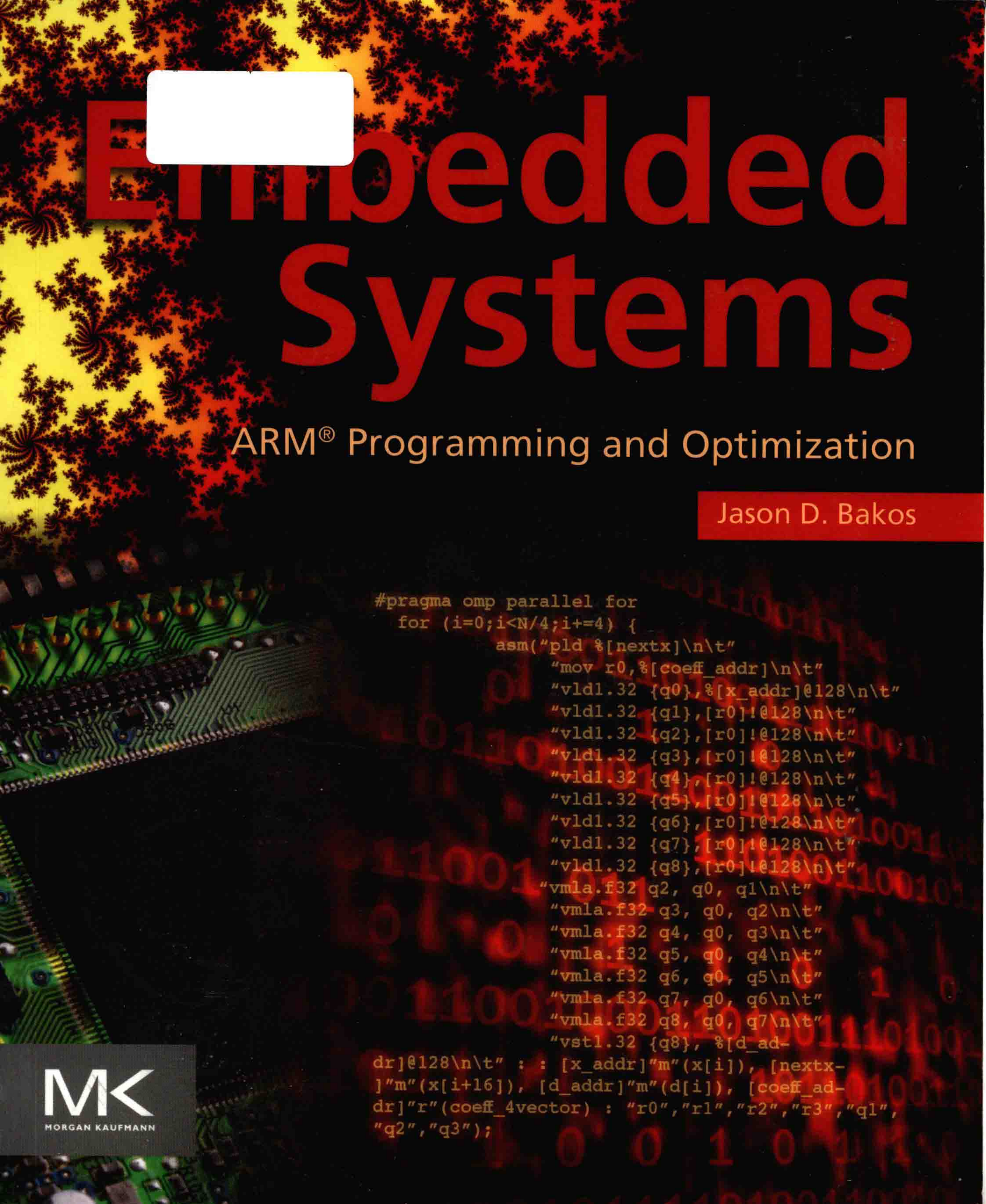
Embedded Systems

ARM® Programming and Optimization

Jason D. Bakos



```
#pragma omp parallel for
  for (i=0;i<N/4;i+=4) {
          asm("pld %[nextx]\n\t"
               "mov r0,%[coeff_addr]\n\t"
               "vld1.32 {q0},%[x_addr]@128\n\t"
               "vld1.32 {q1},[r0]!@128\n\t"
               "vld1.32 {q2},[r0]!@128\n\t"
               "vld1.32 {q3},[r0]!@128\n\t"
               "vld1.32 {q4},[r0]!@128\n\t"
               "vld1.32 {q5},[r0]!@128\n\t"
               "vld1.32 {q6},[r0]!@128\n\t"
               "vld1.32 {q7},[r0]!@128\n\t"
               "vld1.32 {q8},[r0]!@128\n\t"
              "vmla.f32 q2, q0, q1\n\t"
               "vmla.f32 q3, q0, q2\n\t"
               "vmla.f32 q4, q0, q3\n\t"
               "vmla.f32 q5, q0, q4\n\t"
               "vmla.f32 q6, q0, q5\n\t"
               "vmla.f32 q7, q0, q6\n\t"
               "vmla.f32 q8, q0, q7\n\t"
               "vst1.32 {q8}, %[d_ad-
dr]@128\n\t" : : [x_addr]"m"(x[i]), [nextx-
]"m"(x[i+16]), [d_addr]"m"(d[i]), [coeff_ad-
dr]"r"(coeff_4vector) : "r0","r1","r2","r3","q1",
"q2","q3");
```

MK
MORGAN KAUFMANN

Embedded Systems

ARM® Programming and Optimization

Jason D. Bakos

Department of Computer Science and Engineering
University of South Carolina
Columbia, SC

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

Acquiring Editor: *Steve Merken*
Editorial Project Manager: *Nathaniel McFadden*
Project Manager: *Sujatha Thirugnana Sambandam*
Designer: *Mark Rogers*

Morgan Kaufmann is an imprint of Elsevier
225 Wyman Street, Waltham, MA 02451, USA

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: 978-0-12-800342-8

Embedded Systems

For Lumi, Jade, and Justin

Preface

For many years I have worked in the area of *reconfigurable computing*, whose goal is to develop tools and methodologies to facilitate the use of field programmable gate arrays (FPGAs) as co-processors for high-performance computer systems.

One of the main challenges in this discipline is the "programming problem," in which the practical application of FPGAs is fundamentally limited by their tedious and error-prone programming model. This is of particular concern because this problem is a consequence of the technology's strengths: FPGAs operate with fine grain concurrency, where the programmer can control the simultaneous behavior of every circuit on the chip. Unfortunately, this control also requires that the programmer manage fine grain constraints such as on-chip memory usage and routing congestion. The CPU programmer, on the other hand, needs only consider the potential state of the CPU at each line of code, while on-chip resources are automatically managed by the hardware at runtime.

I recently realized that modern embedded systems may soon face a similar programming problem. Battery technology continues to remain relatively stagnant, and the slowing of Moore's Law became painfully evident after the nearly 6-year gap between 65 and 28 nm fabrication technology. At the same time, consumers have come to expect the continued advancement of embedded system capabilities, such as being able to run real-time augmented reality software on a processor that fits in a pair of eyeglasses.

Given these demands for energy efficiency and performance, many embedded processor vendors are seeking more energy-efficient approaches to microarchitecture, often involving targeting the types of parallelism that cannot be automatically extracted from software. This will require cooperation of the programmers to write parallel code. This is a lot of to ask of programmers, who will need to juggle both functionality and performance on a resource- and power-constrained platform that includes a wide range of potential sources of parallelism from multicores to GPU shader units.

Many universities have developed "unified" parallel programming courses that cover the spectrum of parallel programming from distributed systems to manycore processors. However, the topic is most often taught from the perspective of high-performance computing as opposed to embedded computing.

With the recent explosion of advanced embedded platforms such as the Raspberry Pi, I saw a need to develop curriculum that combines topics from computer architecture and parallel programming for performance-oriented programming of embedded systems. I also wanted to include interesting and relevant projects and case studies for the course to avoid the traditional types of dull course projects associated with embedded systems courses (e.g., blink the light) and parallel programming courses (e.g., write and optimize a Fast Fourier Transform).

While using these ideas in my own embedded systems course, and I often find the students competing among themselves to achieve the fastest image rotation or the fastest Mandelbrot set generator. This type of collegial competition cultivates excitement for the material.

USING THIS BOOK

This book is intended for use in a junior- or senior-level undergraduate course in a computer science or computer engineering curriculum. Although a course in embedded systems may focus on subtopics such as control theory, robotics, low power design, real-time systems, or other related topics, this book is intended as an introduction to *performance-oriented* programming for lightweight system-on-chip embedded processors.

This book should accompany an embedded design platform such as a Raspberry Pi, on which the student can evaluate the practices and methodologies described.

When using this text, students are expected to know the C programming language, have a basic knowledge of the Linux operating system, and understand basic concurrency such as task synchronization.

INSTRUCTOR SUPPORT

Lecture slides, exercise solutions, and errata are provided at the companion website:
textbooks.elsevier.com/9780128003428

Acknowledgments

Several students assisted me in the development of this book.

During spring and summer 2013, undergraduate students **Benjamin Morgan**, **Jonathan Kilby**, **Shawn Weaver**, **Justin Robinson**, and **Amadeo Bellotti** evaluated the DMA controller and performance monitoring unit on the Raspberry Pi's Broadcom BCM2835 and the Xilinx Zynq 7020.

During summer 2014, undergraduate student **Daniel Clements** helped develop a uniform approach for using the Linux perf_event on the ARM11, ARM Cortex A9, and ARM Cortex A15. Daniel also evaluated Imagination Technology's OpenCL runtime and characterized its performance on the PowerVR 544 GPU on our ODROID XU Exynos 5 platform.

During summer 2015, undergraduate student **Friel "Scottie" Scott** helped evaluate the Mali T628 GPU on the ODROID-XU3 platform and proofread Chapter 5.

Much of my insight about memory optimizations for computer vision algorithms were an outgrowth of my graduate student **Fan Zhang**'s dissertation on auto-optimization of stencil loops on the Texas Instruments Keystone Digital Signal Processor architecture.

I would like to thank the following reviewers, who provided feedback, insight, and helpful suggestions at multiple points throughout the development of the book:

- **Miriam Leeser**, Northeastern University
- **Larry D. Pyeatt**, South Dakota School of Mines and Technology
- **Andrew N. Sloss**, University of Washington, Consulting Engineer at ARM Inc.
- **Amr Zaky**, Santa Clara University

I would like to thank Morgan Kaufmann and specifically to **Nate McFadden** for his constant encouragement and limitless patience throughout the writing. I am especially grateful for Nate's open-mindedness and flexibility with regard to the content, which continually evolved to keep current with new ARM-based embedded development platforms being released while I was developing the content. I also wish to thank **Sujatha Thirugnana Sambandam** for her detail-oriented editing and to **Mark Rogers** for designing the cover.

Contents