JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 16, 199–211 (1992)

INVITED PAPER

Report of the Purdue Workshop on Grand Challenges in Computer Architecture for the Support of High Performance Computing*

HOWARD JAY SIEGEL,†,1 SETH ABRAHAM,1 WILLIAM L. BAIN,2 KENNETH E. BATCHER,3 THOMAS L. CASAVANT,4 DOUG DEGROOT,5 JACK B. DENNIS,6 DAVID C. DOUGLAS,7 TSE-YUN FENG,8 JAMES R. GOODMAN,9 ALAN HUANG,10 HARRY F. JORDAN,11 J. ROBERT JUMP,12 YALE N. PATT,13 ALAN JAY SMITH,14 JAMES E. SMITH,15 LAWRENCE SNYDER,16 HAROLD S. STONE,17 RUSS TUCK,18 AND BENJAMIN W. WAH19

1*Purdue University,* 2*Intel Corporation SSD,* 3*Kent State University,* 4*University of Iowa,* 5*Texas Instruments,* 6*Massachusetts Institute of Technology,* 7*Thinking Machines Corporation,* 8*Pennsylvania State University,* 9*University of Wisconsin,* 10*AT&T Bell Laboratories,* 11*University of Colorado,* 12*Rice University,* 13*University of Michigan,* 14*University of California at Berkeley,* 15*Cray Research, Inc.,* 16*University of Washington,* 17*IBM Thomas J. Watson Research Center,* 18*MasPar Computer Corporation, and* 19*University of Illinois*

The "Purdue Workshop on Grand Challenges in Computer Architecture for the Support of High Performance Computing" was sponsored by the National Science Foundation to identify critical research topics in computer architecture as they relate to high performance computing. Following a wide-ranging discussion of the computational characteristics and requirements of the grand challenge *applications,* the workshop identified four major *computer architecture* grand challenges as crucial to advancing the state of the art of high performance computation in the coming decade. These are: (1) idealized parallel computer models; (2) usable peta-ops (10^{15} ops) performance; (3) computers in an era of HDTV, gigabyte networks, and visualization; and (4) infrastructure for prototyping architectures. This report overviews some of the demands of the grand challenge applications and presents the above four grand challenges for computer architecture. © 1992 Academic Press, Inc.

I. INTRODUCTION

A. Origin of the Workshop

"Grand Challenges: High Performance Computing and Communications" is the title of the widely distributed "blue book" [3] that describes the United States Federal High Performance Computing and Communications (HPCC) program. The goal of this program is "to accelerate significantly the commercial availability and utilization of the next generation of high performance computers and networks." The booklet presents a set of "grand challenge problems"—applications that need the major gain in processing power that the HPCC initiative is expected to provide. These problems are characterized by massive data sets, complex operations, and/or irregular data structures that exceed the limits of current supercomputers and programming paradigms.

However, the blue book does not explicitly explore what developments in computer architecture are needed to support the grand challenge applications. This topic arose in discussions between Dr. Zeke Zalcstein of the National Science Foundation and Professor H. J. Siegel of Purdue University. Dr. Zalcstein felt it was important to explore, in a workshop environment, what the relevant key issues in computer architecture are. This report is the result.

"The Purdue Workshop on Grand Challenges in Computer Architecture for the Support of High Performance Computing" was held at Purdue University on December 12 and 13, 1991, to identify critical research topics in computer architecture as they relate to high performance computing. The workshop was sponsored by the Computer Systems Program of the Division of Computer and Computation Research at the National Science Foundation and brought together a small but diverse group of computer architecture researchers. Professors H. J. Siegel and Seth Abraham, both of the School of Electrical Engineering at Purdue University, were the workshop cochairs, and Dr. Zeke Zalcstein was the NSF liaison.

B. The Workshop Charter

To fully appreciate the architectural grand challenges that were the "output" of this meeting, it is instructive to keep in mind the "input" to which the group was responding. To clarify this, the plan for the workshop is quoted below from the invitation sent to the participants.

There is a desire to advance significantly the state of the art of high performance computing. The grand challenges for high per-

* This work was supported by the National Science Foundation Division of Computer and Computation Research Computer Systems Program under Grant CCR-9200735.

† For complete affiliations, please see author biographies following References.

formance computing have been discussed in terms of the applications that can make use of the computing power to be made available. The focus of this workshop can be stated succinctly as follows: what are the grand challenges facing computer architecture that must be met to build high performance computers? The workshop will focus on the design and construction of the hardware architecture. While the hardware cannot be considered in isolation, application and system software issues are beyond the scope of this workshop. This workshop will consider software aspects and application characteristics only where there is an impact on the hardware design.

The goals of the workshop are to list, characterize, categorize, assess the difficulty of, and interrelate these "grand challenges" for computer architecture for the support of high performance computing. This meeting will indicate the areas of computer architecture research that the participants feel are most important and should receive the most attention.

Computer architects from both academia and industry were invited to the workshop. Some invitees could not attend due to scheduling conflicts. Those who attended the workshop are the coauthors of this report.

While it was recognized that hardware technology and software are important considerations and are strongly interrelated with architecture, the group's instructions from NSF were to focus mainly on the hardware architecture organization. Such a focus was necessary due to the limited time duration of the workshop.

C. The Report

This report presents four architectural grand challenges whose achievements would make significant advances towards the goals of high performance computing and communication. These four challenges were distilled from a great variety of views expressed by individual participants and this report is closer to a union of those views than an intersection.

The workshop cochairs have assembled this report from draft material contributed by all workshop participants. Every attempt has been made to reflect fairly the (sometimes conflicting) views expressed, while maintaining a coherent style.

Section II of the report establishes the background for the group's selection of grand challenges in computer architecture by discussing the demands on architecture implied by the U.S. national commitment to supporting the solution of the grand challenge problems. The grand challenges in computer architecture the group felt were most important are stated in Section III. Each challenge is developed in one of the following four sections. Section IV notes that the program execution model supported by a computer system has a strong influence on the performance achievable for applications, and then recommends work toward unifying existing models and developing more comprehensive models for parallel computation. Section V points out that all components of a computer system must evolve to meet the demand for further orders of magnitude improvement in performance, and that special attention is needed to ensure that high performance is realizable in practical applications. Section VI observes that new developments in computer architecture will be needed to support the new communication-intensive applications made possible by advancing technology. Section VII stresses the need for advanced infrastructure tools and software to support the design and evaluation of prototypes for new architectures. Section VIII concludes the report.

This report presents architectural grand challenge problems to the technical community as issues in computer architecture that deserve study. Our hope is to stimulate interest in funding and supporting research efforts to meet the grand challenges of computer architecture and hasten the day that high performance computers for the grand challenge application problems will be a practical reality.

II. GRAND CHALLENGE APPLICATION PROBLEMS AND COMPUTER ARCHITECTURE

A. Grand Challenge Application Problems

The U.S. Committee on Physical, Mathematical, and Engineering Sciences has identified a set of "grand challenge problems" that set a goal for the HPCC initiative, now funded by the U.S. Congress through several agencies. The grand challenge application problems concern pressing issues of human welfare on planet Earth and problems at the exciting frontiers of science that may open doors to better living for future generations.

The blue booklet published by the committee [3] lists 10 areas as posing "problems whose solution is critical to national needs":

Climate Modeling	Quantum Chromodynamics
Fluid Turbulence	Semiconductor Modeling
Pollution Dispersion	Superconductor Modeling
Human Genome	Combustion Systems
Ocean Circulation	Vision and Cognition.

It is estimated that a serious attack on any of these problems will require computer performance in excess of one trillion floating point operations per second (one teraflops).

The grand challenge problems have enormous computational requirements. Consider, for example, the problem of modeling the weather. In 5 years time, data collection facilities will be in place to define detailed atmospheric structures and permit significant advances in forecasting capabilities. However, today's most powerful supercomputers cannot meet the computational requirements. The goal of improving atmospheric modeling resolution to a 5-km scale and providing timely results is believed to require 20 teraflops of performance.

B. One Teraflops and Beyond

Although substantial progress remains to be achieved in uniprocessor technology, because of inherent physical limitations it is assumed that high performance computing will employ parallel systems. The peak performance of currently available massively parallel computers of practical size and cost is at the level of hundreds of gigaflops (10^9 floating point operations per second). To produce practical massively parallel computers having at least one teraflops (10^{12} floating point operations per second) performance, only engineering effort to fully utilize existing, demonstrated technology is needed. These teraflops computers can become available in a few years; however, there is much debate about whether such machines can be produced at a low-enough cost to make them commercially viable for a large customer base. Furthermore, there is a need for environments that will allow application programmers to realize a significant fraction of such a machine's peak speed.

Providing performance significantly beyond teraflops will require major innovations in computer hardware architecture, packaging, and device technology. Optical technology [9] may offer a breakthrough in performance, but it will require a radical rethinking of computer structure and how the technology can support appropriate models of computation. Of course, cost and usability concerns remain.

Many supporting and related areas must also be developed. Improvement is needed in the infrastructure that supports the design, prototyping, and construction of advanced computer hardware. This is also true for high performance peripherals to match the capabilities of the processors. Reliability and fault tolerance will become increasingly critical issues as high performance machines become incorporated into networks, begin to handle communications-intensive information processing, and satisfy real-time demands. Programmability and usability must be facilitated by new programming models and environments.

C. Effective Use of Potential Performance

Achieving ever greater levels of peak performance is not the only challenge resulting from the goals of high performance computing; a significant challenge is to make those levels of performance easily accessible to the end user. We are living in a new era of computing in which the U.S. national laboratories will no longer be the dominant users of high performance computation, and it is no longer feasible to spend 10 person-years of effort to implement an important problem on a supercomputer. In contrast to this circumstance, in many situations the computational models used with current massively parallel computers are dismal in comparison to those familiar to users of conventional computers and workstations. The feeling one senses among some in the community is that increased difficulty of programming is a necessary price to be paid for the benefits of high performance. One of the challenges is to show that this need not be so.

In the near future, most high performance computing will be at the level of 100 megaflops to several gigaflops and will be performed by machines assigned to individuals or small groups of workers, or used in operational information/communication systems of business and industry. The effective use of large-scale parallel machines in these roles requires programming support at least comparable in power and generality to that available on present day workstations. The required programmability demands the adoption of more general models of computation. Development of satisfactory computational models for parallel computers that are efficiently supported by the hardware is a grand challenge of computer architecture. Without support for such computational models, the impact of architectural advancements will be severely impaired.

D. Programming for Massively Parallel Computation

Current programming practice for most massively parallel computers is based on the *data parallel* model of computation [5]. In this model, the principal data structures of a problem (usually large data arrays) are partitioned and assigned to the processors of the machine. It is rare to see large-scale parallel computation where hundreds of processors are performing functionally distinct parts of a job (this is sometimes referred to as *functional parallelism*).

In the case of machines having a distributed memory architecture, a data parallel algorithm is expressed as machine code that is executed by all processors and the necessary communication among processors is implemented by manually coding explicit message-passing commands or by the use of a logically shared address space; the former approach is currently prevalent. Compilers available and under development will automate this process by letting the programmer specify data partitioning and by automatically generating the communications code for the given data partitioning.

A widespread misconception is that the two most important parts of the high performance field are architectures and algorithms. However, the interface between the architecture and the algorithm is a crucial issue as well. The effective programmability of the machine is limited by the computational model and how well that model is supported by the hardware and software of the system, as mentioned in the previous subsection. A major challenge is to move toward architectures that can efficiently implement a truly general-purpose parallel computation model. Architectures must support environments that facilitate functional parallelism in a massive way, as well as data parallelism.

E. The Goal of General-Purpose Parallel Computation

General-purpose computation is not well defined. At one extreme, the term means simply that one is able to perform any algorithm expressed in a complete language. At the other extreme, a general-purpose computer is expected to be efficient for applications ranging from science and engineering to business and industry.

Important programmability features that are standard for general-purpose workstations are not typically available for massively parallel computers. One of these is the ability to execute programs much larger than the physical main memory of the machine without having to program the swapping of information between main memory and disk; this is the familiar virtual memory idea implemented in all workstations. Another limitation concerns the linking of separately compiled programs; there are no standards for communicating large partitioned data structures between compiled modules. Realizing these features within the framework of massively parallel machines is a major challenge in computer science—one that is often lost amid the concentration on hardware and algorithms.

Two of the major issues to be addressed are: (1) providing a global virtual memory for massively parallel computers, and (2) expressing and supporting parallelism and the interaction of concurrent activities. The model of computation supported by the architecture must have the properties necessary to create the desired programming environment. A basic approach to the challenge is to choose a model of computation that simultaneously serves as the specification of an architecture and the target language for high-level programs. However, portability of parallel programs is also an important consideration.

F. Demands of the New Applications

The enormous rise in computer performance is making qualitative changes in the expectations and interests of users. For example, experience with larger computational grids and three-dimensional modeling of physical phenomena is motivating the use of more sophisticated data structures. In weather modeling, more effective methods are possible if computing resources are concentrated on unstable portions (e.g., storm systems) of the simulated space. However, unstructured grids make efficient usage of the processors in a parallel machine difficult.

Other areas include symbolic manipulation, compiling, heuristic search, etc. These types of computation are important in image analysis [6] and may be crucial to solving the human genome problem [3]. University research has shown that these problems often have high levels of parallelism. However, as mentioned earlier, these problems are characterized by massive data sets, complex operations, and/or irregular data structures that exceed the limits of current supercomputers and programming paradigms. Making massive parallelism readily available in an effective and "user-friendly" manner for applications involving these characteristics requires the development of new techniques for mapping tasks onto parallel architectures.

Finally, the computing technology of the 1990s will enable access to vast information sources such as digital libraries, visualization images, and multimedia information objects [1]. Future computers must deal with such data entities as though they were the simple textual messages of today. The challenge is to incorporate into computers a high capacity to handle and transform these data.

III. GRAND CHALLENGE PROBLEMS IN COMPUTER ARCHITECTURE

A. The Architectural Grand Challenges

The workshop opened with a wide-ranging discussion surrounding the computational characteristics and demands of the grand challenge *application* problems. From these requirements, the participants translated the application-centered grand challenges into grand challenges for *computer architecture* for high performance computing. From a lengthy list of challenges, the attendees selected four primary challenges for presentation:

(1) idealized parallel computer models,
(2) usable peta-ops performance,
(3) support of I/O and intensive communications, and
(4) infrastructure for prototyping architectures.

It was recognized that the list from which these four were selected was by no means exhaustive, and that these four challenges overlapped and interacted.

This subsection summarizes these grand challenges for computer architecture. Sections IV through VII examine each problem in more detail and consider approaches for attacking them.

Grand Challenge 1: Idealized Parallel Computer Models. The model of parallel computation is fundamental to progress in high performance computing because the model provides the interface between parallel hardware and parallel software. It is the idealization of computation that computer architects strive to support with the greatest possible performance. The model is the specification of the computational engine that language and operating systems designers can assume as they seek to enhance the power and convenience of parallel machines. It is not clear that a single model can fulfill all of the requirements, but it is essential to reduce the multitude of alternatives to the fewest possible number. Therefore, it is important to identify one "universal" or a small number of "fundamental" models of parallel computation that serve as a natural basis for programming languages and that facilitate high performance hardware implementations.

Grand Challenge 2: Usable Peta-Ops Performance. Grand challenge applications require usable computer performance orders of magnitude greater than the giga-ops performance available today and the tera-ops performance that may be achieved soon. This computer performance cannot be obtained by simply interconnecting massive quantities of existing cpu, memory, and I/O resources, because the collective overhead associated with these interconnected resources can produce a system that is unmanageable to program and ineffectively utilizes its components. The challenge is to (1) dramatically improve and (2) effectively harness the base technologies impacting processors, memory, and I/O into a computer system such that the grand challenge applications programmer has easy access to a peta-ops (10^{15} operations per second) of usable processing performance.

Grand Challenge 3: Computers in an Era of HDTV, Gigabyte Networks, and Visualization. Technology will be able to support startling new communications-intensive applications. For example, concurrent access by thousands of people to a digital version of the Library of Congress may be within reach in this decade. Digital video will enable workstations of the future to treat images as easily as characters and words are treated today. How can computer architecture and new communications technology evolve to facilitate such applications?

Grand Challenge 4: Infrastructure for Prototyping Architectures. Given that computer generations change every 2 to 3 years, new ideas on architecture must be evaluated and prototyped quickly. Prototype development involves not only hardware, but also software in the form of compilers and operating systems. An infrastructure is needed to facilitate the study of the effects of new hardware technologies and machine organizations against different application requirements. This computer architecture challenge is to develop sufficient infrastructure to allow rapid prototyping of hardware ideas and the associated software in a way that permits realistic evaluation.

B. Multidisciplinary Approach

The architectural grand challenges stated above are inherently multidisciplinary and involve team efforts that cross boundaries from software to hardware to applications. Early efforts in the development of parallel computers have shown that their viability and usability is a strong function of the supporting software systems. A substantial component of effort must be devoted to the automation of the software development process to exploit the power of the underlying hardware. This includes such problem areas as algorithm selection, algorithm optimization, data mapping, and parallelization. In the arena of high performance parallel computers, it is more important than ever for computer architects to consider the issues of system software, application needs, and usability when designing and implementing machines. Computer architects must design systems that will efficiently support the software tools that will make the systems useful; it is a symbiotic relationship that must be leveraged to the fullest extent.

IV. GRAND CHALLENGE 1: IDEALIZED PARALLEL COMPUTER MODEL

This architectural grand challenge is to identify one "universal" or a small number of "fundamental" models of parallel computation that abstract the essential features of parallel machines. The desired model is an idealized parallel computer analogous to the familiar von Neumann machine. This idealized machine model must characterize those capabilities that are so fundamental to parallel computation that all but the most specialized parallel computers can be expected to provide them. The abstraction need not imply any structural information, but it should capture implicitly the relative costs of parallel computation.

A parallel computation model differs from the von Neumann model in the ways parallel computing differs from serial computing, e.g., having multiple processors and a communications structure. Implementation details such as the number of processors and the interprocessor communications structure are unimportant except to the extent to which they affect performance. The challenge of constructing such a model is to be "precise enough" about performance without being "too explicit" about the implementation details.

This challenge is one of the most widely discussed topics in parallel architecture circles. The need for a parallel model characterizing the capabilities and costs of parallel computers has long been recognized [8, 10]. Such a model is essential for computer architects, software developers, and algorithms designers.

For parallel computer architects, the model should define those capabilities that are critical to parallel computation and should execute as fast as possible in any parallel computer design. Enhancing these basic features (as caches have enhanced memory references for the von Neumann model) then becomes the focus of computer engineering and architecture research. The hardware must support the parallel computation model in a cost-effective way by dealing with practical design constraints including packaging, available commodity parts, standard buses and protocols, and many other technological considerations. These considerations lead to the use of multiple highly integrated processors, memory hierarchies, and physically and/or logically distributed memories. Some implementations may even provide a physical structure that is quite different from the logical model. Thus, the model must have practical hardware realizations, but not dictate specifics of those realizations. The goal of computer architecture research will continue to be what it has always been: finding hardware realizations that perform the computations of the model faster.

For software developers, the model will specify those facilities that can be assumed in the underlying parallel computers. Languages and compilers can target this idealized machine model and then be specialized to any particular hardware platform, as is done for portable compilers for sequential machines. The model must be capable of providing information about the relative costs of parallel computations. This is essential so that language designers can judge the efficiency of the likely implementations of their language constructs, and compiler writers can develop efficient execution-time virtual machines. The model must be capable of supporting a wide range of high-level programming structures. Moreover, it should permit program specification with a minimum of explicit synchronization. With such a model as a guideline, it should be possible to develop efficient and portable parallel programming systems.

Finally, for algorithm designers and programmers, this fundamental model of parallel computation will provide the basis for program development and accurate algorithm analysis, as well as providing the foundation for a realistic theory of parallel algorithms. To do so, the model must provide meaningful information on the relative costs of computation, communication, and synchronization. It must also provide a basis for useful feedback of performance and debugging information to the programmer. The model, therefore, is the foundation on which efficient algorithms and programs can be developed.

The properties described above are goals. A model can be useful even if it does not achieve all of them. Nevertheless, they serve as a yardstick by which proposed computation models can be judged.

Developing a model to meet the above specifications will be a challenge. However, with the understanding of parallel computers, algorithms, and languages expanding, the prospects for creating an ideal model of parallel computation improve. The obvious approaches have advantages and disadvantages.

1. Existing model: There is no existing model of parallel computation that satisfies the conditions above; for example, the well-known PRAM model does not capture communication costs. There does not appear to be an existing model that is generally applicable, provides the necessary information, and is practically realizable.

2. New model: Discovering an entirely new model of parallel computation is perhaps the most ambitious solution to the problem. It is not only difficult to fulfill the above goals, such as building a physical realization, but it appears that feedback and experience are needed to correct and enhance a model. Starting from first principles is difficult and success is perhaps unlikely. However, due to limitations of existing models, this may be a worthwhile pursuit for the adventurous.

3. Evolution: Perhaps the most productive approach would be to revise an existing model to resolve its inadequacies. This adaptation, for example, might add structure-specifying capabilities to the shared memory model or assistance for barrier synchronization in a message-passing model [2, 8]. It may also involve combining features of different existing models. The ideal is not likely to be developed simply by going down the list of goals and adding features to the model to achieve each goal. Rather, a more satisfactory solution may be derived from the combination of an existing model and an enhancement that is tightly integrated to the other features of the model.

Among the challenges in formulating an idealized model of parallel computation and having it be widely accepted is the need to balance generality and specificity. The model must be sufficiently abstract so as not to limit the creativity of machine designers. However, as previously stated, to be useful to software developers and algorithm designers, it must provide realistic information on the relative costs of computation, communication, and synchronization. At a certain level, the von Neumann model has managed to strike this balance for sequential computation. Though the parallel case is more complicated, achieving a balance is certainly possible in principle.

In summary, this computer architecture challenge is to formulate a model of parallel computation that abstracts the operational features and the costs of parallel computation. This should be approached in a way that will serve as a target for architects to implement and the foundation on which software developers and programmers can build.

V. GRAND CHALLENGE 2: USABLE PETA-OPS PERFORMANCE

This architectural grand challenge is to dramatically improve and effectively harness the base technologies into a future computer system that will provide usable peta-ops of computer performance to grand challenge application programmers. Meeting this challenge may require research leading to the realization of each of the following: (1) a uniprocessor whose microarchitecture alone will provide, transparent to the software, a factor of 10 improvement in performance over what is feasible today; (2) a scalable, logically shared-memory parallel processing system node that will provide a seamless address space that will include a programmer-friendly connection to its I/O subsystem; (3) a massive interconnection of these system nodes that will not be severely degraded by communication software; (4) modularity of design that will allow advances in base technologies such as optical interconnects and semiconductor physics to be reflected in improved system performance without requiring massive changes to the rest of the computing paradigm; (5) massive improvements in available memory bandwidth and effective utilization of that bandwidth; (6) built-in hardware fault tolerance that will allow functioning of this massively concurrent hardware in the presence of the faults that one can expect will usually be present; and (7) cost-effectiveness that will enable a successful commercialization of the hardware.

Each of these components is important to the goal of providing usable peta-ops performance. The current state of processor, memory, and I/O technology lacks these components: uniprocessors do not exploit available instruction stream parallelism; shared-memory multiprocessors do not scale; address-space partitioning of the memory hierarchy and I/O space introduces translation overhead resulting in execution-time inefficiencies and difficulty in programming and debugging; exposure of underlying hardware idiosyncrasies adversely affect introduction of new base technologies such as optical links and new semiconductor devices; usable bandwidth is only a fraction of the peak bandwidth available; latency in information transfer adversely affects throughput; etc. A more complete discussion of the importance of each component problem is contained in this section, along with a brief discussion of approaches to solving that problem. If these component challenges can be met, it will be possible to have future computer systems consisting of integrated processors, memory, and I/O subsystems that provide peta-ops of usable computer performance to grand challenge applications programmers.

Ten different component problems that need to be addressed to achieve usable peta-ops performance were identified. Not all participants in the workshop agreed on the method of approach for dealing with each component, or even (more fundamentally) on the relative importance of addressing each component. Nonetheless, with this disclaimer of nonconsensus, listed below are approaches to several components of this grand challenge, along with expanded discussions of the importance of each.

1. Optimal uniprocessors: The uniprocessor executes the single instruction stream produced by the compiler. If it can exploit the existing parallelism present in the instruction stream with its microarchitecture, the performance it would obtain would be transparent to the software. It is expected that a factor of 10 improvement in performance can be realized at this level of the execution hierarchy.

Most important is to start with a clean sheet of paper and not be concerned with existing software investment and the constraints that compatibility imposes. Understandably, this is not easy to undertake in an industrial environment. But it may be critical to undertake to achieve peta-ops performance.

Second, one must understand the division of labor between what the compiler can provide and what the execution-time hardware should provide. This division should take into account dramatic increases in hardware capability that will be available in the next few years, for example, 10 to 30 million transistors on a chip and optical interconnects. One should design with these technologies in mind.

One should understand the capabilities and limitations of compiler technology, and should use these in determining how best to utilize the hardware resources in designing the microarchitecture of the uniprocessor. The computational characteristics of the codes in expected workloads should also be considered. Choices with respect to superscalar, superpipelined, VLIW, depth of pipelining, degree of branch prediction, and additional hardware assists (such as a branch target cache) must be made in light of both semiconductor capability and compiler technology.

2. Scalable parallel processing system nodes: One element of a peta-ops machine is the scalable logically shared memory parallel processing system node. In this context, scalable implies that the node may be used effectively in massively parallel systems that have a shared address space and provide usable peta-ops performance. It is the architectural element for which future compilers will be required to generate optimized code. The development of a logically shared memory parallel processing system requires substantive awareness of the capabilities of compilers and operating systems and the detailed understanding of the individual uniprocessors, as well as knowledge of the basic issues indigenous to parallel processing itself, such as interconnection structures, cache consistency protocols, and synchronization mechanisms.

The goal is to integrate the system design with compiler optimization technology to provide performance that is a significant fraction of $N \times P$, where N is the number of processors and P is the power of a component uniprocessor. This means that the aggregate power grows with the number of processors, and that the power that can be applied to a single process also scales similarly with the number of processors.

3. I/O subsystems: As processor speeds continue to improve dramatically and memory sizes (and to a lesser extent, memory access times) continue to improve, the bottleneck to a balanced high performance computing system increasingly becomes the I/O subsystem. I/O subsystems should be designed to accommodate the following features. They should be usable by applications and by most of the operating system with little or no knowledge of device technology or low-level interfaces. The actual interfaces implemented should permit performance close to that available from the raw hardware, with high levels of parallelism. Once they are defined, these interfaces should not be changed over time without significant reason so that investments in applications and operating system software are maintained. The interfaces should implement default parameters which give good performance over a wide variety of workloads and technologies (e.g., block sizes). Interfaces should support the use of I/O devices as part of a uniform memory address space. References to I/O devices should be independent of the topology of the overall system and of how and where the I/O devices are connected. I/O devices should be designed to incorporate modern VLSI technology to the maximum extent possible so as to improve performance and reliability.

4. Uniform address space: Grand challenge applications will deal with large amounts of data (e.g., large data bases, extremely large data sets, HDTV video images). Some mechanism for addressing these data must be developed.

Considerable time and effort are required to manage the memory system. To the extent that the architectural design gives a memory hierarchy of low average access time and high average bandwidth without significant explicit programmer effort, software development is greatly facilitated, and the generality of the software (with respect to system configuration across sites and across time) is greatly enhanced.

One approach is to design a technology transparent memory hierarchy providing a very large address space that automatically provides, with high probability, very low mean access time and high bandwidth. The memory system should be logically sharable among large numbers of processors. This sharing should provide a consistency model. The memory should be scalable to a large number of processors without bottlenecks or loss of performance. It should be able to integrate I/O devices and devices at remote systems into the address space.

5. Technology evolution: Technological evolution is enhanced by hiding the detailed knowledge of underlying hardware idiosyncrasies so as to facilitate the introduction of new technologies. Current high performance computing engines are designed on the basis of connectivity and serial bandwidth on the order of hundreds of connections and megabits/second, and chip densities on the order of 2 to 3 million transistors/chip. Architectural design methods will change dramatically when (in the next few years) optical links provide thousands of connections and gigabits/second bandwidth, and semiconductor technology provides chip densities of 10 to 30 million transistors/chip.

One approach to managing the evolution is, to the extent possible, partition and modularize the design. Also, drive the implementation details to the low-level hardware structures, while retaining at the module interconnection level as high a level of abstraction as possible.

6. Memory bandwidth and access time: The actual performance of processors is strongly influenced (and limited) by the ability of the memory system to provide instructions and operands, and to accept results. Unfortunately, while processor performance has been growing at a rate of 50 to 100% per year for the last 7 years, DRAM performance (measured in access time) has been growing at a rate of only 7% per year [4]. This exponentially growing disparity in need versus supply of memory performance provides a grand challenge to architects.

Memory bandwidth can be increased by addressing (1) individual DRAM device bandwidth, using techniques such as adding more pins (at approximately the same package cost), employing multi-chip modules that add signal wires in some other form, and implementing block mode data transfers; (2) processor-to-memory interconnect, e.g., reexamining the partitioning of processors and DRAMS vis-à-vis the same chip, same memory control unit, etc.; and (3) inserting supporting computational capabilities directly into the memory architecture. Memory access time can be reduced by improved caching techniques and improved cache designs [7]. Because processor cycle times have been reduced faster than memory access time and bandwidth have improved, this issue continues to deserve attention.

7. Software component of communications latency: As processor speed and network bandwidth continue to improve, communications latency has not kept pace. This is because the latency for short messages is dominated by software overhead. If massively parallel systems are to maintain and improve their computation/communication ratios (which fundamentally determine the speedup of an application), then communications latency must be aggressively reduced. These improvements will also enable the exploitation of finer grain programming models than are practical today.

Hardware techniques for implementing software protocols for message passing need to be developed. These techniques should seek to eliminate operating system overhead at the sending/receiving ends, while maintaining system protection. They should implement the necessary protocols to form messages, inject these messages into the network, and remove them at the receiving end. They should also deal with retransmissions and other reliability issues.

8. Fault tolerance: To achieve peta-ops of performance, parallelism should be exploited at all levels. This includes a massive number of nodes. Without substantial built-in fault tolerance, the mean time between failures will decrease rapidly as the number (and complexity) of the components increases. Maintaining acceptable system availability will become a major concern.

Research is needed to analyze the failure modes and rates for massively parallel systems. Architectural techniques are needed to detect, isolate, and recover from failures while minimizing the need to terminate applications and/or restart the system. These techniques will impact node and network designs, as well as operating systems. For example, adaptive routing techniques are required to deal with failures in the interconnection network. Error detection and reporting techniques are needed to propagate failure information to unaffected nodes. Efficient checkpointing schemes must be developed to allow the rollback recovery of affected applications (which may be sending messages).

9. Reducing latency: Reducing or hiding latency makes the problem-solving speed depend primarily on the bandwidth of system components. However, it is easier to increase bandwidth than to reduce latency when scaling large systems suitable for grand challenge problems. Several mechanisms can be invoked to reduce latency, including optimal use of caches, multiplexing the execution of multiple threads, pipelining macro operations, multiprogramming, parallelism (asynchronous) of computation and I/O, and facilitating process and data migration.

10. Cost-effectiveness: An obstacle to building petaops/petaflops systems lies in improving the cost-effectiveness of existing architectural approaches. This is required to reduce the cost of such systems to a level that makes them affordable to build.

Meeting this challenge requires making substantial progress in the following areas: memory bandwidth and access time, communication bandwidth and latency (either explicit or implicit, as in the case of memory being treated as a single global address space), I/O bandwidth and latency, processing power, and high-density packaging. Each involves dramatically improving a critical aspect of performance with minimal change in subsystem cost.

VI. GRAND CHALLENGE 3: COMPUTERS IN AN ERA OF HDTV, GIGABYTE NETWORKS, AND VISUALIZATION

The combination of computing and communications technology in the 1990s will enable access to vast information sources such as digital libraries, images, visualization of physical processes, and interactive multimedia. For example, just as today's processors manipulate individual characters, the processors of the year 2000 will manipulate images. How can computer architecture and new communications technology evolve to enable such applications?

The enabling technology for communications-intensive computing exists today in primitive form and will evolve rapidly in the next decade. This technology includes high-bandwidth networking, high-definition imaging, new compression/decompression techniques, gigaflops arithmetic, and high-density memory devices. The potential applications can bring a dramatic change in the way we live and work. The digital library provides easy and inexpensive access to information sources on a scale never before achieved. Scientific visualization builds physical understanding of complex phenomena and enables scientists to solve problems orders of magnitude more difficult than can be solved with conventional use of computers. Communications can be enhanced by combining voice, animated images, and text where formerly there was only voice or text in isolation, and, only in recent years, video to some extent.

The applications addressed by this architectural grand challenge are those in which a major portion of the computer power is devoted to the processing of high-bandwidth streams of data. Such computers will attach to gigabyte networks and high-definition displays to provide a means for viewing and sharing the massive pools of data that can be processed at one site.

In recent years, there have been both evolutionary and revolutionary advances in base technology. Evolutionary advances in memory devices have lowered the cost per bit and greatly increased capacity. Similar evolutionary advances exist in processing, communications, and storage of all types. Revolutionary advances are bringing quantum leaps in communications and storage. Examples are the application of optical transmission to long-distance networks and optical storage for write-once permanent data storage.

Needs for proposed applications, such as concurrent access to national databases and interactive HDTV visualization, are beyond the reach of the most aggressive existing systems. These applications require improvements in all aspects of system bandwidth, processing power, memory capabilities, and storage far in excess of today's systems. Furthermore, these applications have strict cost thresholds that must be met to make them practical to pursue. Applications for high performance computing that have yet to be conceived will further stretch the bounds of performance and cost.

The computer architecture challenge is to apply technology advances to the applications in new and innovative ways that produce results unachievable in the past. Revolutionary improvements can come from the innovative application of evolutionary technologies to existing applications.

The approach to this grand challenge will be through the selection of high priority problems to be addressed. Then, research efforts devoted to these priority areas will be used to solve fundamental problems, demonstrate the art, and create the market for commercialization of the technology. The following list of problems serves as an indication of the potential directions of this architectural grand challenge.

1. Highly concurrent access to huge, centralized databases such as to a digitally stored Library of Congress: This can be approached with a combination of high-speed communications, advances in data-base organization, and new means for incorporating high-speed processing capability into a database system. Such processing might be in the form of intelligent memory subsystems or in dedicated coprocessors. A possible objective is to shift the processing load from central processors to specialized units to increase performance and to lower system cost. The system must also support the high-speed transfer of digital images, multimedia, hypermedia, and hypertext.

2. HDTV interactive video: This may require the incorporation of specialized communications and buffering components with coprocessors, such as digital signal processors, to produce the required processing and I/O rates for high-definition video. The HDTV interface can transform a workstation into a video phone in which electronic mail or real-time conversations can take place using multimedia: TV image, text, voice, computer generated graphics, and synthesized sound. Documents and information sources can be created as a combination of such sources. Consider, for example, creating video images for a high-definition display that may contain 2 million pixels per image. Images will be transmitted in some compressed form that might require as much as hundreds or thousands of floating-point operations per pixel to reconstruct. Because 30 or 60 frames are required each second, the data processing requirements alone exceed several gigaflops. Additionally, moving such vast amounts of data through the system rapidly will prove challenging. Given that such computing capability must find its way into cost-effective consumer products as well as the scientific computing arena, the architectural challenges are formidable.

3. Large transaction systems: Managing this problem will involve the use of new technologies for data networks, distributed transaction storage, and a means for accessing and updating a shared, distributed data base. This enables the largest commercial and government computer users to provide centralized services on a scale never before achieved. The research should investigate special techniques for communications, journaling and logging, recovery, and consistency control that are suitable for large-scale transactions systems.

4. Advanced interactive design systems that produce "instant" design samples through modeling in plastic or through holographic imaging: These systems require internal communications designed for very high bandwidth, and special high performance attachment to mechanical and video peripherals. Special needs include processing power sufficient to manipulate detailed 3D representations of objects.

5. Virtual reality: This research area requires the merger of new sensor technology with new 3D graphics, video processing, and multimedia techniques to create new levels of virtual world fidelity. Applications include design visualization of such objects as automobiles, aircraft, buildings, and the human anatomy.

6. Portable high performance computers for on-site processing in special situations: This research requires special packaging techniques plus advanced technology for low power consumption and cooling to reduce the size and weight of high performance systems. Typical applications are environments where data reduction has to be done at the site of data collection, and are exemplified by seismic applications and space-borne applications.

VII. GRAND CHALLENGE 4: INFRASTRUCTURE FOR PROTOTYPING ARCHITECTURES

A grand challenge in the development of new architectural ideas is the testing of architectural alternatives and their interactions with software, technology, and applications. The design of computer systems not only involves simulation tools and hardware prototyping facilities, but also requires compilers, operating systems, and application programs that execute on the hardware. Thus, rapid prototyping tools must include facilities for hardware and software integration.

The problem is important because it is costly and time consuming to test ideas and evaluate alternate architectural decisions, especially when hardware and software integration is needed. With computer generations changing every 2 to 3 years, it is not feasible to evaluate promising approaches for a fixed environment, but rather the evaluation requires a "guess" as to the technology and requirements of the future. Simulation is often a poor substitute for prototyping because many facets of the problem may be simplified or overlooked.

To provide an infrastructure for testing new architectural ideas and alternatives, it is essential for researchers to have easy access to *new* commercial computers as well as powerful prototyping facilities. The former allows grand challenge applications to be implemented and evaluated quickly, while the latter allows new ideas to be tested and prototyped with a short lead time.

The first goal can be achieved by providing one or more national facilities in which new commercial architectures and experimental parallel processing systems can be accessed. Support by a fast computer network, multimedia access, technical consultation, and on-line documentation are essential. Such facilities are currently available to a certain extent.

The second goal can be achieved by providing national facilities for testing new architectural ideas. Currently, such support is provided by MOSIS in the development of custom chips. However, the concern here goes past chips and on to full systems. The design and evaluation of system-level prototypes takes an inordinate amount of time, especially when it is necessary to integrate hardware and software together. To this end, support of more powerful hardware and software simulation tools can aid designers in rapidly developing new prototypes. Software tools for such rapid prototyping include the use of a common parallel programming model and the development of portable compiler and operating system modules so a working software system can be assembled quickly. In simulating complete systems comprised of both hardware and software, better tools that span the spectrum from chip-level timing analysis to program-level debugging are desirable.

VIII. CONCLUSIONS

The grand challenge application problems are far more difficult than any problems yet solved by computers. They require systems of unheralded capability. Such systems appear to be within reach by the year 2000 at reasonable cost, but only if significant advances are made in a large number of interrelated areas. Advances in device technology can supply only some of the improvement. The remainder has to be provided by architectures, algorithms, matching architectures and algorithms, system models, and new ideas in structuring systems to meet the application problem challenges.

Computers for the grand challenge application problems will necessarily have characteristics not present today, such as advanced visualization, access to geographically distributed data bases, multigigabyte main memories, and terabyte-per-second communications links. These characteristics need to be factored into the design of architectures to create the hardware and software features that can support and exploit them.

This report has discussed some of the grand challenge problems in computer architecture for the support of high performance computing. In particular, (1) inventing a useful and widely accepted idealized parallel computer model or small set of models; (2) implementing systems that provide sustained usable peta-ops performance; (3) designing architectures that provide the capabilities needed in an era of HDTV, gigabyte networks, and visualization; and (4) creating an infrastructure for the rapid prototyping of new architectural organizations with the associated system software.

These problems are presented to the technical community as issues in computer architecture that demand further study if success is to be achieved in this nation's grand challenge applications. The purpose of this report is to help stimulate some of the research needed to make high performance computers for the grand challenge application problems a practical reality.

ACKNOWLEDGMENTS

The coauthors express their gratitude to Dr. Zeke Zalcstein of NSF for suggesting the workshop and making it possible. In addition, the workshop cochairs thank Dr. Zalcstein for his advice in organizing and focusing the workshop. The cochairs appreciated the efforts of the following people at Purdue University for their assistance in administering the workshop and producing the report: their secretaries Mary DeBruicker and Carol Edmundson; the technical typists Vicky Spence and Kitty Cooper; the workshop student assistants James B. Armstrong, Sameer Gupta, Gene Saghi, and Daniel W. Watson; the financial coordinator Brenda McElhinney; and Continuing Education Administration Conference Coordinator Susan Umberger. We also thank Pearl Wang and Gene Saghi for their comments on the manuscript. A summary of this report was presented at *Frontiers '92: The Fourth Symposium on the Frontiers of Massively Parallel Computation,* sponsored by the IEEE Computer Society and NASA, October 1992.

REFERENCES

1. Fox, E. A. Advances in interactive digital multimedia systems. *Computer* **24,** 10 (Oct. 1991), 9–21.
2. Goodman, J. R., Vernon, M. K., and Woest, P. J. Efficient synchronization primitives for large-scale cache-coherent multiprocessors. *Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-III)*, Apr. 1989, pp. 64–75.
3. *Grand Challenges: High Performance Computing and Communications.* Committee on Physical, Mathematical, and Engineering Sciences, National Science Foundation, 1991.
4. Hennessy, J. L., and Patterson, D. A. *Computer Architecture: A Quantitative Approach.* Morgan Kaufman, San Mateo, CA, 1990, p. 426.
5. Hillis, W. D., and Steele, G. L., Jr. Data parallel algorithms. *Comm. ACM* **29,** 12 (Dec. 1986), 1170–1183.
6. Kanal, L., Kumar, V., and Gopalakrishnan, P. S. (Eds.). *Parallel Algorithms for Machine Intelligence and Pattern Recognition,* Springer-Verlag, New York, 1990.
7. Smith, A. J. Cache memories. *Comput. Surveys* **14,** 3 (Sept. 1982), 473–530.

8. Snyder, L. Type architecture, shared memory and the corollary of modest potential. *Annual Rev. Comput. Sci.* **1** (1986), 289–318.

9. Streibl, N., Brenner, K.-H., Huang, A., Jahns, J., Jewell, J., Lohmann, A. W., Miller, D. A. B., Murdocca, M., Prise, M. E., and Sizer, T. Digital optics. *Proc. IEEE* **77**, 12 (Dec. 1989), 1954–1969.

10. Tuck, R. An optimally portable SIMD programming language. *Frontiers '88: The Second Symposium on the Frontiers of Massively Parallel Computation*, Oct. 1988, pp. 617–624.

HOWARD JAY SIEGEL is a professor and coordinator of the Parallel Processing Laboratory in the School of Electrical Engineering at Purdue University. He received two B.S. degrees from the Massachusetts Institute of Technology, and the M.A., M.S.E., and Ph.D. degrees from Princeton University. His current research focuses on interconnection networks, heterogeneous computing, and the use and design of the PASM reconfigurable parallel computer system (a prototype of which is supporting active experimentation). He is a Fellow of the IEEE and was a Coeditor-in-Chief of the *Journal of Parallel and Distributed Computing*. (Parallel Processing Laboratory, School of Electrical Engineering, 1285 Electrical Engineering Building, Purdue University, West Lafayette, IN 47907-1285)

SETH ABRAHAM is an assistant professor in the School of Electrical Engineering at Purdue University, West Lafayette, IN. He received the B.S. degree in computer engineering, the M.S. degree in electrical engineering, and the Ph.D. degree in computer science, all from the University of Illinois at Urbana–Champaign. His research interests include interconnection networks and high performance computer architectures. (Parallel Processing Laboratory, School of Electrical Engineering, 1285 Electrical Engineering Building, Purdue University, West Lafayette, IN 47907-1285)

WILLIAM L. BAIN is currently developing Block Island Technologies, a parallel software company. He worked at Bell Laboratories from 1978 to 1981 and at Intel Corporation from 1981 to 1992. He received his Ph.D. in electrical engineering from Rice University in 1978. His research interests include parallel computer architecture, concurrent programming, and parallel simulation. (Block Island Technologies, 15455 NW Greenbrier Parkway, Suite 210, Beaverton, Oregon 97006)

KENNETH E. BATCHER is a professor in the Department of Mathematics and Computer Science at Kent State University, Kent, OH. He received a B.S.E.E. degree from Iowa State University and an M.S. and Ph.D. from the University of Illinois. His current research focuses on parallel sorting algorithms, shuffle-exchange networks, and other interconnection networks. He received the ACM–IEEE Computer Society Eckert–Mauchly Computer Architecture Award in 1990. (Department of Mathematics and Computer Science, Kent State University, Kent, OH 44242-0001)

THOMAS LEE CASAVANT is an associate professor with the Department of Electrical and Computer Engineering at the University of Iowa. From 1986 to 1989, he was on the faculty of the School of Electrical Engineering at Purdue University, where he also served as director of the Parallel Processing Laboratory. He received the B.S. degree in computer science in 1982, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Iowa in 1983 and 1986, respectively. His research interests include parallel processing, computer architecture, programming environments for parallel computers, and performance analysis. He is a member of the IEEE Computer Society and the ACM. (Department of Electrical and Computer Engineering, University of Iowa, Iowa City, IA 52242)

DOUG DEGROOT is a senior member of the technical staff in the Advanced Technologies and Components Division of Texas Instruments. He received both a B.S. and a Ph.D. degree from the University of Texas at Austin. His current activities center on reconfigurable parallel processing architectures, architectural synthesis, and visualization design and analysis tools. He is coprincipal architect of the TI Tapestry/Aladdin parallel processor. DeGroot is past chairman of the ACM SIGARCH, and he is a subject area editor of the *Journal of Parallel and Distributed Computing* and an associate editor of the *IEEE Transactions on Parallel and Distributed Systems*. (Texas Instruments, Advanced Technologies and Components, 6550 Chase Oaks Boulevard, MS 8435, Plano, TX 75023)

JACK B. DENNIS is Professor Emeritus in the Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology, where he led a research group in computer system architecture that developed and refined concepts for dataflow computation. He has founded a company to develop and market dataflow computers for scientific computation. Professor Dennis earned his undergraduate and graduate degrees in electrical engineering at MIT. He is a Fellow of the IEEE and the 1984 recipient of the ACM–IEEE Computer Society Eckert–Mauchly Award for his work in advanced computer architecture. (Massachusetts Institute of Technology, Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139)

DAVID C. DOUGLAS is the leader of the Connection Machine system architecture group for Thinking Machines Corporation in Cambridge, MA. He received a B.S.E.E. and an M.S.E.E. from the Massachusetts Institute of Technology, and has contributed to the architectures of the CM-2 and CM-5 versions of the Connection Machine. He is currently working on future CM architectures. (Thinking Machines Corporation, 245 First Street, Cambridge, MA 02142)

TSE-YUN FENG is Binder Professor of Computer Engineering in the Department of Electrical and Computer Engineering at the Pennsylvania State University. He received his B.S. degree from National Taiwan University, M.S. degree from Oklahoma State University, and Ph.D. degree from the University of Michigan. His current research interests are in the area of parallel and concurrent processing, interconnection networks, and computer architecture. He is a Fellow of the IEEE and Editor-in-Chief of the *IEEE Transactions on Parallel and Distributed Systems*. (Department of Electrical Engineering, 121 Electrical Engineering East, The Pennsylvania State University, University Park, PA 16802)

JAMES R. GOODMAN is a professor of computer sciences at the University of Wisconsin at Madison. He received the B.S. degree from Northwestern University, the M.S.E.E. degree from the University of Texas at Austin, and the Ph.D. degree from the University of California at Berkeley. His current research interests focus on high performance computing, particularly memory systems and synchronization. He is currently on sabbatical at the Advanced Computer Research Institute (A.C.R.I.) in Lyons, France. (Department of Computer Sciences, University of Wisconsin at Madison, Madison, WI 53706)

ALAN HUANG is head of the Digital Optics Research Department at AT&T Bell Laboratories in Holmdel, NJ. He received a B.S. and M.S. in electrical engineering from Cornell University and a Ph.D. from Stanford University. His research interests have included computer architecture, VLSI algorithms, and broadband switching networks. His current interests involve optical digital computing and optical interconnection networks. (AT&T Bell Laboratories, Room 4G-514, Crawfords Corner Road, Holmdel, NJ 07733)

HARRY F. JORDAN is a professor in the Departments of Electrical and Computer Engineering and of Computer Science at the University of Colorado and program manager for digital optical computing at the Center for Optoelectronic Computing Systems. He received the B.A. degree from Rice University and the M.S. and Ph.D. from the University of Illinois. His interests in computer systems center on the interface between hardware and software with a focus on the application and performance of multiple instruction stream computers; an interest in optical computing and its effect on computer architecture has led to his involvement in a project to build and operate an optical, stored program, digital computer. He is a subject area editor of the *Journal of Parallel and Distributed Computing*. (Departments of Electrical and Computer Engineering and Computer Science, Campus Box 425, University of Colorado, Boulder, CO 80309-0425)

J. ROBERT JUMP is a professor in the Department of Electrical and Computer Engineering at Rice University. He received the B.S. and M.S. degrees from the University of Cincinnati and the Ph.D. degree in computer science from the University of Michigan. His research interests are in the area of parallel computer architecture with special focus on interconnection networks and the simulation of parallel systems. He has served as an associate editor of the *IEEE Transactions on Computers* and is currently serving as an associate editor of the *IEEE Transactions on Parallel and Distributed Systems*. (Department of Electrical Engineering, Rice University, P.O. Box 1892, Houston, TX 77251)

YALE N. PATT is a professor of electrical engineering and computer science at the University of Michigan, Ann Arbor, where he teaches graduate and undergraduate courses in computer architecture and directs Ph.D. students in experimental research on the implementation of high performance computer systems. He received his B.S. from Northeastern University and M.S. and Ph.D. from Stanford University, all in electrical engineering. His current research focuses on optimizing concurrency at the single-instruction stream level (exploiting the HPS paradigm) and at the tightly coupled multiprocessor level. He is a Fellow of the IEEE, an associate editor of the *IEEE Transactions on Computers*, and a member of the editorial board of *Computer* magazine. (The University of Michigan, Department of Electrical Engineering and Computer Science, 1101 Beal Avenue, Ann Arbor, MI 48109-2110)

ALAN JAY SMITH is a professor of computer science at the University of California at Berkeley. He received his B.S. in electrical engineering from the Massachusetts Institute of Technology and his M.S. and Ph.D. from Stanford University. His research interests are in computer system performance, particularly memory hierarchies. He is a Fellow of the IEEE, chairman of the ACM Special Interest Group on Computer Architecture (1991–1993), was chairman of the ACM Special Interest Group on Operating Systems (SIGOPS) from 1983–1987, was on the board of directors of the ACM Special Interest Group on Measurement and Evaluation (SIGMETRICS) from 1985–1989, was an ACM National Lecturer (1985–1986) and an IEEE Distinguished Visitor (1986–1987), is an associate editor of the *ACM Transactions on Computer Systems* (*TOCS*), a subject area editor of the *Journal of Parallel and Distributed Computing*, and is on the editorial board of the *Journal of Microprocessors and Microsystems*. He was program chairman for the Sigmetrics '89 / Performance '89 Conference and program cochair for the Second (1990) Hot Chips Conference. (Computer Science Division, 573 Evans Hall, University of California at Berkeley, Berkeley, CA 94720)

JAMES E. SMITH received the B.S., M.S., and Ph.D. degrees from the University of Illinois. He has been with Cray Research, Inc., in Chippewa Falls, WI, since June of 1989. At Cray Research, he heads a small research team that is participating in the development of future supercomputer architectures. (Cray Research, Inc., 900 Lowater Road, Chippewa Falls, WI 54729)

LAWRENCE SNYDER is a professor in the Department of Computer Science and Engineering at the University of Washington in Seattle. He was awarded a B.A. in mathematics and economics from the University of Iowa, and he received his Ph.D. in computer science from Carnegie–Mellon University. His research interests are parallel computer architecture, including routing and networks, and parallel programming languages and environments. He is inventor of the CHiP architecture and the Poker parallel programming environment, co-inventor of Chaos routing, a Fellow of the IEEE, and an associate editor of the *IEEE Transactions on Parallel and Distributed Systems*. (Department of Computer Science FR-35, University of Washington, Seattle, WA 98195)

HAROLD STONE is engaged in computer architecture research at the IBM T. J. Watson Research Laboratory in Yorktown Heights, NY. He has been on the faculty at Stanford University and the University of Massachusetts prior to joining IBM. He received a B.S.E.E. degree from Princeton University and M.S.E.E. and Ph.D. degrees in electrical engineering from the University of California at Berkeley. He was elected fellow of the IEEE in 1986, received the IEEE Emanuel R. Piore Award in 1992 for work in parallel computation, and is an associate editor of the *IEEE Transactions on Parallel and Distributed Systems*. (IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598)

RUSS TUCK is systems architect at MasPar Computer Corporation. He received his B.S., M.S., and Ph.D. in computer science from Duke University, and did his dissertation research at the University of North Carolina at Chapel Hill. His research interests include Autonomous SIMD (ASIMD) architectures and data parallel languages. His focus at MasPar is to architect future generations of massively parallel systems and help bring them into the present. (MasPar Computer Corporation, 749 North Mary Avenue, Sunnyvale, CA 94086)

BENJAMIN W. WAH is a professor in the Department of Electrical and Computer Engineering and the Computer and Systems Research Laboratory of the University of Illinois at Urbana–Champaign. He received his B.S. in EECS from Columbia University, M.S. degrees from Columbia University in EECS and the University of California at Berkeley in CS, and a Ph.D. degree from the University of California at Berkeley in engineering. He has published extensively in the areas of computer architecture, parallel processing, artificial intelligence, distributed databases, and computer networks. He is a Fellow of the IEEE, is Associate Editor-in-Chief of the *IEEE Transactions on Knowledge and Data Engineering* (Editor-in-Chief, as of January 1993), and serves on the IEEE Board of Governors. (Computer and Systems Research Laboratory, University of Illinois at Urbana-Champaign, 1308 West Main Street, MC225, Urbana, IL 61801)