PROCEEDINGS

SPIE—The International Society for Optical Engineering

High-Speed Computing, Digital Signal Processing, and Filtering Using Reconfigurable Logic

John Schewel
Peter M. Athanas
V. Michael Bove, Jr.
John Watson
Chairs/Editors

20–21 November 1996
Boston, Massachusetts

SPIE P PROCEEDINGS SERIES

Volume 2914

PROCEEDINGS
SPIE—The International Society for Optical Engineering

High-Speed Computing, Digital Signal Processing, and Filtering Using Reconfigurable Logic

John Schewel
Peter M. Athanas
V. Michael Bove, Jr.
John Watson
Chairs/Editors

20–21 November 1996
Boston, Massachusetts

Sponsored and Published by
SPIE—The International Society for Optical Engineering

SPIE
P
PROCEEDINGS
SERIES

Volume 2914

SPIE is an international technical society dedicated to advancing engineering and scientific applications of optical, photonic, imaging, electronic, and optoelectronic technologies.

The papers appearing in this book comprise the proceedings of the meeting mentioned on the cover and title page. They reflect the authors' opinions and are published as presented and without change, in the interests of timely dissemination. Their inclusion in this publication does not necessarily constitute endorsement by the editors or by SPIE.

Please use the following format to cite material from this book:
Author(s), "Title of paper," in *High-Speed Computing, Digital Signal Processing, and Filtering Using Reconfigurable Logic,* John Schewel, Peter M. Athanas, V. Michael Bove, Jr., John Watson, Editors, Proc. SPIE 2914, page numbers (1996).

Library of Congress Catalog Card No. 96-69766
ISBN 0-8194-2316-5

Published by
SPIE—The International Society for Optical Engineering
P.O. Box 10, Bellingham, Washington 98227-0010 USA
Telephone 360/676-3290 (Pacific Time) • Fax 360/647-1445

Printed in the United States of America.

Using Reconfigurable Hardware to Customize Memory Hierarchies

Peixin Zhong and Margaret Martonosi
Department of Electrical Engineering
Princeton University
Princeton, NJ 08544-5263
{pzhong, martonosi}@ee.princeton.edu

Abstract

Over the past decade or more, processor speeds have increased much more quickly than memory speeds. As a result, a large, and still increasing, processor-memory performance gap has formed. Many significant applications suffer from substantial memory bottlenecks, and their memory performance problems are often either too unusual or extreme to be mitigated by cache memories alone. Such specialized performance "bugs" require specialized solutions, but it is impossible to provide case-by-case memory hierarchies or caching strategies on general-purpose computers.

We have investigated the potential of implementing mechanisms like victim caches and prefetch buffers in reconfigurable hardware to improve application memory behavior. Based on technology and commercial trends, our simulation-based studies use a forward-looking model in which configurable logic is located on the CPU chip. Given such assumptions, our results show that the flexibility of being able to specialize configurable hardware to an application's memory referencing behavior more than balances the slightly slower response times of configurable memory hierarchy structures. For our three applications, small, specialized memory hierarchy additions such as victim caches and prefetch buffers can reduce miss rates substantially and can drop total execution times for these programs to between 60 and 80% of their original execution times. Our results also indicate that different memory specializations may be most effective for each application; this highlights the usefulness of configurable memory hierarchies that are specialized on a per-application basis.

Keywords: memory latency, configurable computing, victim cache, prefetching.

1 Introduction

Due to rapid increases in microprocessor speeds, the performance gap between processors and main memory is widening. Cache memories are typically used in computer systems to bridge this performance gap and reduce the average memory access time. Although caches work well in many cases, they may still fail to provide high performance for certain applications.

Several hardware and software techniques have been proposed to improve cache performance in such cases. For example, prefetching techniques aim to hide the large latency out to main memory by bringing data to the cache before it is referenced. Victim caches attempt to reduce conflict misses in low-associativity caches. These hardware techniques have variable results depending on the application's memory referencing behavior. Their disadvantage is that they represent wasted transistor space on the CPU chip for those applications where they are ineffective. On the other hand, the drawback to purely software-based techniques (such as blocked matrix accesses or compiler-inserted prefetching directives) is that it can be difficult to statically analyze a program's memory behavior and determine when such techniques will be useful. For these reasons, this paper explores implementing memory hierarchy additions in *programmable* hardware.

Programmable logic, such as field-programmable gate arrays (FPGAs), has gained tremendous popularity in the past decade. Programmable logic is popular because a given chip's behavior can be configured and customized for different functions during different sessions. Customization on a per-application basis is feasible because the reconfiguration process is fast and can be done with the device in the system. Some FPGAs can even be partially reconfigured while the rest of the device is in use.

The configurability of FPGAs makes them heavily used for prototyping, but they have also been used to build high-performance application specific compute engines [12]. In addition, there has been work (such as PRISC [21] and PRISM [2]) on supplementing conventional processors with configurable coprocessors to accelerate performance for different applications

Thus far, most configurable computing projects have focused heavily on *computation* as opposed to *data access*. In this paper, we explore the potential of using configurable hardware to make application-specific improvements to memory behavior. We envision a library of parameterized memory hierarchy additions that can be invoked to target cases where the cache system does not work well for a particular application.

As fabrication technology improves, more and more transistors fit on a single chip. It seems likely that we will soon see processor chips that include a region of on-chip configurable logic. This configurable logic can clearly have many uses; our work does not preclude configuring the logic for more traditional compute-oriented uses, but simply attempts to explore an alternative use.

In Section 2, we will first discuss the structure of the configurable memory unit that we envision. Following that, in Section 3, we present case studies of applying the ideas of configurable memory to several applications. In Section 4, we discuss some of the hardware organization and implementation issues inherent in our approach. A brief account of related work is included in Section 5 and Section 6 presents some discussion and our conclusions.

2 Configurable Hardware in Memory Hierarchies

We believe that configurable logic can result in significant performance improvement by improving average memory access latencies. Our overriding goal is to minimize the number of cache misses that result in accesses to main memory. Researchers have proposed methods that promise performance improvements of up to 3X by reducing cache misses using full-custom hardware [19]. These methods are rarely included in commercial processors, however, because they do not provide performance improvements for a broad enough set of applications. Our current research shows the potential of these approaches using flexible, configurable hardware instead.

Integrated circuits are expected to soon grow to contain over 100 million transistors. As this growth takes place, we must determine ways to best make use of these additional transistors. Rather than simply devoting increased chip areas to increased cache sizes, our research explores other methods for using the transistors to reduce (or better tolerate) memory access latencies.

In current research projects, configurable logic is typically incorporated into the architecture using an attached processor array model. As shown on the left hand side of Figure 1, an accelerator based on FPGAs and dedicated static RAM (SRAM), is attached to the I/O bus of a conventional host processor. The conventional processor and configurable logic array operate asynchronously. The host supplies control signals and monitors results while the logic array processes data obtained from an external source such as a frame-buffer. The major problem with this model is the high communication latency between processor and configurable logic, due to their physical and logical separation.

As shown on the right hand side of Figure 1, the expected integration of configurable logic on-chip gives us more flexibility not only in its logical placement within the architecture, but also in its expected uses. In addition, on-chip configurable logic has more flexibility in its connectivity to processors and caches. We can now begin considering uses for configurable logic that are infeasible (because of latency and connectivity constraints) with the attached processor model.

2.1 Logical Placement, Connectivity

The logical placement of the configurable logic is driven by its intended memory optimization function. The right hand side of Figure 1 shows two distinct possibilities, and in fact, each of these configurable blocks may expend their transistor budgets on some combination of configurable gates and associated SRAM. (In this figure, the logical positions for configurable logic are indicated by diamonds labelled "C1" and "C2".)

The configurable logic closest to the processor can detect L1 cache misses and manage prefetch buffers, stream buffers, or a victim cache. Similar functions can be performed by the second block of configurable logic, for the L2 cache. In addition, this logic could observe memory accesses between nodes of a dis-

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE