Revitalizing Computer Architecture Research

Third in a Series of CRA Conferences on Grand Research Challenges in Computer Science and Engineering
December 4-7, 2005

Prepared by
Conference Co-Chairs

Mary Jane Irwin, Pennsylvania State University
John Paul Shen, Nokia Research Center – Palo Alto

Computing Research Association 1100 17th Street, NW, Suite 507 Washington, DC 20036-4632
URL: http://www.cra.org E-mail: info@cra.org Tel: 202-234-2111 Fax: 202-667-1066

Organizing Committee Members

Mary Jane Irwin, Penn State University (Co-Chair)
John Paul Shen, Nokia Research Center – Palo Alto (Co-Chair)

Todd Austin, University of Michigan
Luiz Andre Barroso, Google
Susan Eggers, University of Washington
Elmootazbellah Elnozahy, IBM
Mark Horowitz, Stanford University
Mike Johnson, Texas Instruments
Chuck Moore, AMD
Ravi Nair, IBM
Dave Patterson, University of California, Berkeley
Justin Rattner, Intel
Anand Sivasubramaniam, Penn State University

Acknowledgments: *Funding for this conference was provided by National Science Foundation Grant No. CCF-0537399.*

Contents

1. Introduction

Technology Outlook

Truly innovative research with the potential of making a profound impact should aim at a horizon of 10+ years. By 2015 the process technology available to computer architects will likely be the 16 nm or the 11 nm technology node with about 100-billion-transistor integration capacity per die. A synopsis of the ITRS roadmap presented by Intel's Shekbar Borkar at the conference is shown in Table 1 that highlights technology direction through 2018. By then, 3D stacking of multiple dies will be widely available which will facilitate extreme integration of building-block dies in diverse technologies. These building-block dies can include: CPU, GPU, *PU, SRAM, DRAM, Flash, MEMS, interconnect fabric, and so on. Unlike several decades ago when computer architects used to push the limits of technology, computer architects are now being pulled by the relentless advance of technology, and frequently do not know how best to exploit the available transistors and advanced integration possibilities.

Table 1. ITRS Roadmap Synopsis

	2006	2008	2010	2012	2014	2016	2018
Technology Node (nm)	65	45	32	22	16	11	8
Integration Capacity (BT)	4	8	16	32	64	128	256
Delay (CV/I scaling)	~0.7	>0.7	Delay scaling will slow down				
Energy/Logic Op (scaling)	>0.5	>0.5	Energy scaling will slow down				
Alternate, 3D etc.	Low probability						High Probability
Variability	Medium			High			Very High

Technology Challenges

Scaling beyond the 45 nm technology node introduces significant new challenges. In addition to the well-known power and thermal problems, new problems will arise. Both delay scaling and energy scaling will slow down, and device variability will significantly increase due to random dopant fluctuations, sub-wavelength lithography, and temperature variation and hot spots. Extreme variations will result in devices that exhibit unpredictable behaviors and that can degrade over time. Gradual errors due to variations, time-dependent errors due to degradation, and transient errors due to environmental factors all will conspire to make the design of reliable computing systems extremely difficult. Future lithography will require extreme regularity in the physical design in order to mitigate the negative effects of variation and reliability. One-time factory testing and the traditional burn-in process will no longer suffice. Continual self-testing and

self-monitoring with the ability to dynamically reconfigure and adapt will be essential features in future architectures.

Unprecedented Opportunity

Driven by the promising outlook and serious challenges of the process technology, computer architects have an unprecedented opportunity to be innovative. The purpose of this CRA Grand Research Challenges conference was to bring together 50 to 60 of the most visionary computer architecture researchers in academia and industry to brainstorm and identify the new challenging research problems for the architecture research community to address in the next 10 years. The intent was to think outside of traditional “boxes” and beyond current trendy topics. Separating computing and communication is no longer useful; differentiating between embedded and mainstream computing is no longer meaningful. Extreme mobility and highly compact form factors will likely dominate. A distributed peer-to-peer paradigm may replace the client-server model. New applications for recognition, mining, synthesis, and entertainment could be the dominant workloads. It was an opportune time for the computer architecture research community to spend a few days together to ponder these possibilities and to formulate a grand new research agenda.

2. CONFERENCE SUMMARY

Conference Logistics

The organizing committee received a total of 107 position statements from both industry and academia; based on these submissions, 33 invitations were extended to participate in this working conference. Including organizing committee members, invited industry keynote speakers and panelists, and other invited guests, there were a total of 55 participants (15 from industry, 33 from academia, and 7 from CRA and NSF) at the conference.

In addition to the plenary discussion sessions and the topic-specific breakout groups, there were two keynote presentations and an industry panel session. The keynote addresses were given by Shekhar Borkar of Intel speaking on "Microarchitecture Challenges for 2015," and Jim Larus of Microsoft Research addressing "Software Challenges in Nanoscale Technologies." The industry panel discussion was moderated by John Paul Shen, and included Bob Colwell (consultant), Chuck Moore (AMD), Ravi Nair (IBM), Justin Rattner (Intel), and Steve Scott (Cray) as distinguished panelists.

The first full day of the conference (Monday, December 5, 2005) began with the two industry keynote addresses, followed by the charge for the day's breakout sessions provided by Mary Jane Irwin. There were five breakout groups based on the following five topics: 1) Brain and Nano Computing; 2) Tolerating Extreme Scaling; 3) CMP and Beyond; 4) Self-Healing Systems; and 5) Software-Hardware Interface. At the end of the day all five groups convened in a plenary session where each reported out. The organizing committee then met to review the progress made that day and to plan the agenda for the next day.

The second day of the conference (Tuesday, December 6, 2005) began with the distinguished industry panel, followed by an open-mike session. The charge for the day was given by John Paul Shen prior to the breakout sessions. The breakout groups were organized based on the following potential grand challenge topics: 1) Ultra Energy Efficient Computing; 2) Usage of Abundance of Transistors; 3) Brain Architecture; and 4) Popular Parallel Programming. Again, a plenary report-out session was held at the end of the day, during which attempts were made to formulate specific grand research challenges. After the plenary session the organizing committee met again to review the day's efforts and plan for the final day.

The third and final day (Wednesday, December 7, 2005) began with the continuation of the breakout groups, followed by the final plenary session during which the list of grand research challenges was presented and

debated. The participants agreed on a handful of grand research challenges (summarized in chapter 3). All the slides produced during the conference were collected, and certain assignments were given to a few for follow-up work after the conference. A summary set of slides for the conference were produced from the slides developed by the attendees; these are provided in Appendix B.

Industry Perspectives

In the first keynote address, Shekhar Borkar of Intel focused on trends in technology scaling and the resultant impacts on computer architecture. While transistor count per die will continue to increase following Moore's Law, power and power density limitations will severely hamper the ability to continue to scale performance. The parallelism trend towards using more and more cores to achieve performance is inevitable. Furthermore, these cores must become more energy efficient by at least an order of magnitude. Device variation will be another severe problem. Extreme variations are expected starting with the 32 nm technology node and will continue to get worse. This has significant impacts on reliability, test methodology, and design style. Built-in system resiliency leveraging the available parallelism is absolutely essential.

The second keynote speaker, James Larus of Microsoft Research, focused on major software challenges associated with the emerging nanoscale technologies. The greatest challenge he identified is "taming concurrency." The imminent trend towards multiple and many cores on a die necessitates the availability and abundance of parallel software to take advantage of the widely available parallel machine resources. However, parallel programming has been an unsolved challenge for multiple decades. Concurrency is extremely difficult for a number of reasons: humans do not think concurrently; there is not a common parallel programming model; current concurrency primitives are inadequate; and parallel programs are very hard to analyze and to prove correct. A grand research challenge is to make concurrency usable by the masses, and encourage other uses for concurrency (e.g., reliability, responsiveness, security, safety, and user interactions).

During the industry panel on Tuesday morning, a number of different perspectives were presented by the distinguished participants. Bob Colwell challenged the computer architecture researchers to focus on addressing the needs and desires of diverse end-users of the technologies we develop. Chuck Moore (AMD) suggested the need to formalize abstraction layers for computer architecture, for example, "the 7-layer ISO model for computing." Ravi Nair (IBM) spoke of the need for full system modeling and rapid design and deployment of high-quality systems. Justin Rattner (Intel) highlighted

the difficulties of new platforms that challenge the existing ecosystem—new architectures tend to run far ahead of new software, and new implementations tend to run far ahead of necessary validation techniques. Steve Scott (Cray) emphasized that “one size does not fit all” and the need to exploit heterogeneity and provide ease of use.

A number of common themes were mentioned by most of the panelists. Parallel programming must be made easier and must become *the* mainstream programming method. Architects must design reliable systems even if the underlying components may not operate reliably. Achieving power/performance efficiency will require the use of heterogeneity and specialized building blocks. Having a global mindset with a focus on the end-users is crucial for successful product development; computer architecture researchers need to go beyond technology development to also address and explore compelling usage models.

3. Computer Architecture Research Challenges for 2020

Research challenges must be motivated by a long-term vision of the anticipated applications and usage models. We set our target at the year 2020, by which time we assumed that there will be ubiquitous Internet connectivity with heavy dependence on a variety of wireless connectivity. We also assumed that advancing technology will provide extreme forms of integration and powerful devices in extremely small form factors. With these assumptions, we divided our application vision into two domains: *human-centric* usage models and *infrastructure-centric* usage models.

Human-centric usage models will involve intelligent *spaces*, personal *agents*, and the interaction of spaces and agents. We envision intelligent spaces for work, education and leisure/entertainment. These spaces can incorporate active displays and other sensory devices to provide the immersive experience. The agents are essentially personal gadgets or devices that are quite feature rich and potentially provide significant stand-alone functions. These agents are also quite mobile and can roam seamlessly from space to space. The spaces can track the movement of the roaming agents and can proactively provide useful real-time information.

Infrastructure-centric usage models can include traditional server farms and data centers, as well as the fabric for supporting human-centric usage models. Very large scale information fusion, storage, and analysis are foundational. Providing communication and synchronization between intelligent spaces, and proactively pushing real-time and customized information to an extremely large number of roaming agents, are all part of the infrastructure-centric usage models. In addition to a large number of mobile agents, there could also be an enormous number of distributed and embedded sensors that are part of the overall fabric.

Given this “System 2020 Vision,” what can computer architecture researchers contribute to it? This vision clearly involves architecting of computing and communication systems at an extremely large scale. There are massive information storage, migration, caching and consistency issues. The traditional tradeoffs and balancing between computation and bandwidth are simply magnified by many orders of magnitude. Whether the roaming agents should be very thin clients tightly supported by servers at massive data centers or rich-featured clients occasionally backed up by servers is an interesting question. There are also massive programming issues. This is beyond the already difficult problem of parallel programming for individual multicore chips; this involves the programming of many multicore chips that

are networked with other systems near and far, large and small. How do we manage such huge amounts of parallelism, with mixed programming capabilities, huge variations of latencies of the underlying systems, and massive reliability issues? These are certainly grand research questions and challenges. The four specific challenges that emerged from the conference—a featherweight supercomputer, popular parallel programming, systems you can count on, and new models of computation—are detailed below.

A Featherweight Supercomputer

Most technologists believe that the technology trend characterized by Moore's Law will continue for at least another decade. At the heart of this trend is the ability to continue to integrate more and more devices and, hence, functionality on one chip. In addition to integration on a single die, integration involving multiple dies will become prevalent. We have had multi-die integration on a single substrate in the form of SOC (system on chip), as well as stacking of multiple dies via wire bonding in the form of MCP (multi-chip package). The next step is the direct 3D stacking of multiple dies using TSV (through-silicon vias). By 2020 we can expect such aggressive forms of integration to produce extremely powerful systems in very small form factors.

An admirable goal for 2020 could be to produce a supercomputer capable of one teraops of computation throughput in a single package that consumes only one watt—that is, 1 teraops/watt. One teraops/watt translates into an energy efficiency of 0.001 nanojoules/op. The current generation of high-end microprocessors (deeply pipelined out-of-order designs) consume about 30 to 40 nj/op running SPECint. The more recent designs that attempt to achieve much better performance/watt are getting close to 10 nj/op on SPECint. To get to 0.001 nj/op requires another *four orders of magnitude* improvement on energy/op (EPO). Fortunately we have examples of current embedded processors and DSPs that have EPOs of less than 1 nj/op and special-purpose engines with even lower EPO. Technology scaling will improve EPO and more parallel workloads can also reduce EPO; however, these will not be enough to achieve the multiple orders of magnitude improvements required. EPO optimization for achieving ultra-energy-efficient processors constitutes one major research challenge for computer architects and processor designers in the coming decade.

The potential application and impact of the 1 teraops/watt compute building block ("featherweight supercomputer") can be huge. We envision at least four major areas of impact: 1) personal all-purpose device; 2) pervasive intelligent sensors; 3) embedded supercomputing appliances; and 4) energy-efficient data centers.

The personal all-purpose device can be the size of our current cell phones, but contains all the functionalities that we currently have in our laptop computer, cell phone, MP3 and video player, PDA, portable game machine, garage-door opener, car keys, credit cards and TV remote controls. This device can roam seamlessly across all wireless domains, including 3G, Wi-Fi, Wi-MAX, Bluetooth, and UWB. Whenever it comes within proximity of other devices, automatic recognition and authentication can occur, possibly followed by automatic synchronization and file transfers. Messages, media contents, data files, and software can be transparently pushed to these devices from the server. Location- and circumstance-aware information filtering can be automatically performed. If desired, each device can even function as a mobile web server by making its own content accessible on the Internet.

One example for pervasive intelligent sensors is video surveillance in public spaces. Instead of piping all the video streams from distributed cameras to a centralized processing center, local processing of the real-time video can be performed at the camera for quick detection of intrusion and suspicious activities. Similar systems can be deployed for monitoring children and the elderly for emergency events. Sophisticated video analytics are needed for processing the real-time video streams and for activating immediate response.

Embedded supercomputing appliances can be employed in numerous application domains. One example is in medical applications where they can be used for real-time mobile/wearable diagnostics and monitoring.

Large data centers are becoming the norm, not the exception. Witness the new server farm installation being constructed by Google that is rumored to be able to accommodate thousands of large servers. Such large data centers are becoming a major source of power consumption. One now famous quote by Eric Schmidt, CEO of Google, is that: “What matters most to the computer designers at Google is not speed, but power—low power, because data centers can consume as much electricity as a city.” The featherweight supercomputer can be used as the building block for constructing highly energy-efficient data centers.

In addition to significant reduction of EPO, there are other research challenges for achieving the one teraops/watt featherweight supercomputer. The power consumption of one watt is also a proxy for other design constraints—battery life and system cost, for example. The EPO requirement of 0.001 nj/op also implies very efficient microarchitecture design executing very-well-behaved software. The microarchitecture will likely incorporate heterogeneous cores and programmable accelerators. Highly efficient mapping of application software to the underlying hardware resources would

be essential. The software stack must accommodate diverse forms of concurrency, real-time responsiveness, fault- and error-tolerance, and the ability to dynamically adapt computation and communication based on connectivity and available network resources.

Popular Parallel Programming

Parallel programming is not a new research problem; there has been an entire research community devoted to it for at least three decades. So why is this old problem a major research challenge for the computer architecture community in the coming decade? There are several ways to answer this question. First, solving this problem has become an absolute necessity. We are at a dead end on the single chip performance curve without integrating multiple cores on one chip. The alternative is to continue to aggressively scale the clock rate of single processor chips, but this has serious power consumption implications. Given the technology trends and the necessity for multicore chips in order to stay on the performance curve, parallel programming is now a *must*.

Second, the pervasive availability of multicore chips today and many-core chips in the not-too-distant future provides the necessary economic impetus for the development of parallel software. For the last several decades, highly parallel machines have only been available to a very small sector of computer users. Now, we can have a parallel computer on every desktop and in every laptop. Sure, one can say that just because you build it doesn't mean that they will come. An obvious, but far from ideal, solution is to run all the software on one of the chip's core and put the other cores into sleep mode. This will keep the chip within the power budget, but certainly won't provide the performance growth that users have come to demand from next-generation parts.

The grand challenge is to make parallelism pervasive and parallel programming mainstream in order to enable software to make effective use of the widely available parallel hardware and continue the performance improvement trend of the past several decades. Parallel software is the key to unleashing the performance potential of parallel machines. The goal is to reach the point where, when we say 'computers' we naturally mean parallel computers, and when we say 'programming' we naturally mean parallel programming. The goal is to make parallel programming accessible to the average programmer. The development of parallel software should be a core component of the undergraduate computer science and engineering curriculum.

Pervasive parallel programming will need languages for expressing parallelism, parallel programming models, parallel algorithms and

development environments, runtime environments, and associated architecture and microarchitecture support. Programmability, composability, and correctness are essential. Current new approaches that are promising include transactions and streams. The entire infrastructure should support multi-modal parallelism, including: 1) fine-grain data-parallelism, 2) embarrassing parallelism with completely independent tasks, and 3) irregular parallelism with medium-sized tasks with dependencies. This third type of parallelism is the most challenging, and will require new innovative solutions that can facilitate scaling to dozens of cores and hundreds of concurrent threads. Microarchitecture and hardware support for parallel program debugging, lightweight thread communication, performance tuning, and dynamic adaptation and optimization are crucial contributions that the computer architecture research community can definitely provide.

In addition to improving performance, parallelism can and should be exploited for other benefits. Given the anticipated extreme variations in device behavior and the increasing susceptibility of the underlying technology to soft and transient errors, the ability for a system to detect and tolerate these undesirable effects will be absolutely essential. Parallelism can be leveraged to achieve such reliability and robustness goals. Core sparing, thread migration, redundant threads, and threads whose job is to monitor the health and/or temperature of the die are just the basic starting points for this new path of research. We may need to consider architecting attribute-based resources, where the attributes can include reliability and robustness status bits that the runtime system can use for dynamic resource adaptation and reconfiguration.

Systems You Can Count On

Device unreliability is projected to be a key barrier to meeting the promise of Moore's Law over the next decade. As technology scales further, hardware will face numerous sources of errors including process variations, high-energy particle strikes, aging, insufficient burn-in, thermally induced timing errors, and design bugs. At the same time, software is becoming increasingly complex and its robustness continues to be challenging. According to a 2002 report from NIST, software defects cost the U.S. economy an estimated $59.5 billion annually or 0.6% of the GDP. This cost will worsen as average developers write parallel programs to exploit multicore chips. Further, as computing devices are increasingly networked, they are vulnerable to malicious attackers who can steal information in (or accessible from) a device, corrupt it, and access and compromise services of the entity they are impersonating.

As computers become an integral part of our personal environment and societal infrastructure, their dependability (including reliability and security)

should no longer be an afterthought or a luxury afforded only by high-end niche systems. Instead, dependability must be a first-class design constraint for all systems and across the system stack. Unfortunately, designing dependable systems is harder than ever. It is inevitable that shipped hardware and software will have or develop defects. No matter how secure we design our hardware and systems, vulnerabilities will be discovered by clever and determined attackers. Future systems must detect these defects and vulnerabilities and work around them dynamically, at low cost.

Thus, a third grand challenge is to design hardware and software systems that users can count on by providing self-healing, trustworthy hardware and software systems. What role can architects play in addressing this grand challenge? As we develop new solutions for hardware reliability, we have the opportunity to address the fragility of the entire system from the ground up in an integrated and holistic way, together with the software communities working on software dependability. Sitting at the hardware-software interface, architects have the opportunity to define how commodity *systems* can be architected for dependability. Our grand challenge, therefore, is not about incremental or piecemeal hardware solutions to reliability problems, but a rethinking of the entire system stack to make system reliability and security a first-class design objective and judiciously sharing this responsibility between the hardware and the software. Traditional solutions will not suffice; architects *must* be involved in developing new approaches for several reasons.

First, reliability and security concerns now pervade everyday systems (embedded, desktops, and servers) that are already dominated by other constraints such as cost, area, performance, and power. Traditional dependability solutions have not been concerned with power and cost, performance, and area concerns have also been less stringent. This has led to hardware reliability solutions that are overly conservative, and, hence, too expensive (e.g., extensive redundancy). Software reliability practices are often eschewed by developers due to their high performance overheads. Similarly, low-level hardware reliability solutions from the circuits' community tend to be conservative and influenced by worst-case scenarios that rarely happen for real workloads.

In contrast, architectural approaches can be far more efficient. They can be tailored for anticipated problems and for real workloads, possibly at the cost of lower, but acceptable, fault coverage. Workload-aware dynamic resource management can reduce vulnerability to soft errors and slow aging. Tight architectural integration has resulted in novel checker processors that are far cheaper than traditional replication of the processor. Judicious use of available redundancy (idle cycles, functional units, cores, storage) can boost reliability in a cost-effective way.

For software dependability, since hardware ”owns” the program state, it is natural and more efficient to provide architectural support for monitoring and logging this state. Software-controlled hardware monitoring of executions can detect difficult software bugs for sequential and concurrent programs (e.g., timing-related bugs) orders of magnitude faster than software-only solutions. Coupled with hardware-assisted fast checkpoint/ recovery, hardware support can enable software bug detection and recovery during production runs in ways not feasible before. Hardware support can also provide or enable better programming abstractions and isolation to reduce the probability of software defects, error propagation, or virus contamination across different software. However, existing architectures provide little to no support for software dependability.

Many of the above techniques can also be employed for efficient hardware checking of possible security violations. Many security vulnerabilities are essentially software bugs. Additionally, hardware support for trusted paths, authentication credentials, secure attention key, and so on can be much more efficient than software-only solutions. Hardware support for security is a nascent but growing research area that has only scratched the surface of possibilities.

There is another reason that architects must be key players in designing systems we can count on. Because it will be impossible to ship fully tested products, systems must test themselves in the field, detect or anticipate failures, and reconfigure themselves with low overhead. There has been much work in reconfigurable architectures for performance, energy, and thermal management. It is similarly natural to consider architectural reconfigurability for reliability and security. To provide effective configurability for dependability will require unifying the control loops already present for performance, energy, and temperature with reliability and security adaptation.

A third reason is that different applications have different needs and notions of dependability (e.g., media vs. text search vs. health monitoring), meaning that different tradeoffs in performance, cost, and power must be made. To effectively exploit the opportunities available through such tradeoffs requires generic interfaces to express these application-level dependability needs. Architects must be involved to develop the right hardware-software interface to deliver the right reliability and security level at the right performance and power. Further, architects need to design hardware to dynamically respond to these needs. This is a significant shift from the past “all or nothing” philosophy to a more quality of service (QoS)-like philosophy that provides a spectrum of possible dependability driven QoS points.

The software and hardware dependability problem cuts through multiple system layers and so should the solution. This is an opportunity for architects to work closely with other designers to truly affect the way systems are built from the ground up, including: designing a dependability-aware hardware-software interface; dependability-aware resource management; cooperative failure avoidance, detection, and recovery mechanisms that span circuits to applications; cooperative hardware/ software cross-layer policies to trade off performance, power, reliability, and security; and workload-aware, rigorous dependability analysis tools.

New Models of Computation

Essentially all commercial processors built today are based on principles first proposed by John von Neumann in 1945. Using the so-called "stored program concept" processors fetch and then execute instruction from their memory hierarchy. Many of the grand challenges facing the computer architect in the '70s, '80s and '90s had to do with trying to solve the "memory wall" that grew ever taller due to the rapidly increasing performance disparity of the main memory relative to the processor core. Typical cycle comparisons for today's machines are 1:100—that is, it takes 100 processor cycles to fetch an instruction or data item from main memory. While work to solve the memory wall continues, the grand challenge proposed here is to develop promising new models of computation that are *not* von-Neumann-based and, thus, do not encounter the performance limitations of having to fetch instructions from a memory prior to execution.

Several past architectures have been proposed, and some even implemented, that are not von Neumann based. These include data flow machines and artificial neural networks (ANNs). To quote wikipedia "dataflow architectures do not have a program counter or (at least conceptually) main memory and the executability and execution of instructions is determined based on availability of input arguments to the instructions." Artificial neural networks model some of the properties of biological neural networks. They are composed of (hundreds of) thousands of simple processors each of which generates an output signal based on its weighted inputs and its "activation" value. Neural processors are connected to each other in special ways with each connection having an individual "weight". Networks learn by changing the weights of the connections. While interesting prototypes of both dataflow and ANN processors have been built, neither of these has achieved commercial success or wide usage.

It is probable that new technologies—nanotechnologies (e.g., carbon nano tubes [CNTs], quantum cellular automata [QCAs], etc.)—will *demand* new models of computation. These new technologies will certainly require innovations across the entire computational stack, including

microarchitectures, execution models, instruction sets, compilation algorithms, languages, and programming models. As such, it will be important to design full systems that can successfully integrate the innovations crossing disciplinary boundaries. Because they are so different from conventional CMOS devices, these nanotechnologies, which are currently only on the distant horizon, will likely inspire novel architectures that differ significantly from the traditional, von Neumann architectures.

The most powerful computing machine of all is the human brain. Is it possible to design and implement an architecture that mimics the way the human brain works? Artificial neural networks are only one small step in this direction. Neuroscientists are just beginning to unravel the workings of the human brain. It will be many years—perhaps many (many) decades—before we understand the workings of the human brain well enough to design and build hardware that mimics its functionings. Thus, there is great risk in this last grand challenge if it is cast as “brain computing,” but even greater return if it *does* turn out to be feasible.

Partial steps in this direction would be building systems that augment the brain, as in prosthetics (hearing for the deaf, vision for the blind, mobility for the quadriplegic), and in augmenting brain functions (enhancing the senses, inserting images and memories into the brain, perhaps in brain-to-brain communication). Here the risk is small; indeed there is progress already in human augmentation while the return is large.

4. Follow-Up Action Items

Report Out

This report, along with an accompanying set of slides (see Appendix B), will be made available by CRA to the broad computer science and engineering research community. A preliminary draft of the report was distributed to computer architecture researchers and developers at the annual International Symposium on Computer Architecture (ISCA) in June 2006. Additionally, an evening panel session presented the vision 2020 and resulting grand challenges to ISCA conference attendees. The goal for this panel presentation was to expose and engage the entire computer architecture research community on the research vision and challenges that came out of the CRA conference. Feedback from ISCA attendees was subsequently integrated into the report draft and the accompanying slides.

Research Infrastructure

One of the original objectives of the CRA GRC conference was to reignite a sense of excitement for the entire computer architecture research community. Vital support components for the research needed to fulfill the grand challenges proposed in this report are the next-generation tools, benchmark programs and data sets, and the implementation/emulation platforms necessary for experimentation and evaluation. The days of individual researchers designing and implementing prototypes including custom hardware and hand-crafted software are over except for the rare case. Thus, research infrastructures that can be used by many researchers—in particular low-cost, reasonable performance emulation platforms—are one very tangible way to bring the community together and to leverage new experimental infrastructures to embark on new research directions and generate new vitality for architecture systems research.

Funding Strategy

Another objective of the CRA GRC conference was to stimulate new and increased funding for computer architecture research. Other than distributing this report and the ISCA slides to the various funding agencies, are there more effective means to engage the funding agencies? Other than government funding agencies, how can we approach the industry to invest more in funding computer architecture research in academia? New funding models may have to emerge.

Appendix A. Conference Participants

Sarita Adve
University of Illinois
Urbana-Champaign

Arvind
MIT

Todd Austin
University of Michigan

Luiz Andre Barroso
Google

Andy Bernat
Computing Research Association

Shekhar Borkar
Intel

Doug Burger
University of Texas at Austin
Austin

Bradley Calder
UC San Diego

Bill Carlson
IDA/CCS

Fred Chong
UC Davis

Almadena Chtchelkanova
National Science Foundation

Robert Colwell
Colwell, Inc.

Patrick Crowley
Washington University
St. Louis

Bill Dally
Stanford University

Susan Eggers
University of Washington

Elmootazbellah Elnozahy
IBM

Michael Foster
National Science Foundation

Antonio Gonzalez
Intel, Barcelona

Dirk Grunwald
University of Colorado
Boulder

Mark Horowitz
Stanford University

Mary Jane Irwin
Penn State University

Mike Johnson
Texas Instruments

Lukas Kencl
Intel Research, Cambridge UK

Peter Kogge
University of Notre Dame

Christos Kozyrakis
Stanford University

Jim Larus
Microsoft Research

Edward Lee
UC Berkeley

Steve Levitan
University of Pittsburgh

Margaret Martonosi
Princeton University

Chuck Moore
AMD

Todd Mowry
Intel Research
Pittsburgh & CMU

Ravi Nair
IBM

Mark Oskin
University of Washington

Dave Patterson
UC Berkeley

Li-Shiuan Peh
Princeton University

Timothy Pinkston
University of Southern California

Justin Rattner
Intel

Steven Reinhardt
University of Michigan

Steve Scott
Cray Inc.

John Paul Shen
Nokia Research Center – Palo Alto

Anand Sivasubramaniam
Penn State University

Kevin Skadron
University of Virginia

Jim Smith
University of Wisconsin
Madison

Sean Smith
Dartmouth College

Gurinder Sohi
University of Wisconsin
Madison

Christof Teuscher
Los Alamos National Lab

Josep Torrellas
University of Illinois
Urbana Champaign

Peter Varman
National Science Foundation

T.N. Vijaykumar
Purdue University

Uri Weiser
Intel

David Wood
University of Wisconsin
Madison

Katherine Yelick
UC Berkeley

Yuanyuan Zhou
University of Illinois
Urbana-Champaign

Appendix B. PowerPoint Slides

———————————————————————————————————

The PowerPoint slides are best viewed in slideshow mode and are available online at
http://www.cra.org/Activities/grand.challenges/architecture/slides.pps

System 2020: Research Challenges in Computer Architecture

What is the next big thing?

What are the mega trends?

What are the anticipated usage models?

The computing paradigm ala Google

The computing paradigm ala Nokia

CRA System 2020 Workshop

What are the components of a Grand Challenge?

1W Featherweight Supercomputer

Featherweight Challenges

Popular Parallel Programming (P^3)

P^3 Challenges

Dependable Systems

Dependable Systems Challenges

New Computing Models

“Brain” Challenges

Check out previous Grand Challenges Conference Reports