Embedded Systems Design with Platform FPGAs
CONTENTS

Preface ........................................................................................................... xi
Acknowledgments ......................................................................................... xv

Chapter 1 Introduction ................................................................................. 1
1.1 Embedded Systems ............................................................................... 3
  1.1.1 Embedded and General-Purpose ................................................ 4
  1.1.2 Hardware, Software, and FPGAs .............................................. 6
  1.1.3 Execution Models .................................................................... 7
1.2 Design Challenges .............................................................................. 10
  1.2.1 Design Life Cycle ................................................................. 10
  1.2.2 Measures of Success ............................................................. 12
  1.2.3 Costs ..................................................................................... 17
1.3 Platform FPGAs ................................................................................. 20
Chapter in Review ...................................................................................... 23
1.A Spectrometer Example ......................................................................... 25
  1.A.1 Scenario .................................................................................. 25
  1.A.2 Two Solutions ......................................................................... 25
  1.A.3 Discussion ............................................................................... 26
1.B Introducing the Platform FPGA Tool Chain ................................... 27
  1.B.1 Getting Started with Xilinx Platform Studio ......................... 28
  1.B.2 Using Xilinx Platform Studio ............................................... 29
Exercises ..................................................................................................... 41
References .................................................................................................. 42

Chapter 2 The Target .................................................................................. 43
2.1 CMOS Transistor ............................................................................... 44
2.2 Programmable Logic Devices ......................................................... 47
2.3 Field-Programmable Gate Array ....................................................... 49
  2.3.1 Function Generators ............................................................... 49
  2.3.2 Storage Elements .................................................................. 51
  2.3.3 Logic Cells .............................................................................. 51
  2.3.4 Logic Blocks ............................................................................ 52
  2.3.5 Input/Output Blocks ............................................................... 53
  2.3.6 Special-Purpose Function Blocks ........................................... 53
2.4 Hardware Description Languages ........................................ 57
   2.4.1 VHDL ................................................................. 58
   2.4.2 Verilog ............................................................... 68
   2.4.3 Other High-Level HDLs ........................................... 73
2.5 From HDL to Configuration Bitstream .................................. 74
   Chapter in Review ....................................................... 78
2.A Xilinx Virtex 5 ............................................................. 81
   2.A.1 Look-Up Table ...................................................... 82
   2.A.2 Slice ................................................................. 82
   2.A.3 Configurable Logic Block ........................................ 83
   2.A.4 Block RAM .......................................................... 83
   2.A.5 DSP Slices .......................................................... 84
   2.A.6 Select I/O ............................................................ 85
   2.A.7 High-Speed Serial Transceivers ................................ 85
   2.A.8 Clocks ................................................................. 86
   2.A.9 PowerPC 440 ......................................................... 87
2.B Xilinx Integrated Software Environment ............................. 88
   2.B.1 Overview of Commands ......................................... 88
2.C Creating and Generating Custom IP .................................. 94
   2.C.1 Xilinx Core Generator ............................................ 94
   2.C.2 Create/Import Peripheral Wizard ............................... 99
   2.C.3 Hardware Core Project Directory .............................. 105
   Exercises ................................................................ 111
   References .................................................................. 112

Chapter 3 System Design .................................................. 115
3.1 Principles of System Design .............................................. 117
   3.1.1 Design Quality ...................................................... 117
   3.1.2 Modules and Interfaces ......................................... 120
   3.1.3 Abstraction and State .......................................... 123
   3.1.4 Cohesion and Coupling ........................................ 125
   3.1.5 Designing for Reuse ............................................. 128
3.2 Control Flow Graph ..................................................... 129
3.3 Hardware Design ......................................................... 133
   3.3.1 Origins of Platform FPGA Designs .......................... 133
   3.3.2 Platform FPGA Components ................................. 135
   3.3.3 Adding to Platform FPGA Systems ....................... 142
   3.3.4 Assembling Custom Compute Cores ....................... 144
3.4 Software Design .......................................................... 153
   3.4.1 System Software Options ...................................... 154
   3.4.2 Root Filesystem ................................................... 156
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4.3 Cross-Development Tools</td>
<td>158</td>
</tr>
<tr>
<td>3.4.4 Monitors and Bootloader</td>
<td>159</td>
</tr>
<tr>
<td>Chapter in Review</td>
<td>161</td>
</tr>
<tr>
<td>3.A Platform FPGA Architecture Design</td>
<td>162</td>
</tr>
<tr>
<td>3.A.1 Xilinx EDK and IBM CoreConnect</td>
<td>162</td>
</tr>
<tr>
<td>3.A.2 Building Base Systems</td>
<td>166</td>
</tr>
<tr>
<td>3.A.3 Augmenting Base Systems</td>
<td>167</td>
</tr>
<tr>
<td>3.A.4 XPS Project Files</td>
<td>167</td>
</tr>
<tr>
<td>3.A.5 Practical Example: Floating-Point Adder</td>
<td>169</td>
</tr>
<tr>
<td>3.A.6 Base System</td>
<td>169</td>
</tr>
<tr>
<td>3.A.7 Create and Import Peripheral Wizard</td>
<td>170</td>
</tr>
<tr>
<td>3.A.8 Core Generator</td>
<td>170</td>
</tr>
<tr>
<td>3.A.9 User Logic</td>
<td>171</td>
</tr>
<tr>
<td>3.A.10 Modify Hardware Core Project Files</td>
<td>175</td>
</tr>
<tr>
<td>3.A.11 Connecting the Hardware Core to the Base System</td>
<td>176</td>
</tr>
<tr>
<td>3.A.12 Testing the System</td>
<td>176</td>
</tr>
<tr>
<td>3.B Embedded GNU/Linux System</td>
<td>178</td>
</tr>
<tr>
<td>3.B.1 Organization of Unix Filesystem</td>
<td>178</td>
</tr>
<tr>
<td>3.B.2 Configuration Software and Tools</td>
<td>179</td>
</tr>
<tr>
<td>3.B.3 Cross-Development Tools and Libraries</td>
<td>184</td>
</tr>
<tr>
<td>3.B.4 Cross-Compiling Linux</td>
<td>187</td>
</tr>
<tr>
<td>3.B.5 Building a Root Filesystem</td>
<td>190</td>
</tr>
<tr>
<td>3.B.6 Booting Linux on the ML510</td>
<td>192</td>
</tr>
<tr>
<td>Exercises</td>
<td>194</td>
</tr>
<tr>
<td>References</td>
<td>195</td>
</tr>
</tbody>
</table>

**Chapter 4 Partitioning**                                                      | 197  |
| 4.1 Overview of Partitioning Problem                                     | 198  |
| 4.2 Analytical Solution to Partitioning                                  | 201  |
| 4.2.1 Basic Definitions                                                  | 202  |
| 4.2.2 Expected Performance Gain                                           | 205  |
| 4.2.3 Resource Considerations                                             | 207  |
| 4.2.4 Analytical Approach                                                 | 209  |
| 4.3 Communication                                                         | 211  |
| 4.3.1 Invocation/Coordination                                             | 214  |
| 4.3.2 Transfer of State                                                   | 218  |
| 4.4 Practical Issues                                                      | 223  |
| 4.4.1 Profiling Issues                                                    | 223  |
| 4.4.2 Data Structures                                                     | 226  |
| 4.4.3 Manipulate Feature Size                                             | 228  |
Chapter in Review ........................................................ 228
4.A Profiling with Gprof ...................................................... 229
4.B Linux Kernel ............................................................... 233
  4.B.1 Kernel Modules .......................................................... 233
  4.B.2 Address Spaces .......................................................... 236
  4.B.3 Application View ......................................................... 238
  4.B.4 Char Device Driver ...................................................... 239
  4.B.5 Summary ................................................................. 241
Exercises ................................................................... 242
References ................................................................ 245

Chapter 5 Spatial Design ................................................ 247
5.1 Principles of Parallelism .................................................. 248
  5.1.1 Granularity ............................................................... 249
  5.1.2 Degree of Parallelism ................................................... 252
  5.1.3 Spatial Organizations ................................................... 252
5.2 Identifying Parallelism .................................................... 258
  5.2.1 Ordering ................................................................. 258
  5.2.2 Dependence .............................................................. 260
  5.2.3 Uniform Dependence Vectors ......................................... 264
5.3 Spatial Parallelism with Platform FPGAs ......................... 266
  5.3.1 Parallelism within FPGA Hardware Cores ....................... 268
  5.3.2 Parallelism within FPGA Designs ................................... 273
Chapter in Review ........................................................ 274
5.A Useful VHDL Topics for Spatial Design ......................... 275
  5.A.1 Constants and Generics ................................................ 275
  5.A.2 User-Defined Types .................................................... 276
  5.A.3 Generate Statements ................................................... 278
  5.A.4 Design Constraints ...................................................... 279
5.B Debugging Platform FPGA Designs ............................... 280
  5.B.1 Simulation ............................................................... 280
  5.B.2 Software Addressable Registers ..................................... 283
  5.B.3 Xilinx ChipScope ........................................................ 284
Exercises ................................................................... 292
References ................................................................ 293

Chapter 6 Managing Bandwidth .................................... 295
6.1 Balancing Bandwidth .................................................... 296
  6.1.1 Kahn Process Network ............................................. 298
CONTENTS

6.1.2 Synchronous Design ................................................ 300
6.1.3 Asynchronous Design ................................................ 300
6.2 Platform FPGA Bandwidth Techniques ................................ 300
  6.2.1 On-Chip and Off-Chip Memory ....................................... 301
  6.2.2 Streaming Instrument Data ............................................ 313
  6.2.3 Practical Issues ........................................................... 316
6.3 Scalable Designs .......................................................... 317
  6.3.1 Scalability Constraints ................................................ 317
  6.3.2 Scalability Solutions .................................................... 321
Chapter in Review .......................................................... 324
6.A On-Chip Memory Access ................................................ 326
  6.A.1 FIFOs ...................................................................... 326
  6.A.2 Block RAM ............................................................... 328
  6.A.3 LocalLink Interface ...................................................... 328
  6.B.1 Programmable I/O ....................................................... 330
  6.B.2 Central DMA Controller ................................................ 330
  6.B.3 Bus Master ............................................................... 332
  6.B.4 Native Port Interface .................................................... 336
Exercises ................................................................... 346
References ................................................................. 346

Chapter 7 Outside World .................................................. 347
  7.1 Point-to-Point Communication ........................................... 348
    7.1.1 RS-232 ..................................................................... 349
    7.1.2 Other Low-Speed Communication .................................... 350
  7.2 Internetworking Communication ..................................... 350
    7.2.1 Concepts .................................................................. 350
    7.2.2 Application Interface .................................................. 355
    7.2.3 Higher Level Protocols .............................................. 358
    7.2.4 Operating System Configuration .................................... 362
Chapter in Review .......................................................... 363
  7.A High-Speed Serial Communication .................................. 365
    7.A.1 RocketIO .................................................................. 365
    7.A.2 Aurora Example ....................................................... 365
    7.A.3 LocalLink Interface ...................................................... 366
    7.A.4 Clock Correction ....................................................... 366
    7.A.5 Error Testing ............................................................ 367
    7.A.6 Loopback ................................................................. 367
  7.B Low-Speed Communication ........................................... 368
7.B.1 Generating the Hardware Base System ........................................... 368
7.B.2 Testing the Design ....................................................................... 373
Exercises ......................................................................................... 374
References ......................................................................................... 374

Glossary ........................................................................................... 375

Index ................................................................................................. 381
PREFACE

Xilinx, Inc. introduced the Field-Programmable Gate Array (FPGA) in 1984 as an advanced programmable logic device. It is now part of a multi-billion dollar market and FPGAs have made their way into products as diverse as digital cameras, automobiles, and network switches that drive the Internet. FPGAs have even flown to Mars (Ratter, 2004).

Almost since its inception, people have recognized the potential of using these devices to build custom computing architectures, but to date the market is overwhelmingly “glue logic” and prototyping. Nonetheless, advances in process technology have yielded modern FPGAs with very large capacities and a wide range of features built into the chip. The confluence of these features — which include multiple processors, large amounts of memory, hundreds of multipliers, and high-speed I/O — have reached a critical mass: now more than ever, Platform FPGAs are poised to realize a more prominent role in computing systems.

This ability to deploy sophisticated computing systems on a single FPGA device is likely to make a significant impact on embedded computing systems. While small (indeed tiny) 8 and 16 bit computing systems are and will remain a very important segment of the embedded systems market, trends over the last several years suggest that the use of standard, off-the-shelf 32-bit processors for embedded systems is growing fast. These higher-end embedded systems come with high levels of integration, often incorporating a significant portion of the system on a (fixed and manufactured) chip. There are many benefits to this level of integration but one significant drawback is that much of the system architecture is predetermined and may not be optimal for the particular targeted application. The alternative, developing a system architecture on a custom System-on-a-Chip (SoC), is too costly except for high volume (millions of units) products. When the application fits the resources provided, all is well. But oftentimes the application does not fit and some of the integrated resources are wasted while additional discrete hardware is required to make up the deficiency. With the Platform FPGA, engineers gain all of the advantages of integration but retain the flexibility to engineer a balanced system architecture on a per-application basis.

Armed with multiple kinds of buses, various direct communication links, bridges, I/O components, and an assortment of other special-purpose Intellectual Property cores, Platform FPGA designers can readily customize their system architecture. For example, with hundreds of distributed block RAMs, a designer can configure one large addressable memory, disperse the RAM throughout the system as individual buffers, or design some combination of the two. Application-specific functionality (i.e., custom hardware cores) can be designed and incorporated. Although Application-Specific Integrated Circuits (ASICs) will always outperform the identical FPGA implementation, the FPGA solution avoids the expense, risk, and time-to-market issues associated with manufacturing of an ASIC. The FPGA is a virtual blank slate giving the engineer the ability to provision the resources to best fit the application after the manufacture, test, and verification of the physical device. This flexibility increases efficiency of the system solution and every discrete component saved reduces cost and increases reliability.
Of course, there is a price to be paid for this enormous degree of hardware flexibility. In addition to compilers, debuggers, and other conventional software tools used for processor-based embedded system development, the Platform FPGA designer has to be fluent in hardware design and synthesis as well as system integration tools. Decisions that previously were largely constrained by architecture choice — such as how to partition the application between hardware and software — now have a much larger solution space. Beyond understanding the characteristics of a particular processor’s system bus, the designer has to weigh the strengths and weaknesses of multiple communication mechanisms. Balancing complex networks of on-chip components presents a novel challenge for those used to working with predetermined architectures.

In many ways, these challenges embody computer engineering. However, until the Platform FPGA emerged, practical issues (such as the cost of building custom silicon solutions) made a hands-on study prohibitively expensive. Students learned computer system architecture from textbooks and conceptual models. The practical aspects of engineering custom computer systems from the ground up was typically the purview of a few specialists. Knowledge and practical tools — such as how to create a Board Support Package for an embedded system — was typically learned on the job.

The aim of this textbook is to introduce the reader to system development on a Platform FPGA. The focus is on embedded systems but it also serves as a general guide to building custom computing systems. The text describes the fundamental technology in terms of hardware, software, and a set of principles to guide the development of Platform FPGA systems. The goal is to show how to systematically and creatively apply these principles to construction of application-specific embedded system architectures. There is a strong focus on using Free and Open Source software to increase productivity.

The organization of each chapter in the book includes two parts. The white pages describe concepts, principles, and general knowledge. The gray pages include a technical rendition of the main issues of the chapter and show the concepts applied in practice. This includes step-by-step details for a specific developer board and chain so that readers can carry out the same steps on their own. Rather than try to demonstrate the concepts on a broad set of tools and boards, the text uses a single set of tools (Xilinx Platform Studio, Linux, and GNU) throughout and uses single developer board (Xilinx ML-510) for the examples. The belief is that a single system, completely described, is more valuable to the reader than partial information about a range of systems.

**How to Read This Book**

This book was designed to make it easy for a number of different readers to quickly find the information they need. If you are an undergraduate with a software background who might not have had a course in electronics, the white pages in chapter two start with a basic transistor and describes how an FPGA — a solid state device — implements programmable hardware. If you are a practicing engineer with experience in embedded systems but are new to FPGAs, then reading just the blue pages will let you skip the theory and focus on the practical aspects of building Linux-based systems on an FPGA. If you are a student in a senior design course, the lectures probably focus on project management. You won’t find that material here but you may find this text to be a handy practical guide to complete your project. On the other hand, if you have a very specific project involving cutting-edge technology, then you may not find that
particular topic covered in sufficient depth here. However, if you are looking to get started, need a general overview of the concepts, and still want enough step-by-step details to implement a real, working system this may prove to be the perfect text for you!

**Note to Instructors**

There are several roles this text may play in a typical computer curriculum. In many departments, a single embedded systems course is a technical elective offered to Seniors. The core material is delivered within the context of a substantial, semester-long project. This text provides a suitable introduction to embedded systems and — because it provides all of the practical material students need to carry-out their projects — the instructor has the freedom to introduce select topics in class. For curricula with a focus area in embedded systems (offering multiple technical electives), this text may be used to teach a course on reconfigurable computing. Such a course would complement a comprehensive embedded systems course that gives a full treatment of the area including small systems and an expanded description of real-time issues. Finally, the IEEE-CS/ACM Joint Task Force on Computing Curricula (Ironman Draft) includes as a basic computer engineering component a “culminating project.” This often manifests itself in current curricula as a Senior Design or Capstone course. As such, the course draws together subjects taught independently in the curriculum and helps bridge the students’ academic and professional careers. FPGAs serve exceedingly well (and are often used) in such a course. Their flexibility allows a wide range of potential projects and those projects necessarily encompass both hardware and software components: the essence of computer engineering. For that reason, many instructors may find this text to be an excellent resource for the students in their culminating project.

**Online Materials**

Many of the commands, scripts, and URL links referenced in the text are included in the text. However, for the reader’s convenience, the publisher maintains a Web site with all of these materials online including scripts too long to insert into the text. In addition to links pointing to the latest version of the open source software used in the text, the Web site also archives the exact (known to work) versions used in this text.

**Reference**

ACKNOWLEDGMENTS

There have been many people that have contributed to this project and helped improve the book in numerous ways. We would like to acknowledge the current students in the Reconfigurable Computing Systems lab (University of North Carolina at Charlotte) for reading chapters and providing feedback — sometimes with very short notice. These include William V. Kritikos, Scott Buscemi, Bin Huang, Shanyuan Gao, Robin Jacob Pottathuparambil, Siddhartha Datta, Ashwin A. Mendon, Shweta Jain, Yamuna Rajasekhar, and Rahul R. Sharma. Other colleagues and previous students have also contributed; the authors would also like to thank Brian Greskamp (D.E. Shaw), Srinivas Beeravolu (Xilinx, Inc.), Parag Beeraka (AMD Inc.), and David Andrews (University of Arkansas) for their help. Many of the questions and examples have been adapted from our collaborations with these individuals. Several external reviewers provided valuable feedback. We would like to thank Roy Kravitz (Serveron, A Division of BPL Global), Duncan Buell (University of South Carolina), Cameron Patterson (Virginia Tech), and Jim Turley (Embedded Technology Journal). We would also like to thank the team at Morgan-Kaufmann; in particular, Nate McFadden and Andre Cuello, were instrumental in helping us complete the manuscript on time. Also, we would like to thank Xilinx — they encouraged us to start this project and then gave us the freedom to complete it as we saw fit. Finally, the authors would like to thank our families — Jennie, Joseph, and Hilary — for giving us the time to complete this project. You guys are the best!