UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION,

Petitioner

v.

FG SRC LLC,

Patent Owner

CASE NO.: 2020-01449
PATENT NO. 7,149,867

DECLARATION OF AUSTIN M. SCHNELL

Mail Stop **PATENT BOARD**
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450

Alexandria, VA 22313-1450

I, Austin M. Schnell, declare as follows:

1. I am an associate with the law firm of Pillsbury Winthrop Shaw Pittman LLP. I have served in that role since 2020. I have personal knowledge of the matters set forth in this declaration.

2. Attached hereto as SCHN01 is a true and correct copy of Rajesh Gupta, *Architectural Adaptation in AMRM Machines*, Proceedings of the IEEE Computer Society Workshop on VLSI 2000 (IEEE, April 27–28, 2000), 75–79 ("Gupta"). I retrieved a physical copy of the proceedings from the University of Texas library. In addition to Gupta, SCHN01 includes true and correct copies of the front and back covers of the proceedings, the spine and edges of the proceedings, table of contents, introductory materials, and publication information.

3. Attached hereto as SCHN02 is a true and correct copy of Andrew A. Chien et al., *MORPH: A System Architecture for Robust High Performance Using Customization (An NSF 100 TeraOps Point Design Study)*, Proceedings of Frontiers '96 – The Sixth Symposium on the Frontiers of Massively Parallel Computing (IEEE, October 27–31, 1996), 336–345 ("Chien"). I retrieved a physical copy of the proceedings from the University of Texas library. In addition to Chien, SCHN02 includes true and correct copies of the front and back covers of the proceedings, the spine and edges of the proceedings, table of contents, introductory materials, and publication information.

I declare under penalty of perjury that the foregoing is true and correct.

Date: March 31, 2021

Austin M. Schnell

Appendix SCHN01

Proceedings

IEEE Computer Society Workshop on VLSI 2000

System Design for a System-on-Chip Era

27-28 April 2000 Orlando, Florida

Edited by
Asim Smailagic, Robert Brodersen and Hugo De Man

Sponsored by
IEEE Computer Society Technical Committee on VLSI

Proceedings

Sysi

ip Era

Proceedings

IEEE Computer Society Workshop on VLSI 2000

System Design for a System-on-Chip Era

27–28 April 2000

Orlando, Florida

Edited by
Asim Smailagic, Robert Brodersen, and Hugo De Man

Sponsored by the
IEEE Computer Society Technical Committee on VLSI

IEEE COMPUTER SOCIETY

Los Alamitos, California

Washington • Brussels • Tokyo

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

IEEE Computer Society Order Number PR00534
ISBN 0-7695-0534-1
ISBN 0-7695-0536-8(microfiche)
Library of Congress Number 99-069215

Additional copies may be ordered from:

IEEE Computer Society
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: + 1-714-821-8380
Fax: + 1-714-821-4641
E-mail: cs.books@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: + 1-732-981-0060
Fax: + 1-732-981-9667
http://shop.ieee.org/store/
customer-service@ieee.org

IEEE Computer Society
Asia/Pacific Office
Watanabe Bldg., 1-4-2
Minami-Aoyama
Minato-ku, Tokyo 107-0062
JAPAN
Tel: + 81-3-3408-3118
Fax: + 81-3-3408-3553
tokyo.ofc@computer.org

Editorial production by Anne Rawlinson

Cover art production by Joe Daigle/Studio Productions

Printed in the United States of America by The Printing House

IEEE COMPUTER SOCIETY

Table of Contents

System Level Design Examples

Timing Issues in System Design

Software System Design and Design Environment

Analysis and Synthesis of Asynchronous Circuits

Advances in Multiplier Design

Issues in System Design

Message from the General Chairs

Welcome to Orlando, to WVLSI '2000. We hope and trust that those of you attending the workshop find it to be both enjoyable and a productive use of your time. WVLSI has become a regular annual forum for researchers to exchange ideas in the area of VLSI and system level design, in particular.

This workshop has been successful over the past decade due to the many members of the VLSI community who have volunteered their efforts. In particular, we would like to thank the Program Co-chairs Asim Smailagic, Robert Broderson and Hugo De Man for having put together a program of high technical excellence.

Srinivas Katkoori and Vamsi Krishna helped in coordinating the publicity for the workshop and earn our thanks for their excellent job. We appreciate the help from the past General Co-Chairs, Nagarajan Ranganathan and Anantha Chandrakhasan, in organizing the conference. Crucial administrative help came from the members of the IEEE Computer Society, in particular, Anne Marie Kelly, Mary-Kate Rada and Maggie Johnson.

Welcome once again and enjoy the workshop program.

Vijaykrishnan Narayanan
The Pennsylvania State University, USA

Mary Jane Irwin
The Pennsylvania State University, USA

Message from the Technical Program Chairs

It is our distinct pleasure to welcome you to the IEEE Computer Society Annual Workshop on VLSI in Orlando, FL.

This Workshop explores emerging trends and novel concepts in the area of VLSI. The theme of the Workshop is System Design for a System-on-Chip Era. System Level Design has been identified as a dominant research theme for the next decade. System Design has been gaining significance and momentum recently due to the emergence of system-on-a-chip designs. New visionary approaches at the system design level are needed to exploit the great opportunities created by the continuous advances in technology and miniaturization of the semiconductor devices.

System design is converging on a paradigm which includes general purpose commodity chips (i.e. processors, memories, DSP) and full custom mixed analogy and digital application specific integrated circuits (ASICs) integrated via programmable gate arrays on custom printed circuit boards or complete silicon boards, System-on-a-Chip. These hardware systems will be driven by custom, real time software that utilizes the latest software design paradigms (i.e. object oriented languages, client-server architecture, browser interfaces) and wireless communications to provide users with unique functionality. To be effective, these systems must be optimized taking into account a variety of constraints including complexity, power consumption, heat dissipation, mechanical packaging, ergonomics, and design effort. Also, future system design methodologies are an important topic at the Workshop.

We are glad to have a number of leading scientists and distinguished speakers on the workshop program, providing an unique opportunity for the attendees to hear the recent research results in this technical area. It is the face to face meetings with each other that attendees will probably value most, which is why we have tried to maintain a schedule permitting such interactions.

We would like to acknowledge the effort and help from the program committee members, and thank the authors and invited speakers for their contributions to an outstanding technical program. We gratefully acknowledge a diligent work of Anne Rawlinson, of the IEEE Computer Society Press, on the workshop proceedings.

It is our sincere hope that each attendee will benefit greatly from participating in this conference, and will find these proceedings to be a valuable source of information for your future work.

Asim Smailagic
Carnegie Mellon University

Robert Brodersen
University of California at Berkeley

Hugo De Man
IMEC, Belgium

General Chairs

Vijaykrishnan Narayanan
The Pennsylvania State University, USA

Mary Jane Irwin
The Pennsylvania State University, USA

Technical Program Chairs

Asim Smailagic
Carnegie Mellon University, USA

Robert Brodersen
University of California, Berkeley, USA

Hugo De Man
IMEC, Belgium

Program Committee

Gaetano Borriello, *University of Washington, USA*
Don Bouldin, *University of Tennessee, USA*
Francky Catthoor, *IMEC, Belgium*
Anantha Chandrakasan, *Massachusetts Institute of Technology, USA*
Mike Connors, *Intel Corporation, USA*
Rolf Ernst, *University of Braunschweig, Germany*
Georges Gielen, *Katholieke Universiteit Leuven, Belgium*
Mike Gordon, *Cambridge University, UK*
Rajesh Gupta, *University of California, Irvine, USA*
Sorin Huss, *Darmstadt University of Technology, Germany*
Fritz Prinz, *Stanford University, USA*
Teresa Meng, *Stanford University, USA*

Hıdetoshi Onodern, *Kyoto University, Japan*
Jef Van Meerbergen, *Philips, Netherlands*
Pierre Paulin, *SGs-Thomson, Grenoble, France*
Jan Rabaey, *University of California, Berkeley, USA*
Zoran Salcic, *University of Auckland, New Zealand*
Dan Siewiorek, *Carnegie Mellon University, USA*
Mani Srivastava, *UCLA, USA*
Diederik Verkest, *IMEC Belgium*
Vijaykrishnan Narayanan, *The Pennsylvania State University, USA*
Jacob White, *Massachusetts Institute of Technology, USA*
Wayne Wolf, *Princeton University, USA*

Publicity Chairs

Srinivas Katkoori
University of South Florida, Tampa, USA

Vamsi Krishna
HP Labs, USA

Registration Chair

Srinivas Aruru
Intel Corporation, USA

Steering Committee

A. Mukherjee
University of Central Florida, Orlando, USA

D. W. Bouldin
University of Tennessee, Knoxville, USA

N. Ranganathan
University of South Florida, Tampa, USA

P. A. Subramanyam
AT&T Bell Labs, Murray Hill, USA

J. A. B. Fortes
Purdue University, West Lafayette, USA

Architectural Adaptation in AMRM Machines

Rajesh Gupta

Information and Computer Science
University of California, Irvine
Irvine, CA 92697
rgupta@ics.uci.edu

Abstract

Application adaptive *architectures use architectural mechanisms and policies to achieve system level performance goals. The AMRM project at UC Irvine focuses on adaptation of the memory hierarchy and its role in latency and bandwidth management. This paper describes the architectural principles and first implementation of the AMRM machine proof-of-concept prototype.*

1. Introduction

Modern computer system architectures represent design tradeoffs and optimizations involving a large number of variables in a very large design space. Even when successfully implemented for high performance, which is benchmarked against a set of representative applications, the performance optimization is only in an average sense. Indeed, the performance variation across applications and against changing data set even in a given application can easily be by an order of magnitude [1]. In other words, delivered performance can be less than one tenth of the system performance that the underlying hardware is capable of.

A primary reason of this fragility in performance is that rigid architectural choices related to organization of major system blocks (CPU, cache, memory, IO) do not work well across different applications.

Architectural Adaptivity provides an attractive means to ensure robust high performance. Architectural adaptation refers to the capability of a machine to support multiple architectural mechanisms and policies that can be tailored to application and/or data needs [2]. There are a number of places where architectural adaptivity can be used, for instance, in tailoring the interaction of processing with I/O, customization of CPU elements (e.g., splittable ALU resources) etc.

In view of the microelectronic technology trends that emphasize increasing importance of communication at all levels, from network interfaces to on-chip interconnection fabrics, communication represents the focus of our studies in architectural adaptation. In this context, memory system latency and bandwidth issues are key determining factors in performance of high performance machines because these can provide a constant multiplier on the achievable system performance [1]. Further, this multiplier decreases as the memory latency fails to improve as fast as processor clock speeds.

Consider a hypothetical machine with processing elements running at 2 GHz with eight-way superscalar pipelines. Assuming a typical 1 microsecond round-trip latency for a cache miss, this corresponds to about 16K instructions, with an average 30% or 4800 instructions being load/store. For a single-thread execution a miss rates as low as 0.02% reduces computing efficiency by as much as 50%. This points to a need for very low miss rates to ensure that high-throughput CPUs can be kept busy. A similar analysis of the bisection bandwidth concludes that active bandwidth management is required to reduce the need for communication and to increase the number of operations before a communication is necessary.

2. The AMRM Project

The Adaptive Memory Reconfiguration Management, or the AMRM, project at the University of California, Irvine aims to find ways to improve the memory system performance of a computing system. The basic system architecture reflects the view that communication is already critical and getting increasingly so [3], and flexible

0-7695-0534-1/00 $10.00 © 2000 IEEE

interconnects can be used to replace static wires at competitive performance in interconnect dominated microelectronic technologies [4,5,6]. The AMRM machine uses reconfigurable logic blocks integrated with the system core to control policies, interactions, and interconnections of memory to processing [7]. The basic machine architecture supports application-specific cache organization and policies, hardware-assisted blocking, prefetching and dynamic cache structures (such as stream, victim caches, stride prediction and miss history buffers) that optimize the movement and placement of application data through the memory hierarchy. Depending upon the hardware technology used and the support available from the runtime environment this adaptation can be done statically or at run-time. In the following section we describe a specific mechanism for latency management that is shown to provide significant performance boost for the class of applications characterized by frequent accesses to linked data structures scattered in the physical memory. This includes algorithms that operate on sparse matrices and linked trees.

2.1 Adaptation for Latency Management

Latency management refers to techniques for hiding long latencies of memory accesses by useful computation. Most common technique for latency hiding is by prefetching of data to the CPU. Prefetching is combined with smaller and faster (cache) memory elements that attempt to prefetch application context(s) rather than single data elements. The pointer-based accesses to data items in memory hierarchies typically yield poor results because the indirection introduces main memory and memory hierarchy latencies into the innermost computational loop. Techniques such as software prefetching (loop unrolling and hoisting of loads) do not adequately solve the problem, as prefetching at the processor leaves multi-level memory hierarchy latency in the critical path. Purely hardware prefetching [8] is also often ineffective because the address references generated by an application may contain no particular address structure. We use an application-specific prefetching scheme that resides in dedicated hardware at arbitrary levels of the memory hierarchy, in all of them, or to bypass them completely. This hardware performs application-specific prefetching, based on the address ranges of data structures used. When there is a reference to an address inside in this range, the prefetch hardware will prefetch the "next" element pointed to by the current element. The pointer field for the next element can be changed at runtime.

This prefetch hardware is combined, for some applications, with address translation and compaction hardware in the memory controller that works well with data structures that do not quite fit into a single cache line. The address translation is done transparently from the application using hardware assist *translate* in the cache controller and a corresponding hardware assist *gather* in the memory controller. Simulation results using this prefetch hardware show a 10X reduction in read miss rates and 100X reduction in data volume reduction for sparse matrix multiply operations [9].

3. The AMRM System Prototype

While AMRM simulation results continue to provide valuable insights into the space of architectural mechanisms and their effectiveness [7][9][10], a system implementation is needed to bring together different parts of the AMRM project (including compiler and runtime system algorithms to support adaptivity). The AMRM system prototype is divided into two phases. First phase consists of implementation of a board-level prototype; followed by a second phase single-chip implementation of the cache memory system. At the time of this writing, the first phase of the project prototype implementation has recently completed. The rest of this paper describes the system design and implementation of the Phase I prototype and its relationship to the ongoing second phase ASIC prototype.

The AMRM phase I prototype board is designed to serve two purposes. It can simulate a range of memory hierarchies for applications running on a host processor. The board supports configurability of the cache memory via an on-board FPGA-based memory controller. The board is also designed to be used, in future, as a complete system platform, with on-board memory serving as main memory, via a mezzanine card containing the Phase II AMRM ASIC implementation.

Whereas the goal of the AMRM prototype is to build an adaptive cache memory system, in general, the memory hierarchy performance cannot be de-coupled from the processor instruction set architecture, implementation, and compiler implementation. (This is particularly true of the CPU-L1 path that is often pipelined using non-blocking caches.) It would, therefore, be desirable to evaluate any proposed changes to the memory hierarchy for several processor architectures rather than being tied to one specific processor type and its

software. In order to be able to evaluate the effect with different processor architectures as well as to circumvent the implementation difficulties, our Phase I implementation attaches an additional memory hierarchy to a system through a standard peripheral bus. Thus, the board provides a PCI interface that allows a host processor to use the board as a part of its memory hierarchy. Applications running on the host processor are instrumented automatically using the AMRM compiler to use the memory on the AMRM board. Thus direct program execution can proceed on the host processor while the extra memory hierarchy is being exercised.

3.1 AMRM System Goals

One goal of the AMRM prototype system is that it be adaptable to many different memory hierarchy architectures. Another goal of the AMRM system is that it be useful for running real time program execution or even memory simulations. The latter is accomplished by making the AMRM memory available to the user and converting user program to access this memory "directly". The former is accomplished through the use of a sequence of address/command type requests "run" through various memory system configurations. The AMRM system is to be fast enough to support extensive execution or simulation.

A CPU interfaces to the reconfigurable AMRM memory system through the PCI bus. AMRM accepts CPU PCI requests for memory operations, issues them to the attached memory system, and sends back the data for memory read operations as well as memory access time information.

3.2 AMRM prototype architecture

Figure 1 shows the main components of the AMRM prototype board. It consists of a general 3-level memory hierarchy plus support for the AMRM ASIC chip implementing architectural assists with in the CPU-L1 datapath. The host interface is managed by a Motorola PLX 9080 processor. The FPGAs on the board contain controllers for the SRAM, DRAM and L1 cache. A 1 MB SRAM is used for tag and data store for the L1 cache. A total of 512 MB of DRAM is provided to implement part of the cache hierarchy (and also to serve as main memory by reloading the memory controller into the FPGA.)

The board implementation necessarily hard-wires certain parameters of the memory hierarchy. This includes the board's clock. In order to perform detailed and accurate simulation of diverse memory hierarchy configurations at any clock speed, a hardware "virtual" clock has been implemented as part of the performance monitoring hardware. Performance monitoring hardware primarily includes various event counters, which are memory-mapped and readable from the host processor. The "virtual clock" emulates a target system's clock: the clock rate is determined by the target system's memory hierarchy design and technology parameters. For example, the delay for an L1 cache hit, miss fetch etc in terms of virtual clocks can be configured by the host to emulate a given target cache design. Thus, the use of virtual clock allows us to simplify the hardware implementation. For instance, the tag and data stores of L1 cache can be a single RAM while the timing may reflect a design with two separate RAM's.

3.3 Command Interface to the AMRM Board

The memory hierarchy on the AMRM board can be used by an application running on the host processor by writing commands to specific addresses in the PCI address space. Each command consists of a set of four words that specify the operation (e.g., memory read/write, register read/write), the address of the location to access and data in case of a write. For read commands, a read response is generated and data is written into the host's memory.

For debugging purposes and to enable the cache to be flushed by the host, there are commands to access memory banks directly, i.e., without going through the caches. Commands are also available to read/write the status, configuration registers and performance counters.

The onboard command processors reads a command and launches its execution in the AMRM board. Data is read from the cache and sent back to the processor if it hits in the cache. It takes m Virtual Clock cycles. Otherwise it is requested from the next level in the hierarchy. Writes take n virtual clocks. Upon load command completion the data can be written into the system memory for access by the host processor. Both parameters n and m can be programmed under compiler control.

3.4 Virtual Clock System

The virtual clock system consists of a master clock counter, *Vtime*, the virtual clock signal, *Vclk*,

Ready inputs, and associate virtual clock generation logic. The *Ready* input from each major memory hierarchy module specifies that this unit has completed the current virtual clock cycle activities. A new *Vclk* edge/period is generated when all *Ready* inputs reach 1. The *Vtime* can be read out by the host processor to determine the current virtual time. It can also be automatically supplied to the CPU via host memory (as opposed to the AMRM board memory).

Each major unit in the memory hierarchy is designed to generate *Ready* and wait for the *Vclk*, when appropriate. In most cases the designs actually use *AutoReady* counters local to each unit which can be loaded with a programmable number of cycles. An *AutoReady* counter generates *Ready* using the *Vclk* while its output is non-zero. An idle unit not processing any requests also outputs a *Ready* signal every *Vclk*. For instance, Consider the AMRM Read command addressed to the on-board memory hierarchy. It includes a delay (ΔT) from the previous memory access. This delay is used to advance the virtual clock forward before starting the new access. This is accomplished by loading it into the *AutoReady* counter. This allows other memory hierarchy activity to proceed in parallel with CPU computation. For instance, a prefetch unit may be accessing memory during the ΔT-cycle delay.

3.5 AMRM chip functionality

The AMRM board provides for incorporation of an AMRM chip that uses an ASIC implementation of the AMRM cache assist mechanisms. It is positioned between L1 cache and the rest of the system and can be accessed in parallel with the L2 cache. It can thus accept and supply data coming from or going to the L1 cache. For instance, it may contain a write buffer or a prefetch unit to access L2. It also has access to the memory interface and thus can, for instance, prefetch from memory. The AMRM ASIC design is currently in progress. This chip will include a processor core with adaptive memory hierarchy. When plugged into the AMRM board, the ASIC will use onboard DRAM as main memory by simply reconfiguring the memory controller in FPGA1.

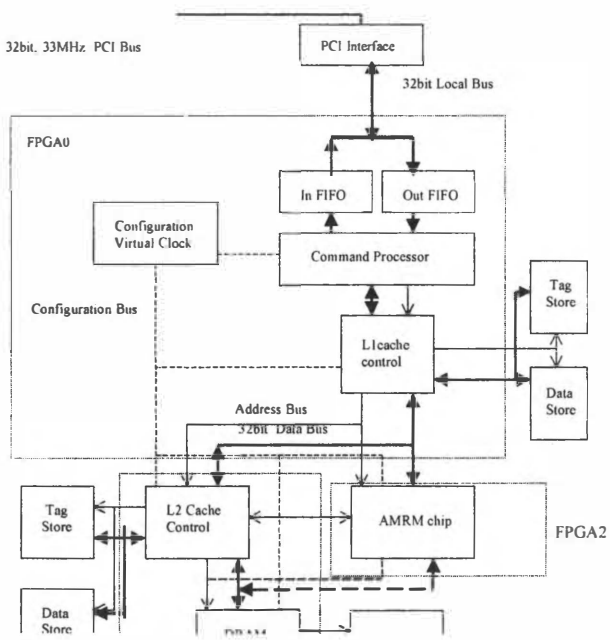


Figure 1: AMRM Phase I Prototype Board

4. Summary

Traditional computer system architectures are designed for best machine performance averaged across applications. Due to the static nature of these architectures, such machines are limited in exploiting application characteristics unless these are common for a large number of applications. Unfortunately, a number of studies have shown that no single machine organization fits all applications, therefore, often the delivered performance is only a small fraction of the peak machine performance. Therefore, we believe that there are significant opportunities for application-specific architectural adaptation. The focus of the AMRM project is on architectural adaptations that close the gap between processor and memory speed by intelligent placement of data through the memory hierarchy. Our current work has demonstrated performance gains due to adaptive cache organizations and cache prefetch assists. In future, we envision adaptive machines that provide a menu of application-specific assists that alter architectural mechanisms and policies in view of the application characteristics. The application developer, with the help of compilation tools, selects appropriate hardware assists to customize the machine to match application needs without having to rewrite the application.

5. Acknowledgement

The AMRM project was conceived through the joint effort of Andrew Chien, Alex Nicolau and Alex Veidenbaum. The team includes Prashant Arora, Chun Chang, Dan Nicolaescu, Rajesh Satapathy, Weiyu Tang, Xiao Mei Ji. The project is sponsored by DARPA under contract DABT63-98-C-0045.

6. References

[1] P. M. Kogge, "Summary of the architecture group findings," In *PetaFlops Architecture Workshop (PAWS)*, April 1996.

[2] A. A. Chien, R. K. Gupta, "MORPH: A System Architecture for Robust High Performance Using Customization," In *Proceedings of the Sixth Symposium on the Frontiers of Massively Parallel Computation (Frontiers'96),* October 1996.

[3] W. Wulf, S. McKee, "Hitting the memory wall: Implication of the obvious," In *Computer Architecture News,* 1995.

[4] C. L. Seitz, "Let's route packets instead of wires," In *Proceedings of the 6th MIT Conference on Advanced Research in VLSI,* W. J. Dally, editor, MIT Press, pp. 133-37, 1996.

[5] W. J. Dally, P. Song, "Design of a self-timed VLSI multicomputer communication controller," In *Proceedings of ICCD,* 1987.

[6] R. K. Gupta, "Analysis of Technology Trends: A Case of Architectural Adaptation in Custom Datapaths," In *Proceedings of SPIE session on Configurable Computing,* Boston, November 1998.

[7] AMRM Website: www.ics.uci.edu/~amrm.

[8] D. F. Zucker, M.J. Flynn, R. B. Lee, "A comparison of hardware prefetching techniques for multimedia benchmarks," *Technical report CSL-TR-95-683,* Computer Systems Laboratory, Stanford University, Stanford University, CA 94305, December 1995.

[9] X. Zhang, *et al,* "Architectural adaptation for memory latency and bandwidth optimization," In *Proceedings of ICCD,* October, 1997.

[10] A. Veidenbaum, *et al,* "Adapting Cache Line Size to Application Behavior," In *Proceedings of International Conference on Supercomputing (ICS),* June 1999.

Author Index

IEEE COMPUTER SOCIETY

Press Activities Board

IEEE Computer Society Publications

The world-renowned IEEE Computer Society publishes, promotes, and distributes a wide variety of authoritative computer science and engineering texts. These books are available from most retail outlets. Visit the Online Catalog, *http://computer.org*, for a list of products.

IEEE Computer Society Proceedings

The IEEE Computer Society also produces and actively promotes the proceedings of more than 141 acclaimed international conferences each year in multimedia formats that include hard and softcover books, CD-ROMs, videos, and on-line publications.

For information on the IEEE Computer Society proceedings, send e-mail to *cs.books@computer.org* or write to Proceedings, IEEE Computer Society, P.O. Box 3014, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314. Telephone +1 714-821-8380. FAX +1 714-761-1784.

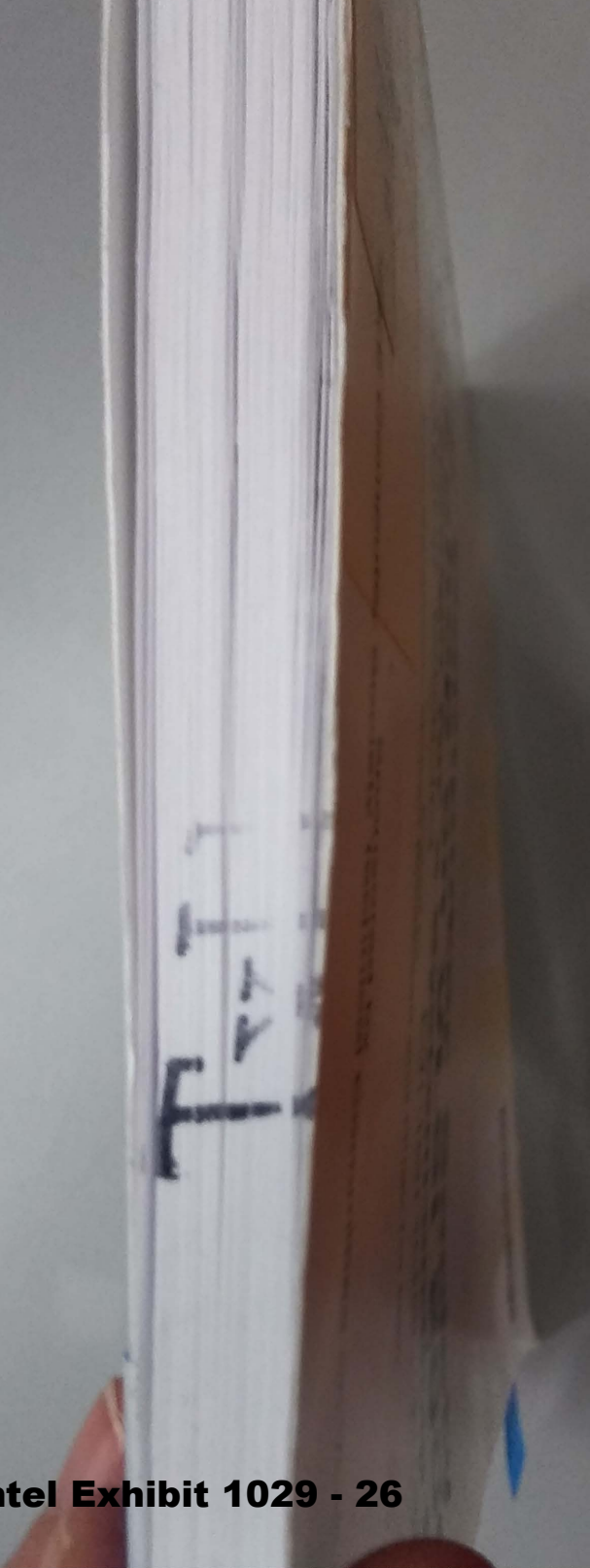
Additional information regarding the Computer Society, conferences and proceedings, CD-ROMs, videos, and books can also be accessed from our web site at *http://computer.org/cspress*

Revised 9 November 1999

Published by the IEEE Computer Society
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314

IEEE Computer Society Order Number PR00534
Library of Congress Number 99-069215
ISBN 0-7695-0534-1

ISBN 0-7695-0534-1
90000
9 780769 505343



TxII

IEEE Computer Society Workshop on VLSI 2000

Appendix SCHN02

Proceedings

Frontiers '96

The Sixth Symposium on The Frontiers of Massively Parallel Computation

October 27 - 31, 1996

Annapolis, Maryland

Sponsored by
IEEE Computer Society

IEEE COMPUTER SOCIETY 50 YEARS OF SERVICE • 1946-1996

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Proceedings

Frontiers '96

nposium on the
ely Parallel Computing

Proceedings

Frontiers '96

The Sixth Symposium on the Frontiers of Massively Parallel Computing

Proceedings

Frontiers '96

The Sixth Symposium on the Frontiers of Massively Parallel Computing

October 27–31, 1996
Annapolis, Maryland

Sponsored by

IEEE Computer Society

In cooperation with

NASA Goddard Space Flight Center
USRA/CESDIS

IEEE Computer Society Press
Los Alamitos, California

Washington • Brussels • Tokyo

IEEE Computer Society Press
10662 Los Vaqueros Circle
P.O.Box 3014
Los Alamitos, CA 90720-1264

IEEE Computer Society Press Order Number PR07551
IEEE Order Plan Catalog Number 96TB100062
ISBN 0-8186-7551-9
Microfiche ISBN 0-8186-7553-5
ISSN 1088-4955

Additional copies may be ordered from:

IEEE Computer Society Press
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: +1-714-821-8380
Fax: +1-714-821-4641
Email: cs.books@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: +1-908-981-1393
Fax: +1-908-981-9667
misc.custserv@computer.org

IEEE Computer Society
13, Avenue de l'Aquilon
B-1200 Brussels
BELGIUM
Tel: +32-2-770-2198
Fax: +32-2-770-8505
euro.ofc@computr.org

IEEE Computer Society
Ooshima Building
2-19-1 Minami-Aoyama
Minato-ku, Tokyo 107
JAPAN
Tel: +81-3-3408-3118
Fax: +81-3-3408-3553
tokyo.ofc@computer.org

Editorial production by Penny Storms
Cover by Kerry Bedford and Alex Torres
Printed in the United States of America by KNI, Inc.

The Institute of Electrical and Electronics Engineers, Inc.

Contents

Message from the General Chair

I wish to welcome you to the Frontiers'96 conference and to beautiful Annapolis, Maryland as well. I am pleased to serve as this year's Conference Chair at an exciting time when our community is going through a significant transition in direction. The Frontiers series of conferences is among the few in the high performance computing arena that has retained its own special focus in spite of the rapidly growing array of conferences in this general area. Frontiers provides a unique forum for representing research directed towards extending the extremes of computing performance. As a result, it is often the first conference at which new concepts in technology, architecture, software applications, and algorithms are presented to the community at large. And this year promises to continue the tradition with a strong emphasis on future systems concepts.

In accordance with direction from the Steering Committee under the leadership of its Chair, Mike Hord, this year's Frontiers conference has as one of its driving themes, "Petaflops Computing." A series of activities has been organized to explore the regime and implications of computing at thousands of times the performance of today's fastest systems. The second Petaflops Frontier workshop (TPF-2) will be held again at Frontiers'96, this time expanded to a day and a half and chaired by George Lake of the University of Washington. In coordination with the National Science Foundation, researchers representing 8 teams sponsored by NSF (with additional support by DARPA and NASA) to investigate architectural approaches to achieving Petaflops will present their findings at this conference. They will contribute to the TPF-2 program and will present their findings in two formal sessions. Their papers can be found within this Proceedings. Finally, a panel session, chaired by David Bailey of the Ames Research Center will open the topic to critical discussion from both panelists and participants.

A second workshop, chaired by Jose Fortes of Purdue University, will focus on the important directions in domain specific computing systems including special purpose, reconfigurable, embedded, systolic arrays, and other types of systems structured to optimize operation for a given class of functions. A second panel organized by Rick Stevens of the Argonne National Laboratory will explore future directions in system software. On this, the tenth anniversary of the Frontiers conference series, it is with some pride that we host a ceremony during which the Smithsonian Institution will formally accept for its collection the original MPP, the proof-of-concept SIMD computer that also launched the first Frontiers conference. The conference is pleased to host keynote talks of exceptional interest to this community on topics including: the DOE ASCI program and its Teraflops computer, the IBM chess playing computer, the HP-Convex scalable shared memory SPP-2000, and future directions for high performance computing in the trans-teraflops era.

I must thank the program chair, Peter Kogge from the University of Notre Dame, and his program committee for assembling an excellent program comprising a most interesting collection of papers from across the community. Finally, I must thank all those who have worked diligently to make Frontiers'96 among the best of this series of conferences. Please join with me and our colleagues to engage in this most interesting and thought provoking symposium.

Thomas Sterling
JPL/California Institute of Technology

Message from the Program Chair

This year's symposium on the Frontiers of Massively Parallel Computation should definitely live up to its billing as "a forum for exploring the...outer boundaries of effective high performance computing." We received 78 submissions from all over the globe, covering all edges of the computing spectrum. From this we accepted 34, about the same percentage as in prior years. Many of the individual decisions were particularly difficult this year, with a great many of the papers representing truly fine work. In the end, we chose a set that we believed presented both a wide ranging and thought provoking set of views that balanced architecture, applications, and systems.

To this base we added eight papers focused on scaling the next true frontier in computing, namely finding combinations of technology, architecture, and software to apply significant fractions of a petaflop against real applications. Each of these papers originates with one of the studies funded as part of the NSF "100 teraflop Point Design" effort, and represents the result of a rigorous review process in its own right.

These papers were then leavened with an outstanding set of invited speakers, panel sessions, and workshops.

In conclusion, I would like to say thank you to all those who participated in Frontiers'96. True leadership came from the top with the Program General Chair, Thomas Sterling, now of Cal Tech and JPL. Without his energy and work, especially during unexpected events such as the government shutdown at critical times in the conference's early months, this effort could not have succeeded. This hard work was continued by the Program vice chairs, Ken Batcher, Rick Stevens, and Geoffrey Fox, and all those on the program committee who did such an superb effort in handling the review process in such a professional and timely fashion. Thanks carry over even further to the actual referees who did an absolutely outstanding job of returning high quality and thoughtful reviews. I truly have never before been in a conference paper selection meeting where we could really focus on organizing a truly intellectually satisfying set of presentations. Finally, the real unsung heroes are Michele O'Connell and Georgia Flanagan, whose day-in and day-out management of the activities, all handled with a touch of humor, helped make this conference possible.

Peter M. Kogge
University of Notre Dame

Conference Committee

Symposium General Chair
Thomas Sterling, *JPL/California Institute of Technology*

Program Chair
Peter Kogge, *University of Notre Dame*

Program Vice Chairs
Architectures:
Ken Batcher, *Kent State University*
Applications & Algorithms:
Rick Stevens, *Argonne National Laboratory Systems*
Software:
Geoffrey Fox, *Syracuse University*

Finance Chair
James Fischer, *NASA GSFC*

Local Arrangement Chair
Georgia Flanagan, *USRA/CESDIS*

Publicity Chair
Lawrence D. Picha, *Boeing Information Services, Inc.*

Committee Coordination Chair
Michele L. O'Connell, *Boeing Information Services, Inc.*

Steering Committee Chair
Mike Hord, *Hughes Information Technology Systems*

Steering Committee
Larry Davis, *University of Maryland*
Judy Devaney, *National Institute of Standards and Technology*
Jack Dongarra, *Oak Ridge National Laboratory*
John Dorband, *NASA Goddard Space Flight Center*
Milt Halem, *NASA Goddard Space Flight Center*
Paul Messina, *California Institute of Technology*
Merrill Patrick, *National Science Foundation*
David Schaefer, *George Mason University*
Paul Schneck, *MITRE Corporation*
H.J. Siegel, *Purdue University*
Francis Sullivan, *Center of Computing Sciences*
Pearl Wang, *George Mason University*

Program Committee

Ken Batcher, *Kent State University*
Donna Bergmark, *Cornell Theory Center*
Fran Berman, *University of California, San Diego*
Randy Bramley, *Indiana University*
Jay Brockman, *University of Notre Dame*
Bill Carlson, *Supercomputing Research Center*
Dan Z. Chen, *University of Notre Dame*
Alok Choudhary, *Syracuse University*
Walter Ermler, *Department of Energy*
Charbel Farhat, *University of Colorado*
Geoffrey Fox, *Syracuse University*
Kanad Ghose, *Binghamton University*
Andrew Grimshaw, *University of Virginia*
Jim Hendler, *University of Maryland*
Joseph Ja'Ja', *University of Maryland*
Carl Kesselman, *California Institute of Technology*
Peter Kogge, *University of Notre Dame*
Walter Ligon, *Clemson University*
Andrew Lumsdaine, *University of Notre Dame*
Rogert Martino, *National Institute of Health*
Matthew O'Keefe, *University of Minnesota*
Constantine Polychronopolous, *University of Illinois*
Donna Quammen, *George Mason University*
Sanjay Ranka, *University of Florida*
Dan Reed, *University of Illinois*
Joel Saltz, *University of Maryland*
David Schaefer, *George Mason University*
Sanjeev Setia, *George Mason University*
Edwin Sha, *University of Notre Dame*
H.J. Siegel, *Purdue University*
Thomas Sterling, *JPL/California Institute of Technology*
Ken Stevens, *NASA Ames Research Center*
Rick Stevens, *Argonne National Laboratory*
Tarek El-Ghazawi, *George Washington University*
Joseph Torrella, *University of Illinois*
Alex Veidenbaum, *University of Illinois*
Pearl Wang, *George Mason University*
Elizabeth Williams, *Supercomputing Research Center*
Steve Zalasak, *NASA Goddard Space Flight Center*
Mary Zosel, *Lawrence Livermore National Laboratory*

Referees

Tarek S. Abdelrahman, *University of Toronto*
Anurag Acharya, *Univ. of Maryland College Park*
George Adams III, *Purdue University*
Gagan Agrawal, *Univ. of Maryland College Park*
Jennifer Anderson, *U of Illinois, Urbana Champaign*
Johnnie Baker, *Kent State University*
Kenneth Batcher, *Kent State University*
Peter Beckman, *Indiana University*
Donna Bergmark, *Cornell University*
Rajesh Bordawekar, *Northwestern University*
Michael Bridgland, *Institute for Defense Analyses*
Eugene Brooks, *Lawrence Livermore National Lab*
William Carlson, *Institute for Defense Analyses*
Prachya Chalermwat, *George Washington Univ.*
Joy Chantraporchai, *University of Notre Dame*
Steve Chapin, *Kent State University*
Alok Choudhary, *Northwestern University*
Neil Coletti, *Institute for Defense Analyses*
John Conroy, *Institute for Defense Analyses*
Ovidiu Daescu, *University of Notre Dame*
Rosalinda de Fainchtein, *NASA Goddard Space Flight Ctr*
Luiz De Rose, *Univ. of Illinois, Urbana Champaign*
Jose Delgado-Frias, *SUNY, Binghamton*
H. Dietz, *University of California, San Diego*
Andris Dimits, *Lawrence Livermore National Lab*
Val Donaldson, *University of California, San Diego*
John Dorband, *NASA Goddard Space Flight Center*
Jesse Draper, *Institute for Defense Analyses*
Richard Draper, *Institute for Defense Analyses*
Richard Eckert, *SUNY, Binghamton*
Guy Edjlali, *University of Maryland College Park*
Tarek El-Ghazawi, *George Washington University*
Hoda El-Sayed, *George Washington University*
Robert Falgout, *Lawrence Livermore National Lab*
Ian Foster, *Argonne National Labs*
Geoffrey Fox, *Syracuse University*
Kyle Gallivan, *U of Illinois, Urbana Champaign*
Kanad Ghose, *SUNY, Binghamton*
Robert Glamm, *University of Minnesota*
Sanjay Goil, *Northwestern University*
Ian Harris, *University of California, San Diego*
Xiaobo (Sharon) Hu, *University of Notre Dame*
Junbin Huang, *University of Notre Dame*
Hsi-Ren Hung, *University of Notre Dame*
Costin Iancu, *University of Notre Dame*
Mahmut Kandimir, *Northwestern University*
Kevin Klenk, *University of Notre Dame*
Alice Koniges, *Lawrence Livermore National Lab*
David Koufaty, *U. of Illinois, Urbana Champaign*
Steven Kratzer, *Institute for Defense Analyses*
Leslie Lander, *SUNY, Binghamton*
Chenhua Lang, *University of Notre Dame*
Asish Law, *Ohio State University*
Walt Ligon, *Clemson University*
Wen-Yen Lin, *University of California, San Diego*
Tom Loos, *Indiana University*
Steven Louis, *Lawrence Livermore National Lab*
Peter MacNeice, *Hughes, STX*
Maria-Cristina Marinescu, *Univ. of Notre Dame*
Rami Melhem, *Pittsburgh University*
J. Mellor-Crummey, *Univ. of California, San Diego*
Clark Mobarry, *NASA Goddard Space Flight Center*
Bongki Moon, *Univ. of Maryland, College Park*
Lakkshmi Narayanaswamy, *Univ. of Notre Dame*
Sorin Nastea, *George Mason University*
Lionel Ni, *Michigan State University*
Lisa Nicklas, *George Mason University*
Matthew O'Keefe, *University of Minnesota*
Kevin Olson, *NASA Goddard Space Flight Center*
Scott Pakin, *Univ. of Illinois, Urbana Champaign*
Hassan Peyravi, *Kent State University*
Constantine Polychronopoulos, *Univ. of Illinois, Urbana Champaign*
Eleftherios Polychronopoulos, *University of Patras*
Jerry Potter, *Kent State University*
Chalermwat Prachya, *George Washington Univ.*
Dan Pryor, *Institute for Defense Analyses*
Sanjay Ranka, *University of Florida*
Craig Reese, *Institute for Defense Analyses*
Dana Richards, *George Mason University*
Robert Ross, *Clemson University*
Thomas Ruwart, *University of Minnesota*
James Schwarzmeier, *Cray Research*
Mark Seager, *Lawrence Livermore National Lab*
Shamik Sharma, *Univ. of Maryland, College Park*
Mike Sheliga, *University of Notre Dame*

Steven Smith, *Lawrence Livermore National Lab*
Dan Stanzione, *Clemson University*
Ken Stevens, *NASA Ames Research Center*
Q. Stout, *University of California, San Diego*
Alan Sussman, *Univ. of Maryland, College Park*
Daniel Tabak, *George Mason University*
Yi Tian, *Notre Dame University*
Mitch Theys, *Purdue University*
Sissades Tongsima, *University of Notre Dame*
Thomas Turnbull, *Institute for Defense Analyses*
Manuel Ujaldon, *Univ. of Maryland, College Park*
Rao Vemuri, *Lawrence Livermore National Lab*
Deepa Viswanathan, *Indiana University*
Xiaoming Wang, *University of Notre Dame*
Elizabeth White, *George Mason University*
Elizabeth Williams, *Institute for Defense Analyses*
Deyun Wu, *University of Notre Dame*
Qun Xu, *University of Notre Dame*
Alan Yoder, *University of Notre Dame*
Zhihong Yu, *University of Notre Dame*
Steve Zalesak, *NASA Goddard Space Flight Center*
Francis Zane, *University of California, San Diego*
Mary Zosel, *Lawrence Livermore National Lab*
Victor Zyuban, *University of Notre Dame*

MORPH: A System Architecture for Robust High Performance Using Customization

(An NSF 100 TeraOps Point Design Study)

Andrew A. Chien Rajesh K. Gupta

Department of Computer Science
University of Illinois
Urbana, Illinois 61801
{achien,rgupta}@cs.uiuc.edu

Abstract

Achieving 100 TeraOps performance within a ten-year horizon will require massively-parallel architectures that exploit both commodity software and hardware technology for cost efficiency. Increasing clock rates and system diameter in clock periods will make efficient management of communication and coordination increasingly critical. Configurable logic presents a unique opportunity to customize bindings, mechanisms, and policies which comprise the interaction of processing, memory, I/O and communication resources. This programming flexibility, or **customizability**, *can provide the key to achieving robust high performance.*

The **M**ultiprocess**O**r *with* **R**econfigurable **P**arallel **H**ardware *(MORPH) uses reconfigurable logic blocks integrated with the system core to control policies, interactions, and interconnections. This integrated configurability can improve the performance of local memory hierarchy, increase the efficiency of interprocessor coordination, or better utilize the network bisection of the machine. MORPH provides a framework for exploring such integrated application-specific customizability. Rather than complicate the situation, MORPH's configurability supports component software and interoperabilty frameworks, allowing direct support for application-specified patterns, objects, and structures. This paper reports the motivation and initial design of the MORPH system.*

1 Introduction

Increasing reliance on computational techniques for scientific inquiry, complex systems design, faster than real life simulations, and higher fidelity human computer interaction continue to drive the need for ever higher performance computing systems. Despite rapid progress in basic device technology [1], and even in uniprocessor computing technology [2], these applications demand systems scalable in every aspect: processing, memory, I/O, and particularly communication. In addition to advances in raw processing power, we must also achieve dramatic improvements in system usability. Current day scalable systems are still quite difficult to program, and in many cases effectively precluding use of the most sophisticated (and most efficient) algorithms. Even when successfully used, most systems exhibit substantial performance fragility due to rigid architectural choices that do not work well across different applications.

Based on technology projections for the 2007 design window for the NSF point design studies, our analyses indicate that increases in communication cost relative to computation (gate speeds) make configurable logic practical in an ever broader range of the system. The benefit of configurable logic is that it can be used to customize the machine's behavior to better match that required by the application – in essence a machine can be tuned for each application with little or no performance penalty for this generality. While a broad variety of such architectures are possible, MORPH is a design point which explores the potential of integrating configurability deep into the system core.

Because technology trends continue to increase the importance of communication, the MORPH architecture focuses on exploiting configurability to manage locality, communication, and coordination. In particular, the MORPH design study is exploring improved efficiency and scalability by exploring novel mechanisms for binding and mechanisms which comprise the interaction of processing, memory, I/O, and communication resources. Other innovations explore flexible hardware granularity (e.g. mechanisms and association with processors and memory) and memory system management (e.g. cache coherence, prefetching,

and other data management policies).

Because a wealth of studies indicate that no fixed policies (or even wiring configurations) are optimal, MORPH seeks to exploit application structure and behavior to adapt for efficient execution. Traditionally in high performance computing systems, this information has been provided by optimizing compilers (vectorizing or parallelizing compilers). However, as the use of component software and interoperability frameworks proliferates, we expect such analysis to become increasingly difficult. Therefore, we are exploring a broad range of techniques to identify opportunities for customization based on aggressive compiler analysis, user annotations, profiling, and even on-the-fly monitoring and adaptation.

Because the field of configurable computing is still in its nascence, the range of possible architectures has only begun to be explored. The primary innovations of MORPH are to focus on integration of configurability into the system core, and exploitation of opportunities to optimize communication and coordination. In the remainder of this paper, we sketch the MORPH architecture. Since the project is in the early stages of design, many of the detailed design issues are intentionally left open. Up to-date information on the project and results is available on the Web site: "http://www-csag.cs.uiuc.edu/projects/morph.html."

The remainder of paper is organized as follows. In the following Section 2 we present the technology trends and parameters underlying the design of MORPH. Section 3 presents the overall hardware organization of MORPH, while the software architecture is presented in the following Section 4. Section 5 describes our evaluation environment and examples of customizability used by MORPH machine to achieve high performance. We summarize design and open issues in Section 6.

2 Key Technology Trends

The MORPH design is targeted for construction in 2007, based on the commodity technology that will be available. Because these technology extrapolations are critical underpinnings for the design, we discuss them in some detail. The base processor, memory and packaging technology form the fundamental global system constraints. Advances in programmable logic and computer-aided design will make per-application configuration not only feasible but desirable, and the widespread use of component software will make flexible execution engines desirable for achieving high performance.

Processor, Memory, and Packaging Technology The continued advance of Si technology will produce remarkable processors and memory chips, and areal bonding and multi-chip carriers will provide significant improvements in interchip wiring. However, communication will remain critical in achieving high performance.

Projections from Semiconductor Industry Association (SIA) [1] for a decade hence indicate advanced processes using 0.1 μ feature size. However at the deep sub-micron level, feature size as measured by transistor channel length is increasingly irrelevant for velocity-saturated carrier transport [3]. Both logic density and speed are dominated by the interconnect density. Pitch for the finest interconnect is projected at 0.4-0.6 μ. On logic devices, average interconnect lengths are anywhere from 1,000x to 10,000x the pitch, that is, up to 6 mm of intra-chip interconnect delay. This limits the on-chip clock periods to $\approx$ 1 nanosecond, implying that the system performance is going to be increasingly dominated by interconnect delay. With multiple issue and multiple processors on a chip, for example, four 8-way super-scalar on-chip CPUs, processor chips are anticipated to achieve 32 billion instructions per second (BIPS) at 1 GHz clock rates. Memory integration will continue its four-fold increase in density each 3 years, achieving 16 Gbit DRAM and 4Gbit SRAM chips [4].

At a voltage level of 1200-1500 mV, per chip power consumption is expected to be limited to below 200 Watts. Total system power is expected to be 163 Kilo Watts for the MORPH 100 TeraOp configuration, including a 10% loss in power distribution and management. With packaging advances, using reduced on-chip solder bumps are expected to increase I/O's by 10x to approximately 5000, with 90% usable as signal pins. Assuming one word of communication every 100 operations requires signaling rates of 30-40 GHz, even one word every 1,000 operations, 3-4 GHz, well in excess of the SIA's 375 Mhz projected off-chip signaling rate [4], indicating communication is again a critical issue.

Programmable Logic and Computer-Aided Design (CAD) Advances in programmable logic will make their use viable in a much broader range of system elements. In addition, the maturation of CAD technology will allow the flexible exploitation of configurable resources to customize for individual programs.

Programmable logic densities and speed are increasing in similar fashion to SRAM, approximately four times for every three years. The routability and gate utilization in reprogrammable devices will continue to increase at a rapid rate of 5-15% per year due to

better algorithms, CAD tools, and additional routing resources. State of the art programmable logic devices, implemented in a 0.35 μ process technology, provide 100,000 gates and achieve propagation delays of less than 5 ns. As an alternative organization, current FPGA devices also offer up to 40 Kbits of multifunctional memory in addition to over 60,000 usable gates in implementation of specialized hardware functions such as arithmetic or DSP functions. Unlike standard SRAM parts, the embedded memory can be used as multiple-ported SRAM or FIFO providing much greater flexibility in system organization. Current efforts in FPGA design and architecture show significant improvements in the efficiency of the on-chip memory blocks. This trend in memory efficiency and utilization is expected to continue and close the gap between FPGA and SRAM densities and up to 10 Mbits of multi-functional embedded memory would be available in addition to the logic blocks in single-chip reprogrammable devices. In the ten-year period, programmable logic integration levels are expected to reach 2 million gates.

MORPH's design exploits small blocks of reprogrammable interconnect logic to achieve application customizability rather than application-specific functional units or compute elements. Perhaps the most compelling reason for the incorporation of (small) reprogrammable logic blocks even in custom processors has to do with the comparatively low and decreasing delay penalty for reprogrammable logic blocks compared to medium and long interconnects delays. As transistor switching delays scale down to 10 ps range (factor of 1/10 from current delays), the interconnect delays scale down much more slowly. This is because the local interconnect length scales down with pitch (1/3), while the global interconnect length actually increases with increasing die size [5]. As a result, the marginal costs of adding switching logic in the interconnect decreases tremendously, and may even become necessary to provide signal "repeaters" for moderate length interconnects. We are already beginning to see the prevalent use of SRAM-based reprogrammable logic (FPGA and CPLD) parts being used in final product designs. These trends in technology transition to field reprogrammable parts is expected to continue with increasing time-to-market pressures.

Coupled with the advances in technology, the advances in CAD tools and algorithms are beginning to have an impact on how designs are done today. With the emphasis on system models in hardware description languages (HDLs) such as Verilog and VHDL, the process of hardware design is increasingly a language-level activity, supported by compilation and synthesis tools [6]. These tools are beginning to support a variety of design constraints, on performance, size, power, and even the pin-outs. Locking I/O maps to ensure that physical design remains unchanged while logical connections are modified based on applications will soon be a common feature to allow programmable logic to be embedded in key modules of a system and provide on-line programmability to change hardware functionality. Tools for distributed hardware control synthesis to allow dynamic binding of hardware resources [7], and synthesis of protocols to low latency hardware [9, 10, 8] have been successfully demonstrated. With these CAD and synthesis capabilities, embedded programmable logic can be inserted into the key parts of systems, and used to alter behavior dramatically with modest performance overhead.

The architectural implications of these technology trends are enormous: the underlying assumption that reprogrammability is expensive is already beginning to be challenged. As gate switching continues to scale well below 100 ps range (today), local decision making would cost significantly less than the cost of sending information. Such changes will pave the way for a new class of system architectures that exploit flexibility to deliver robust, high performance to applications.

Component Software The exploding complexity of application software is driving increased used of component software (libraries, toolkits, shared abstractions) and interoperability frameworks (to glue components together). These application structures leverage programmer effort, but also imply that programs will increasingly be written in terms of user abstractions (e.g. objects) rather than machine abstractions (registers, cache lines, memory locations).

Over the past ten years, there has been an accelerating trend towards component software, enabling the construction of larger, more complex applications. For example, graphical user interface programmers are leveraged by large graphical user interface libraries such as X Windows or MS Windows '95. Likewise, scientific programmers are increasingly leveraged by libraries such as LAPACK [11, 12], or increasingly domain or application specific libraries such as A++/P++, AMR++, POOMA, LPARX++/KeLP [13, 14, 15] whose millions of lines of code can not only dramatically increase programmer productivity, but in some cases, the sophistication of the libraries can actually deliver higher levels of performance. This phenomenon is occurring in many domains, programming languages, and system contexts because component software allows application programmers to focus on the critical new problems, not solving or reimple-

menting solutions to the same old ones. Experts expect this trend to continue [16], with useful libraries proliferating and increasing in complexity; thereby supporting increased application complexity and programmer productivity.

Beyond library-based sharing in a single program, the use of coordination-interoperability frameworks is having a major impact on application structure and the ease of building large complex applications. Interoperability frameworks such as OLE, CORBA, SOM, etc. [17, 18] define standard interfaces for modules, making it possible to build modular software and compose it with little knowledge of the internal software structure. Thus, large complex programs will be composed of heterogeneous applications including computation, visualization, persistent storage, and even on-line interaction. In future, we expect widespread used of coordination/interoperability frameworks to build complex applications from variegated individual programs. This means that efforts and tools for optimizing applications that must optimize individual programs, coordination frameworks, and entire applications are all essential.

The increasing complexity in software demands a hardware structure which can be easily exploited. However, this need for easily accessible performance comes at a time when hardware technology trends place an increasing importance on issues such as locality, partitioning, and mapping of computations – managing communication. Our solution is a machine architecture which leverages a small amount of programmable logic in several key places to implement a flexible hardware composition structure. This flexibility allows the machine to be configured as convenient or to optimize machine communication, supporting customizability by software and high performance.

3 System Architecture & Organization

The basic architecture of MORPH reflects the observations that in the 2007 technology window, (1) communication is already critical and getting increasingly so [19], and (2) flexible interconnects can be used to replace static wires at competitive performance [20, 21, 22, 23]. The key elements of the MORPH architecture include processing elements and memory elements embedded in a scalable interconnect. The scalable interconnect flexibly connects all parts of the system with fast packet routing, efficiently exploiting the wiring resources provided by the system packaging [21, 22, 20, 23]. The hardware structure allows adaptation of data transport, coordination, association (for granularity), and efficient computation. As an example of its flexibility, MORPH could be used to implement

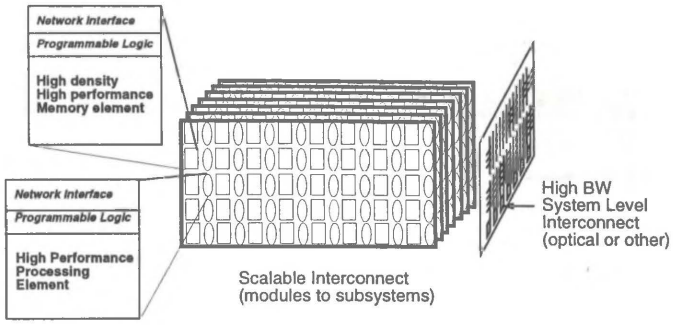


Figure 1: A Flexible 100 TeraOp Architecture

either a cache-coherent machine, a non-cache coherent machine, or even clusters of cache coherent machines connected by put/get or message passing. Varying the mix of processing and memory elements supports a wide range of machine configurations and balances. Examples of other possible changes include changes in cache block size, branch predictors, or prefetch policies.

Our proposed configuration uses 32 BIPS processors, and 8192 processing nodes. The physical memory configuration is driven primarily by cost and packaging (power budgeting) factors. A cost balanced system (based on today's processor to memory price ratios and SIA predictions) would be approximately two memory chips for every processor, or approximately 4 gigabytes/processor. Each processing node consists of 8-16 KB of L1 cache and 128 MB of L2 cache that can be configured as private or shared among on-chip CPUs. Using MCM and areal interconnect technology, our system could be integrated with $\approx$ 30 nodes per card, and with 20 cards per rack, the core of the system would fit in eight racks with room needed for power supplies, I/O cooling fans etc. The communication bandwidth is limited by the wiring of the packages to about 90 GB/sec for a pin-out of 1800 usable MCM pins. The bisection bandwidth would be in the range of 10 TB/sec.

3.1 Architectural Adaptation in MORPH

The critical configurabilities or adaptation flexibilities that this architecture provides include:

- control over computing node granularity (processor-memory association)
- interleaving (address-physical memory element mapping)
- cache policies (consistency model, coherence protocol, object method protocols)

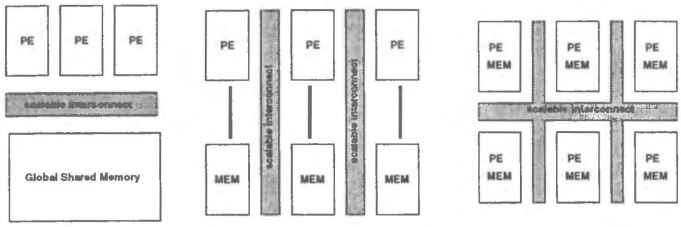


Figure 2: A wide range of logical machine organizations (Address Space and Cache coherence) can be configured.

- cache organization (block size or objects)
- behavior monitoring and adaptation

Depending upon application and runtime environment, customization can be done statically or at run-time.

MORPH's flexible architecture subsumes both the processor-in-memory (PIM) and scalable shared memory approaches. Based on the experience of several "PIM"-like systems [24, 25, 26, 27, 28], there is evidence that PIM organizations represent significant programming challenges, particularly for irregular applications. We believe that the use of more traditional processor-memory structures will yield a machine with more accessible performance than an organization in which processors are accessing primarily their local on-chip memory. In addition, by adding a small amount of programmable logic to the memory units, we can yield much of the benefit of having computational elements within the memory. A shared memory approach represents the opposite extreme, with advantages in programmability, but with questions about scalability.

Since communication is the critical issue, we plan to explore novel techniques for communication. One possibility is to exploit optical arrays and free space or waveguide based interconnects. Since ultra high-speed electrical connections in the tens of Gbps range are limited to a few centimeters, serial links driven by smart pixel arrays could be used to support system backplane connections. Through smart pixel arrays any degree-K interconnection network can be embedded into the backplane. This includes linear arrays, 2D and 3D meshes, toroids, hypercubes, dilated crossbars, orthogonal crossbars, Knockout, CrossOut and shuffle-based networks [29].

While the proposed architecture can subsume a range of traditional parallel machine organizations as shown in Figure 2, the primary use of configurability is to enable customization of mechanism for higher performance. We outline some of the major optimization types below.

- **Low Latency Communication:**

The MORPH architecture can optimize for low-latency communication by adapting the number of memory elements associated with each processing element (optimal PE granularity), configuring the physical I/O resource to match the applications needs (local memory hierarchy, global network) and by adding special hardware structures [30, 31] such as fast barrier or broadcast support for machine subsets or the entire machine, to optimize performance. For example, experience over the last ten years demonstrates that intraprocessor communication mechanisms (data shared through the cache) are much more efficient than even the best interprocessor mechanisms. When machine configuration granularity matches the application, extremely high performance can result. The programmable logic on both processor elements and memory elements allows us to dynamically associate memories with processor chips, changing the node granularity at application set up. In another example, some applications, benefit greatly from low-latency barriers, or high speed broadcast or multicast. Such structures are easily implementable with this configurable hardware.

- **Minimizing Communication:**

Possibilities include custom caching policies (e.g. adaptive invalidate-update, custom block sizes, and even more complex schemes [32]), object-based coherence (e.g. program semantics-based policies and data movement), custom prefetching FSM's (derived from program analysis, or dynamic selection based on effectiveness), and can drive all of these choices with detailed performance data capture, via customizable hardware. For example, false sharing can be eliminated by adapting policies (subblocking) for particular cache blocks. In other examples, object consistency semantics (e.g. write-once policy) can be used to reduce protocol overhead, object sizes could be used to eliminate multiple cache misses for a single object reference, virtual function data requirements could be used to fetch only the needed parts of an object which reduces data movement requirements, and custom encodings can be used for special datatypes, reducing the number of bits that must be transferred.

- **Resource Load Balance:**

A critical issue for scalable systems, particularly with the increasing prominence of irregular and adaptive methods, is efficiently achieving resource load balance. Our abstract architecture supports custom, even array-specific, memory interleaving to avoid memory

bank conflicts, dynamic sharing of memory modules amongst processing elements, enabling the programming of hardware structures to rapidly propagate load information, and distribute tasks. For example, array references which cause bank conflicts can be optimized by changing the address mapping for the node, or even for the individual array. Another example is custom load balance structures which can distribute tasks through hardware priority structures; such structures can help to achieve low latency load distribution, improving application scalability.

4 MORPH Software Architecture

High performance computing systems cannot dictate the software structure of next generation high-performance computing applications: their very complexity will demand the best software structuring and complexity management techniques available. These applications not only require high computational rates, massive memory resources, and high performance I/O, they will be substantially more complex than current generation applications, exploiting sophisticated adaptive algorithms that use complex data structures, combining diverse computational applications (meta-computing), and integrating computation, visualization, databases and scientific exploration. The tools for building such applications will be the best mainstream software technology available: object-oriented programming, component libraries (e.g. POOMA, A++/P++, Scalapack), domain-specific libraries (e.g. KeLP, AMR++), or problem solving environments (perhaps as high level as MATLAB). Applications may consist of several independent programs, composed by procedure calls (shared memory), object-interoperability frameworks (CORBA, OLE, SOM), or even messaging (e.g. MPI [33], TCP/IP [34]). These software structures have direct implications for which implementation techniques are feasible. Achieving good programmability demands tools and techniques which allow applications of this type to achieve high performance on our flexible architecture.

Mapping an application onto a configurable architecture such as MORPH involves selecting an appropriate execution model for program sections (e.g. memory and object consistency models as well as hardware primitives), node memory capacity and the size of domains for cache coherence (to match working set structures), custom operations for optimized communication and coordination (e.g. a memory side atomic swap register or a histogrammer), as well as mapping those structures (along with the computation and data) onto the underlying machine. These are daunting tasks, which for common choices can be achieved via libraries (e.g. globally shared memory, clustered cache-coherent machines, or distributed memory). However these approaches are likely to present too inflexible a view to support both a wide range of applications and demanding irregular, adaptive applications well. We believe that achieving scalable high performance on a wide range of applications demands the development of technologies (automatic and high level abstractions for programmer assisted decisions) to exploit the flexibility of our proposed architecture.

There are two basic types of techniques for identifying opportunities for customization: static analysis (compiler analysis and directives) and dynamic adaptation (profiles and dynamic statistics) to rationally make use of the flexibility to optimize the mapping and execution of the program. It is imperative that good performance be achievable with modest effort and the highest levels of performance be available with reasonable tuning effort.

Automatic techniques which exploit aggressive interprocedural analysis [35, 36, 37, 38, 39, 40, 41, 42], profile data, and run-time statistics to optimize program implementation choices are essential to the programmability of the machine and accessibility of high performance. Aggressive compiler analysis has been essential to high performance computing based on vector, shared memory, and distributed machines. Extensions to interprocedural techniques will continue to yield significant benefits as analysis is broadened to include the entire program. However, the heterogeneous and variegated expression of applications (see above) will limit the range of the regularized semantics amenable to compiler analysis.

Because of the limitations of static program analysis and fundamental hardware technology trends which increase the performance sensitivity of parallel machines, profile data and runtime statistics will increasingly important for achieving robust high performance. For example, profiling and runtime statistics may be essential for automatically tuning cache coherence and blocking. Configurable hardware can be configured as instruments for idiom recognition or traditional statistics collection. These forms of fast monitoring can then be used to drive selection of node granularity (memory stealing), mechanisms, assess incremental miss rates and adaptation to larger node memories (stealing memory). The range of possibilities is endless. A change in memory grain size can be detected, for example, by maintaining a record of the cache misses. We propose to evaluate hardware assists that automatically detect changes in the memory grain size and context sets. This hardware would be synthesized to implement a cache tag recognizer us-

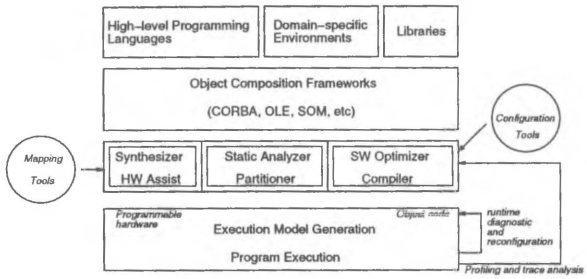


Figure 3: A Software Architecture for a 100 TeraOp Architecture. This picture shows use of runtime diagnosis and synthesized hardware assists.

ing a modified version of the Aho-Corasic algorithm for string matching [43]. If the recognizer detected a desirable configuration change, the hardware modifications could be precomputed or even synthesized on the fly, and programmed into the appropriate hardware.

Profiling and Programmer Annotation While a full complement of programmer annotation, profile-directed analysis, and even on-the-fly performance and diagnosis techniques are essential, the critical issue is the abstractions used to present performance data and system characteristics to the programmer. In addition to traditional views – execution time distribution, cache miss rates, communication volume, load balance, etc., tools for these flexible systems will add aspects of computational efficiency (special operations), internal node communication (memory hierarchy performance, memory bank organization) and external node communication (parallel decomposition), and even suggest/execute program reorganizations which enhance performance.

These techniques are pictured in Figure 3 which illustrates the interplay of the software application structure, automatic and programmer aided optimization, and the software and hardware synthesis to build the implementation. Optimizing compilers will analyze whole programs, generating code structures, specifying hardware structures, and execution models. Hardware will be generated from high-level synthesis techniques and together the hardware and software will be mapped to the underlying machine. Special hardware functionality would generally be mapped to all parts of the machine that require it (as part of their execution model). The extraction of special operations, guidance for selecting special policies, etc. would also be guided by compiler analysis as well as programmer and tool-generated annotations.

5 Application-Driven Customizability

Given the early stage of the MORPH project, and the space constraints here, comprehensive listing of mechanisms for application-driven customizability is not possible. Critical issues in assessing mechanisms include hardware cost, cycle-time impact, configuration cost, effect on software, protection mechanism(s) needed, etc. We describe several illustrative examples below to show the leverage and importance of a flexible architecture. However, with such a small group, these are neither representative nor typical; however, they do illustrate the overall architectural framework that MORPH provides.

Vector Memories: Stride Skewing for Performance Vector memories achieve high performance on regular structures of accesses, but performance drops quickly if accesses fall to the same memory banks, causing bank conflicts, or if accesses cannot be mapped into the vector model with constant stride. The tremendous flexibility of MORPH architecture allows this problem to be easily solved: by using the programmable logic on the processor elements to modify the mapping of addresses to memory elements. For example, shuffling the address lines or using more complex hash functions eliminates the conflicts. However, preserving program correctness, is a little trickier, but also manageable with the programmable logic. By choosing several good hash functions, with complementary structures [44], we can ensure fewer bank conflicts, and by ensuring that addresses are mapped consistently (address ranges, context registers, additional instructions, etc.), correct program execution is ensured. Likewise for sparse matrix operations, where scatter-gather operations would typically be employed, the programmable logic can be used to prefetch the irregular structure efficiently (data structure interpretation) or even remap the addresses, to present them to the processor (and perhaps pack them into the cache) in contiguous addresses.

Optimizing Cache Granularity for Performance Virtually all processors are critically dependent on their cache subsystems to achieve high performance. However, cache performance can be extremely sensitive to the relationship of the working set to cache size (particularly in direct map caches). Programmable logic can be used to diagnose this problem, and if appropriate implement corrective measures which minimize the performance losses (changes in cache sizes, or expanding victim cache buffers). For example, consider an L1 cache of 8KB, for which the critical working

set size is 9KB. The cache misses are collected by a hardware recognizer that invokes a hardware synthesizer to build customized victim caches [45]. The recognizer analyzes cache misses on-the-fly, assessing how a particular size victim cache would affect the cache miss-rate. Note that the recognizer can be extremely simple – no larger than the tag store for the size of victim cache that is being considered. The recognizer can be a simple acceptor automata that accepts only the cache tags in the L1 cache. The recognizer holds a state for the frequently used set of cache tags implicitly (as a decision-diagram representation over the tag bits). As the set of cache tags transitions to a new set of cache tags on each miss the state machine is updated to ensure that the new tag state is accepted by the recognizer. (Once the number of states in the recognizer exceed the maximum allowable, the recognizer starts to recycle used states.) Since each update affects only a very small number of tags it is relatively easy to update the recognizer. Once a state update leads to a known state in the recognizer, the recognizer makes an estimation of the working set size based on the structure of the implicitly represented state machine, in particular, the size of the strongly-connected components and evaluates the possibility of building a victim cache in steps of 1 KB. This technique works irrespective of the associativity of the L1 cache since the recognizer summarizes working set and not how this working set is distributed in the cache. Through simple modifications such as victim caches, the hardware assist can exploit the programmable hardware to eliminate sharp falloffs in performance when working set is slightly larger than the cache size. Note also that this runtime monitoring activity does not affect the critical paths and cost of runtime reconfiguration is amortized over long periods time between which these updates take place.

Programmable Coherence to Reduce Communication Researchers have long recognized that a single data management granularity, and single cache consistency policies [32, 46, 47, 48] could not hope to serve all applications equally well. However, hardwired machines must be designed to handle a single common case, to simplify their implementation and as a compromise across a workload. In environments where communication is expensive (e.g. distributed virtual memory systems), coherence systems are customized to minimize communication by exploiting data compression, computing differences, and using coherence policies based on observed (or declared) behavior [49, 50, 51]. Such systems can reduce communication requirements significantly and also improve latencies, but in conventional systems incur significant computation overhead for the requisite bookkeeping. MORPH's configurable logic will allow custom protocols and similar optimizations to be implemented with low overhead, reaping the communication reduction and lower latency benefits without computational overhead. Of course, there are a wealth of cache system optimizations proposed within parallel machines which could be applied in an application-specific manner to achieve best performance [52, 53, 32].

6 Summary

We have proposed a framework for exploration of a new class of machine architectures that use small amounts of reconfigurable logic blocks to achieve customization in the binding and mechanisms affecting the interaction of processing, memory, I/O, and communication resources. To support complex and irregular applications, these machines present a programmable interface, i.e., efficient shared address spaces and data movement, to build these applications using interoperability frameworks such as CORBA, OLE and SOM. This architecture reflects the realities of evolving software and hardware technology by supporting complex software with a systematic, multiple-perspective approach to tuning software performance and by focusing on exploiting configurable logic to reduce communication bandwidth requirements and latency.

Our proposed design, MORPH, is a proxy for this architecture that supports the use of a high level programming system by allowing it to choose the node granularity (memory per processing element), naming structure (global, local, anything in between), coherence (blocks, objects, etc.), and customize the mechanisms for coordination and interaction. Such naming flexibility can make programming easier, or optimize performance by controlling naming and consistency. If desired, a software system can even choose a default configuration, viewing all of the memory as global or private whichever is most convenient. The distribution of memories reflects the hardware packaging limitations, so there is little downside to the configurability. However, philosophically, our hardware architecture goes beyond simply supporting software, it seeks to exploit the information available in the software to provide higher performance.

In summary, MORPH design leverages a wealth of research into optimal architectural mechanisms, customizable cache coherence, as well as technological advances in interprocedural analysis, programmable hardware, and hardware synthesis technology to achieve general-purpose high performance.

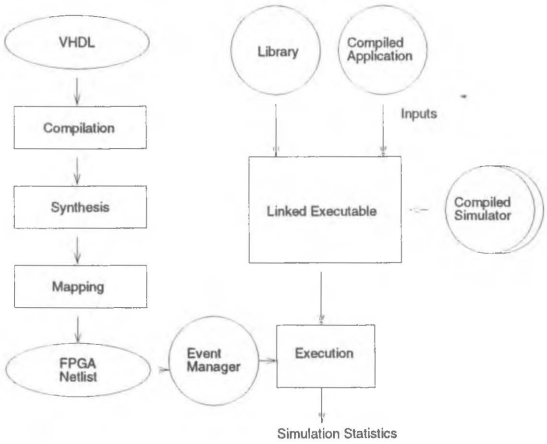


Figure 4: MORPH Simulation Workbench

6.1 Future Work

To evaluate the MORPH architecture, we are pursuing an application-driven methodology. Using a set of well-known numerical kernels, as well as two distinguished applications – computational fluid dynamics and immersive, interactive virtual reality, we are systematically identifying opportunities for customization and assessing its potential benefits.

To date, we have designed a simulation environment to evaluate the effectiveness of custom mechanisms. Figure 4 shows the important components of this simulation environment. Studying the application kernels, we have identified several promising directions for customizability, including cache organization, structure-driven prefetching and replacement (cache management), and several other surprisingly effective optimizations. High-level system simulation is being conducted using (modified) MINT simulator, and results from projected technology spreadsheet [54]. We are also evaluating synthesis paths directly from C-models to speed up evaluation of customizability cost-benefit analysis. We expect to report these results in future papers over the next calendar year.

7 Acknowledgments

The authors gratefully acknowledge support from National Science Foundation award ASC-96-34947. The authors are also supported by an NSF Young Investigator Award CCR-94-57809 and NSF CAREER Award MIP-95-01615.

References

[1] "Seminconductor technology workshop conclusions," Roadmap, Semiconductor Industry Association, 1993.

[2] P. P. Gelsinger, *et. al.*, "Microprocessor Circa 2000," *IEEE Spectrum*, vol. 26, no. 10, pp. 43–47, Oct. 1989.

[3] C. Hu, "Low-voltage cmos device scaling," in *Proceedings of the IEEE International Solid State Circuits Conference*, 1994.

[4] R. Weiss, "64-Gbit DRAMs, 1-GHz Microprocessors Expected by 2010," *Computer Design*, May 1995.

[5] J. M. Rabaey, *Digital Integrated Circuits*, ch. 8. Prentice-Hall, 1996.

[6] R. K. Gupta, *Co-Synthesis of Hardware and Software for Digital Embedded Systems*. Kluwer Academic Publishers, Boston, 1995.

[7] C. Coelho and G. D. Micheli, "Dynamic scheduling and synchronization synthesis of concurrent digital systems under system-level constraints," in *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 175–181, Nov. 1994.

[8] K.-S. Chung, R. K. Gupta, and C. L. Liu, "An algorithm for synthesis of system-level interface circuits," in *Proceedings of the IEEE International Conference on Computer-Aided Design*, Nov. 1996.

[9] G. Borriello and R. Katz, "Synthesis and Optimization of Interface Transducer Logic," in *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 274–277, Nov. 1987.

[10] P. Chou, R. Ortega, and G. Borriello, "Synthesis of the Hardware/Software Interface in Microcontroller-Based Systems," in *Proceedings of the IEEE International Conference on Computer-Aided Design*, (Santa Clara), pp. 488–495, Nov. 1992.

[11] E. Anderson *et. al.*, *LAPACK Users' Guide*. Philadelphia, PA: SIAM, 1992.

[12] J. J. Dongarra, R. Pozo, and D. W. Walker, "LAPACK++: A design overview of object-oriented extensions for high performance linear algebra," in *Proceedings of Supercomputing'93*, pp. 162–171, 1993.

[13] T. Keffer, ed., *Proceedings of the SIAM Object-oriented Numerics Conference*. Rogue Wave Software, 1993.

[14] T. Keffer, ed., *Proceedings of the SIAM Object-oriented Numerics Conference*. Rogue Wave Software, 1994.

[15] S. Baden, "Programming abstractions for run-time partitioning of scientific continuum calculations running on multiprocessors," Tech. Rep. LBL-24643, University of California, Lawrence Berkely Laboratory, 1988. Also in Proceedings of the Third SIAM Conference on Parallel Processing for Scientific Computing, December 1987, Los Angeles, Calfornia.

[16] Various, ed., *Proceedings of the SIGPLAN Conference on Object Oriented Programming, Systems, Languages, and Applications*. ACM Press, 1986-.

[17] "Common object request broker: Architecture and specification," Tech. Rep. Document Number 93-, The Object Management Group, 1993.

[18] "Orb 2.0 rfp submission," Tech. Rep. Document 94.9.41, The Object Management Group, 1994.

[19] W. Wulf and S. McKee, "Hitting the memory wall: Implications of the obvious," *Computer Architecture News*, 1995.

[20] C. L. Seitz, "Let's route packets instead of wires," in *Proceedings of the 6th MIT Conference on Advanced Research in VLSI* (W. J. Dally, ed.), pp. 133–37, MIT Press, 1990.

[21] L. Choi and A. A. Chien, "The design and performance evaluation of the DI-multicomputer," *Journal of Parallel and Distributed Computing*, 199x. Submitted for publication.

[22] L. Choi and A. A. Chien, "Integrating networks and memory hierarchies in a multicomputer node architecture," in *Proceedings of the International Parallel Processing Symposium*, April 1994. Available from `http://www-csag.cs.uiuc.edu/papers/ipps94.ps`.

[23] W. J. Dally and P. Song, "Design of a self-timed vlsi multicomputer communication controller," in *Proceedings of the International Conference on Computer Design*, pp. 230–4, IEEE Computer Society, 1987.

[24] W. J. Dally, *et. al.*, "The message-driven processor," *IEEE Micro*, pp. 23–39, April 1992.

[25] W. Dally, A. Chien, J. Fiske, G. Fyler, W. Horwat, J. Keen, R. Lethin, M. Noakes, P. Nuth, and D. Wills, "The message driven processor: an integrated multicomputer processing element," in *Proceedings of the 1992 IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 416–19, 1992.

[26] E. Spertus, S. C. Goldstein, K. E. Schauser, T. von Eicken, D. E. Culler, and W. J. Dally, "Evaluation of mechanisms for fine-grained parallel programs in the J-Machine and the CM-5," in *Proceedings of the 20th Annual International Symposium on Computer Architecture*, pp. 302–313, IEEE Computer Society, 1993. Availabe from `http://www.cs.cornell.edu/Info/People/tve/ucb_papers/isca93.ps`.

[27] S. Borkar, *et. al.*, "Supporting systolic and memory communication in iWarp," in *Proceedings of the 17th International Symposium on Computer Architecture*, pp. 70–81, IEEE Computer Society, 1990.

[28] S. Borkar, *et. al.*, "iWarp: An integrated solution to high-speed parallel computing," in *Proceedings of Supercomputing '88*, pp. 330–341, IEEE Press, 1988. Orlando, Florida.

[29] T. H. Szymanski and H. S. Hinton, "Graph Embeddings in a Photonic HyperPlane," in *Proceedings of the ICPAT-94*, June 1994.

[30] Cray Research, Inc., Eagan, Minnesota 55121, *CRAY T3D Software Overview Technical Note*, 1992.

[31] Thinking Machines Corporation, 245 First Street, Cambridge, MA 02154-1264, *The Connection Machine CM-5 Technical Summary*, October 1991.

[32] B. Falsafi, *et. al.*, "Application-specific protocols for user-level shared memory," in *Proceedings of Supercomputing '94*, (Washington, D.C.), 1994.

[33] Snir, et. al., *MPI: The Complete Reference.* MIT Press, 1995.

[34] D. E. Comer, *Internetworking with TCP/IP Vol I: Principles Protocols, and Architecture, 2nd edition.* Englewood Cliffs, NJ: Prentice Hall, 1991.

[35] J. Plevyak and A. A. Chien, "Precise concrete type inference of object-oriented programs," in *Proceedings of OOPSLA '94, Object-Oriented Programming Systems, Languages and Architectures*, pp. 324–340, 1994.

[36] J. Plevyak, V. Karamcheti, X. Zhang, and A. Chien, "A hybrid execution model for fine-grained languages on distributed memory multicomputers," in *Proceedings of Supercomputing'95*, 1995.

[37] J. Plevyak and A. Chien, "Compiling object-oriented programs," in *submitted for publication*, 1995.

[38] K. D. Cooper, M. W. Hall, and K. Kennedy, "A methodology for procedure cloning," *Computer Languages*, vol. 19, no. 2, pp. 105–118, April 1993.

[39] M. W. Hall, J. M. Mellor-Crummey, A. Clarle, and R. G. Rodríguez, "FIAT: A framework for interprocedural analysis and transformation," in *Proceedings of the Sixth Workshop for Languages and Compilers for Parallel Machines*, pp. 522–545, August 1993.

[40] M. W. Hall, K. Kennedy, and K. S. McKinley, "Interprocedural transformations for parallel code generation," in *Proceedings of the* 4th *Annual Conference on High-Performance Computing (Supercomputing '91)*, pp. 424–434, Nov. 1991.

[41] J.-D. Choi, M. Burke, and P. Carini, "Efficient flow-sensative interprocedural computation of pointer-induced aliases and side effects," in *Twentieth Symposium on Principles of Programming Languages*, pp. 232–245, ACM SIGPLAN, 1993.

[42] G. Agrawal, J. Saltz, and R. Das, "Interprocedural partial redundancy elimination and its application to distributed memory compilation," in *Proceedings of the 1995 ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 258–269, June 1995.

[43] A. Aho and M. Corasick, "Efficient String Matching: An Aid to Bibliographic Search," *Communications ACM*, vol. 18, no. 6, pp. 333–340, June 1975.

[44] F. Bodin and A. Seznec, "Skewed associativity enhances performance predictability," in *Proceedings of the International Symposium on Computer Architecture*, 1995.

[45] N. P. Jouppi, "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," in *Proceedings of the International Symposium on Computer Architecture*, pp. 364–73, 1990.

[46] Z. Zhang and J. Torrellas, "Speeding up irregular applications in shared-memory multiprocessors: Memory binding and group prefetching," in *Proceedings of the International Symposium on Computer Architecture*, 1995.

[47] P. Stenström, T. Joe, and A. Gupta, "Comparative performance evaluation of cache-coherent numa and coma architectures," in *Proceedings of the 19th International Symposium on Computer Architecture*, pp. 80–91, IEEE Computer Society, IEEE Press, 1992.

[48] M. D. Hill and A. J. Smith, "Experimental evaluation of on-chip microprocessor cache memories," in *Proc. Eleventh International Symposium on Computer Architecture*, June 1984.

[49] K. L. Johnson, M. F. Kaashoek, and D. A. Wallach, "CRL: High-performance all-software distributed shared memory," in *Proceedings of the Symposium on Operating Systems Principles*, 1995.

[50] P. Keleher, A. L. Cox, S. Dwarkadas, and W. Zwaenopol, "Treadmarks: Distributed shared memory on standard workstations and operating systems," in *Proceedings of the 1994 Winter Usenix Conference*, pp. 115–132, Jan. 1994.

[51] J. Bennett, J. B. Carter, and W. Zwaenepoel, "Munin: Distributed shared memory based on type-specific memory coherence," in *Proceedings of the Second ACM SIGPLAN Symposium on the Principles and Practice of Parallel Programming*, 1990.

[52] D. Lenoski and et al., "The Stanford DASH Multiprocessor," *IEEE Computer*, pp. 63–79, Mar 1992.

[53] S. Eggers, *Simulation Analysis of Data Sharing in Shared Memory Multiprocessors.* PhD thesis, University of California at Berkeley, 1989.

[54] P. M. Kogge, "Summary of the architecture group findings," in *PetaFlops Architecture Workshop (PAWS)*, Apr. 1996.

IEEE COMPUTER SOCIETY
50 YEARS OF SERVICE • 1946-1996

http://www.computer.org

Press Activities Board

IEEE Computer Society Press Publications

The world-renowned Computer Society Press publishes, promotes, and distributes a wide variety of authoritative computer science and engineering texts. These books are available in two formats: 100 percent original material by authors preeminent in their field who focus on relevant topics and cutting-edge research, and reprint collections consisting of carefully selected groups of previously published papers with accompanying original introductory and explanatory text.

Submission of proposals: For guidelines and information on CS Press books, send e-mail to cs.books@computer.org or write to the Acquisitions Editor, IEEE Computer Society Press, P.O. Box 3014, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314. Telephone +1 714-821-8380. FAX +1 714-761-1784.

IEEE Computer Society Press Proceedings

The Computer Society Press also produces and actively promotes the proceedings of more than 130 acclaimed international conferences each year in multimedia formats that include hard and softcover books, CD-ROMs, videos, and on-line publications.

For information on CS Press proceedings, send e-mail to cs.books@computer.org or write to Proceedings, IEEE Computer Society Press, P.O. Box 3014, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314. Telephone +1 714-821-8380. FAX +1 714-761-1784.

Additional information regarding the Computer Society, conferences and proceedings, CD-ROMs, videos, and books can also be accessed from our web site at www.computer.org.

6/11/96

Published by the IEEE Computer Society Press
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314

IEEE Computer Society Press Order Number PR07551
IEEE Order Plan Catalog Number 96TB100062
ISBN 0-8186-7551-9
ISSN 1088-4955

ISBN 0-8186-7551-9

90000>

9 780818 675515

Frontiers '96 The Sixth Symposium on The Frontiers of Massively Parallel Cor


TxU

TXU