Reiner W. Hartenstein
Herbert Grünbacher (Eds.)

Field-Programmable Logic and Applications

The Roadmap to Reconfigurable Computing

10th International Conference, FPL 2000
Villach, Austria, August 27-30, 2000
Proceedings

Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

Reiner W. Hartenstein
University of Kaiserslautern, Computer Science Department
P. O. Box. 30 49, 67653 Kaiserslautern, Germany
E-mail: hartenst@rhrk.uni-kl.de

Herbert Grünbacher
Carinthia Tech Institute
Richard-Wagner-Str. 19, 9500 Villach, Austria
E-mail: hg@cti.ac.at

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Field programmable logic and applications : the roadmap to reconfigurable computing ; 10th international conference ; proceedings / FPL 2000, Villach, Austria, August 27 - 30, 2000. Reiner W. Hartenstein ; Herbert Grünbacher (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 2000
(Lecture notes in computer science ; Vol. 1896)
ISBN 3-540-67899-9

CR Subject Classification (1998): B.6-7, J.6

ISSN 0302-9743
ISBN 3-540-67899-9 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Steingräber Satztechnik GmbH, Heidelberg
Printed on acid-free paper SPIN 10722573 06/3142 5 4 3 2 1 0

Memory Access Schemes for Configurable Processors

Holger Lange and Andreas Koch
Tech. Univ. Braunschweig (E.I.S.), Gaußstr. 11, D-38106 Braunschweig, Germany
lange,koch@eis.cs.tu-bs.de

Abstract. This work discusses the Memory Architecture for Reconfigurable Computers (MARC), a scalable, device-independent memory interface that supports both irregular (via configurable caches) and regular accesses (via pre-fetching stream buffers). By hiding specifics behind a consistent abstract interface, it is suitable as a target environment for automatic hardware compilation.

1 Introduction

Reconfigurable compute elements can achieve considerable performance gains over standard CPUs [1] [2] [3] [4]. In practice, these configurable elements are often combined with a conventional processor, which provides the control and I/O services that are implemented more efficiently in fixed logic. Recent single-chip architectures following this approach include NAPA [5], GARP [6], OneChip [7], OneChip98 [8], Triscend E5 [9], and Altera Excalibur [10]. Board-level configurable processors either include a dedicated CPU [11] [12] or rely on the host CPU for support [13] [14].

Design tools targeting one of these hybrid systems such as GarpCC [15], Nimble [16] or Napa-C [17] have to deal with software and hardware issues separately as well as with the creation of interfaces between these parts. On the software side, basic services such as I/O and memory management are often provided by an operating system of some kind. This can range from a full-scale general-purpose OS over more specialized real-time embedded OSes down to tiny kernels offering only a limited set of functions tailored to a very specific class of applications. Usually, a suitable OS is either readily available on the target platform, or can be ported to it with relative ease.

This level of support is unfortunately not present on the hardware side of the hybrid computer. Since no standard environment is available for even the most primitive tasks such as efficient memory access or communication with the host, the research and development of new design tools often requires considerable effort to provide a reliable environment into which the newly-created hardware can be embedded. This environment is sometimes called a *wrapper* around the custom datapath. It goes beyond a simple assignment of chip pads to memory pins. Instead, a structure of on-chip busses and access protocols to various resources (e.g., memory, the conventional processor, etc) must be defined and implemented.

In this paper, we present our work on the Memory Architecture for Reconfigurable Computers (MARC). It can act as a "hardware target" for a variety of hybrid compilers, analogously to a software target for conventional compilers. Before describing its specifics, we will justify our design decisions by giving a brief overview of current configurable architectures and showing the custom hardware architectures created by some hybrid compilers.

R.W. Hartenstein and H. Grünbacher (Eds.) FPL 2000, LNCS 1896, pp. 615–625, 2000.

2 Hybrid Processors

Static and reconfigurable compute elements may be combined in many ways. The degree of integration can range from individual reconfigurable function units (e.g., OneChip [7]) to an entirely separate coprocessor attached to a peripheral bus (e.g., SPLASH [4], SPARXIL [18]).

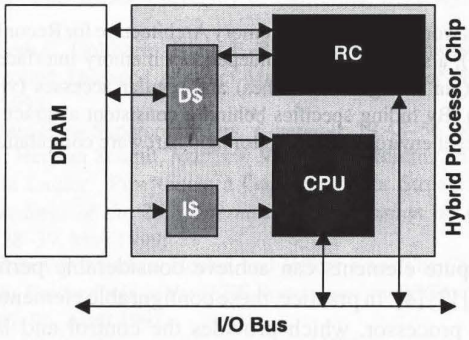

Figure 1. Single-chip hybrid processor

Figure 1 sketches the architecture of a single-chip hybrid processor that combines fixed (CPU) and reconfigurable (RC) compute units behind a common cache (D$). Such an architecture was proposed, e.g., for GARP [6] and NAPA [5]. It offers very high bandwidth, low latency, and cache coherency between the CPU and the RC when accessing the shared DRAM.

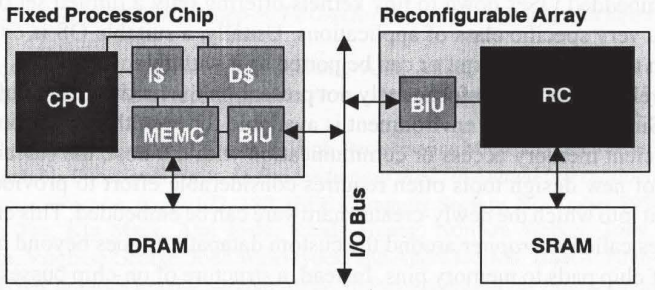

Figure 2. Hybrid processor emulated by multi-chip system

The board-level systems more common today use an architecture similar to Figure 2 . Here, a conventional CPU is attached by a bus interface unit (BIU) to a system-wide I/O bus (e.g., SBus [18] or PCI [11] [12]). Another BIU connects the RC to the I/O bus. Due to the high communication latencies over the I/O bus, the RC is often attached directly to a limited amount of dedicated memory (commonly a few KB to a few MB of

SRAM). In some systems, the RC has access to the main DRAM by using the I/O bus as a master to contact the CPU memory controller (MEMC). With this capability, the CPU and the RC are sharing a logically homogeneous address space: Pointers in the CPU main memory can be freely exchanged between software on the CPU and hardware in the RC.

Operation	Cycles
ZBT SRAM read	4
ZBT SRAM write	4
PCI read	46-47
PCI write	10

Table 1. Data access latencies (single word transfers)

Table 1 shows the latencies measured on [12] for the RC accessing data residing in local Zero-Bus Turnaround (ZBT) SRAM (latched in the FPGA I/O blocks) and in main DRAM (via the PCI bus). In both cases, one word per cycle is transferred after the initial latency.

It is obvious from these numbers that any useful wrapper must be able to deal efficiently with access to high latency memories. This problem, colloquially known as the "memory bottleneck", has already been tackled for conventional processors using memory hierarchies (multiple cache levels) combined with techniques such as pre-fetching and streaming to improve their performance. As we will see later, these approaches are also applicable to reconfigurable systems.

3 Reconfigurable Datapaths

The structure of the compute elements implemented on the RC is defined either manually or by automatic tools. A common architecture [6] [16] [18] is shown in Figure 3 .

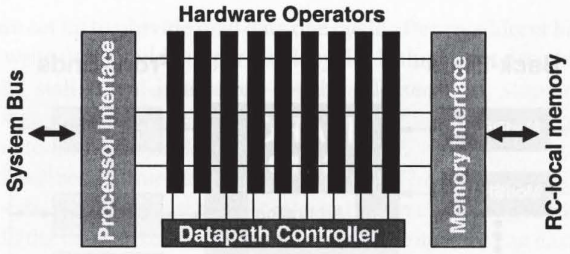

Figure 3. Common RC datapath architecture

The datapath is formed by a number of hardware operators, often created using module generators, which are placed in a regular fashion. While the linear placement shown in the figure is often used in practice, more complicated layouts are of course possible. All hardware operators are connected to a central datapath controller that orchestrates their execution.

In this paper, we focus on the interface blocks attaching the datapath to the rest of the system. They allow communication with the CPU and main memory using the system bus or access to the local RC RAM. The interface blocks themselves are accessed by the datapath using a structure of uni- and bidirectional busses that transfer data, addresses, and control information.

DOCKET ALARM

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.