, this edition also talks more generally about hardware/software co-design and multiprocessors.

One of the changes in the field is that this material is taught at lower and lower levels of the curriculum. What used to be graduate material is now upper-division undergraduate; some of this material will percolate down to the sophomore level in the foreseeable future. I think that you can use subsets of this book to cover both more advanced and more basic courses. Some advanced students may not need the background material of the earlier chapters and you can spend more time on software performance analysis, scheduling, and multiprocessors. When teaching introductory courses, software performance analysis is an alternative path to exploring microprocessor architectures as well as software; such courses can concentrate on the first few chapters.

The new Web site for this book and my other books is http://www.waynewolf.com. On this site, you can find overheads for the material in this book, suggestions for labs, and links to more information on embedded systems.

Acknowledgments

I'd like to thank a number of people who helped me with this second edition. Cathy Wicks and Naser Salameh of Texas Instruments gave me invaluable help in figuring out the C55x. Richard Barry of freeRTOS.org not only graciously allowed me to quote from the source code of his operating system but he also helped clarify the explanation of that code. My editor at Morgan Kaufmann, Chuck Glaser, knew when to be patient, when to be encouraging, and when to be cajoling. (He also has great taste in sushi restaurants.) And of course, Nancy and Alec patiently let me type away. Any problems, small or large, with this book are, of course, solely my responsibility.

Wayne Wolf
Atlanta, GA

试读结束，需要全本PDF请购买 www.ertongbook.com

Preface to the Third Edition

This third edition reflects the continued evolution of my thoughts on embedded computing and the suggestions of the users of this book. One important goal was expanding the coverage of embedded computing applications. Learning about topics like digital still cameras and cars can take a lot of effort. Hopefully this material will provide some useful insight into the parts of these systems that most directly affect the design decision faced by embedded computing designers. I also expanded the range of processors used as examples. I included sophisticated processors including the TI C64x and advanced ARM extensions. I also included the PIC16F to illustrate the properties of small RISC embedded processors. Finally, I reorganized the coverage of networks and multiprocessors to provide a more unified view of these closely related topics. You can find additional material on the course Web site at http://www.marilynwolf.us. The site includes a complete set of overheads, sample labs, and pointers to additional information.

I'd like to thank Nate McFadden, Todd Green, and Andre Cuello for their editorial patience and care during this revision. I'd also like to thank the anonymous reviewers and Prof. Andrew Pleszkun of the University of Colorado for their insightful comments on drafts. And I have a special thanks for David Anderson, Phil Koopman, and Bruce Jacob who helped me figure out some things. I'd also like to thank the Atlanta Snowpocalypse of 2011 for giving me a large block of uninterrupted writing time.

Most important of all, this is the right time to acknowledge the profound debt of gratitude I owe to my father. He taught me how to work: not just how to do certain things, but how to approach problems, develop ideas, and bring them to fruition. Along the way, he taught me how to be a considerate, caring human being. Thanks, Dad.

Marilyn Wolf
Atlanta, GA
December 2011

Contents

Foreword to the First Edition v
Preface to the First Edition vii
Preface to the Second Edition x
Preface to the Third Edition xi

CHAPTER 1 Embedded Computing 1
1.1 Introduction 1
1.2 Complex systems and microprocessors 1
1.2.1 Embedding computers 2
1.2.2 Characteristics of embedded computing applications 4
1.2.3 Why use microprocessors? 6
1.2.4 Cyber-physical systems 7
1.2.5 Challenges in embedded computing system design 8
1.2.6 Performance of embedded computing systems 9
1.3 The embedded system design process 10
1.3.1 Requirements 12
1.3.2 Specification 16
1.3.3 Architecture design 17
1.3.4 Designing hardware and software components 19
1.3.5 System integration 19
1.3.6 Formalisms for system design 20
1.3.7 Structural description 21
1.3.8 Behavioral description 25
1.4 Design example: Model train controller 28
1.4.1 Requirements 29
1.4.2 DCC 30
1.4.3 Conceptual specification 32
1.4.4 Detailed specification 35
1.4.5 Lessons learned 42
1.5 A guided tour of this book 42
1.5.1 Chapter 2: Instruction sets 43
1.5.2 Chapter 3: CPUs 43
1.5.3 Chapter 4: Computing platforms 44
1.5.4 Chapter 5: Program design and analysis 44
1.5.5 Chapter 6: Processes and operating systems 45
1.5.6 Chapter 7: System design techniques 46
1.5.7 Chapter 8: Networks and multiprocessors 47

1.6 Summary 47
What we learned 48
Further reading 48
Questions 48
Lab exercises 50

CHAPTER 2 Instruction Sets 51
2.1 Introduction 51
2.2 Preliminaries 51
2.2.1 Computer architecture taxonomy 52
2.2.2 Assembly languages 54
2.2.3 VLIW processors 56
2.3 ARM processor 57
2.3.1 Processor and memory organization 58
2.3.2 Data operations 59
2.3.3 Flow of control 66
2.3.4 Advanced ARM features 72
2.4 PICmicro mid-range family 73
2.4.1 Processor and memory organization 73
2.4.2 Data operations 73
2.4.3 Flow of control 76
2.5 TI C55x DSP 77
2.5.1 Processor and memory organization 78
2.5.2 Addressing modes 81
2.5.3 Data operations 83
2.5.4 Flow of control 84
2.5.5 C coding guidelines 86
2.6 TI C64x 87
2.7 Summary 90
What we learned 90
Further reading 91
Questions 91
Lab exercises 93

CHAPTER 3 CPUs 95
3.1 Introduction 95
3.2 Programming input and output 96
3.2.1 Input and output devices 96
3.2.2 Input and output primitives 98
3.2.3 Busy-wait I/O 99
3.2.4 Interrupts 101

3.3 Supervisor mode, exceptions, and traps 114
3.3.1 Supervisor mode 114
3.3.2 Exceptions 115
3.3.3 Traps 115
3.4 Co-processors 115
3.5 Memory system mechanisms 116
3.5.1 Caches 116
3.5.2 Memory management units and address translation 123
3.6 CPU performance 128
3.6.1 Pipelining 128
3.6.2 Cache performance 132
3.7 CPU power consumption 133
3.8 Design example: Data compressor 137
3.8.1 Requirements and algorithm 137
3.8.2 Specification 140
3.8.3 Program design 141
3.8.4 Testing 148
3.9 Summary 149
What we learned 149
Further reading 150
Questions 150
Lab exercises 153

CHAPTER 4 Computing Platforms 155
4.1 Introduction 155
4.2 Basic computing platforms 155
4.2.1 Platform hardware components 156
4.2.2 Platform software components 158
4.3 The CPU bus 159
4.3.1 Bus organization and protocol 160
4.3.2 DMA 167
4.3.3 System bus configurations 170
4.4 Memory devices and systems 172
4.4.1 Memory system organization 174
4.5 Designing with computing platforms 176
4.5.1 Example platforms 176
4.5.2 Choosing a platform 176
4.5.3 Intellectual property 179
4.5.4 Development environments 180
4.5.5 Debugging techniques 181
4.5.6 Debugging challenges 183

4.6 Consumer electronics architecture 185
4.6.1 Consumer electronics use cases and requirements ... 185
4.6.2 File systems 187
4.7 Platform-level performance analysis 188
4.8 Design example: Alarm clock 193
4.8.1 Requirements 193
4.8.2 Specification 194
4.8.3 System architecture 197
4.8.4 Component design and testing 200
4.8.5 System integration and testing 200
4.9 Design example: Audio player 200
4.9.1 Theory of operation and requirements 200
4.9.2 Specification 202
4.9.3 System architecture 204
4.9.4 Component design and testing 206
4.9.5 System integration and debugging 206
4.10 Summary 207
What we learned 207
Further reading 207
Questions 207
Lab exercises 210

CHAPTER 5 Program Design and Analysis 213
5.1 Introduction 213
5.2 Components for embedded programs 214
5.2.1 State machines 214
5.2.2 Circular buffers and stream-oriented programming 216
5.2.3 Queues and producer/consumer systems 221
5.3 Models of programs 223
5.3.1 Data flow graphs 224
5.3.2 Control/data flow graphs 226
5.4 Assembly, linking, and loading 228
5.4.1 Assemblers 229
5.4.2 Linking 233
5.4.3 Object code design 235
5.5 Compilation techniques 236
5.5.1 The compilation process 236
5.5.2 Basic compilation methods 237
5.5.3 Compiler optimizations 245
5.6 Program-level performance analysis 254
5.6.1 Elements of program performance 256
5.6.2 Measurement-driven performance analysis 259

5.7 Software performance optimization 262
5.7.1 Loop optimizations 262
5.7.2 Cache optimizations 264
5.7.3 Performance optimization strategies 265
5.8 Program-level energy and power analysis and optimization 266
5.9 Analysis and optimization of program size 270
5.10 Program validation and testing 271
5.10.1 Clear-box testing 271
5.10.2 Black-box testing 278
5.10.3 Evaluating functional tests 279
5.11 Design example: Software modem 280
5.11.1 Theory of operation and requirements 280
5.11.2 Specification 283
5.11.3 System architecture 283
5.11.4 Component design and testing 284
5.11.5 System integration and testing 285
5.12 Design example: Digital still camera 285
5.12.1 Theory of operation and requirements 285
5.12.2 Specification 290
5.12.3 System architecture 293
5.12.4 Component design and testing 296
5.12.5 System integration and testing 296
5.13 Summary 296
What we learned 296
Further reading 297
Questions 297
Lab exercises 305

CHAPTER 6 Processes and Operating Systems 307
6.1 Introduction 307
6.2 Multiple tasks and multiple processes 308
6.2.1 Tasks and processes 308
6.3 Multirate systems 310
6.3.1 Timing requirements on processes 311
6.3.2 CPU usage metrics 316
6.3.3 Process state and scheduling 316
6.3.4 Running periodic processes 317
6.4 Preemptive real-time operating systems 319
6.4.1 Two basic concepts 320
6.4.2 Processes and context 321
6.4.3 Processes and object-oriented design 324

试读结束，需要全本PDF请购买 www.ertongbook.com