Workshop on Advancing Computer Architecture Research (ACAR-II)

Laying a New Foundation for IT: Computer Architecture for 2025 and Beyond

Organizers: Mark Oskin (University of Washington) and Josep Torrellas (University of Illinois).

Steering Committee: Chita Das (Pennsylvania State University), Mark Hill (University of Wisconsin), James Larus (Microsoft Research), Margaret Martonosi (Princeton University), Jose Moreira (IBM Research), and Kunle Olukotun (Stanford University).

Written by: Mark Oskin, Josep Torrellas, Chita Das, John Davis, Sandhya Dwarkadas, Lieven Eeckhout, Bill Feiereisen, Daniel Jimenez, Mark Hill, Martha Kim, James Larus, Margaret Martonosi, Onur Mutlu, Kunle Olukotun, Andrew Putnam, Tim Sherwood, James Smith, David Wood, Craig Zilles

Funded by the Computer Research Association (CRA) Computing Community Consortium (CCC) as a “visioning exercise” meant to promote forward thinking in computer research and then bring those ideas to a funded program.

Held on September 20-21, 2010 in Seattle, Washington

Contact: oskin@cs.washington.edu; torrella@illinois.edu

Website: http://www.cra.org/acar.php

Executive Summary

Computer architecture is a foundation of society's Information Technology (IT). This foundation, however, was laid in the mid-20th-century when resources and aspirations were tens of thousands of times more modest than they are today. In some ways this foundation was for a bungalow and now we are building skyscrapers on top of the very same foundation. Through decades of technology and intellectual advances in computer architecture design, this bungalow has been extended and built upon to fuel the exponential growth in the IT economy,which has spurred the wider economy and societal changes at large.

That bungalow foundation has reached its limits, the better half of a decade ago. To continue to deliver performance improvements, industry has turned to largely evolutionary multicore designs. While this approach is taking us to new capability levels, it has its own limitations: much parallel software is unlikely to performance-scale indefinitely following Moore's Law, and important applications, whole or in part, are not amenable to it. Thus continued ILP efficiency scaling is required.

On September 20-21 a group of researchers met in Seattle to address one seemingly simple question: what now in instruction-level-parallelism (ILP) research? The consensus, articulated here in this report, is a broad and radical research agenda for computer architecture for 2025 and beyond.

First, **the fundamental boundary between hardware and software must change**, as the workshop attendees believed that the necessary performance gains **cannot be achieved through advances in conventional microarchitectures (**Section 3). Old debates such as RISC vs. CISC, or VLIW vs. Superscalar are moot. Instead, we argue in this report that research questioning the very foundations of the execution model is required. Questions such as: Is there a need for the current rigid boundary between software and hardware? Do processors need precise state? Is the von Neumann model the right approach? Should memory be accessed sequentially and with no high-level knowledge of program usage? The answers to these questions were last settled over half a century ago under extremely different technological constraints: scarcity of logic resources versus abundance; abundance of power versus too little to power entire chips we could fabricate; limited software infrastructures, compared to millions of lines of code in complex applications; no consideration of security beyond a handshake, compared to new systems broken into within minutes of being switched on and plugged into the Internet.

Second, computer architectures must support **new capabilities, including significantly more secure and reliable systems** from the ground up rather than as an add on (Section 4). As with instruction set architecture, the core architectural construct (page-based virtual memory) used to build security into modern systems was designed nearly four decades ago, under drastically different technological constraints and use cases. Because of the limited architectural mechanisms used to provide these services, the last two decades have seen the growth and significant impact from viruses, spyware and malware. Unless mechanisms are developed for isolation, secret containment, and computation on distrusted third parties, new use cases, such as cloud computing, will not reach their full potential. People and companies need firmer guarantees than contract-law, ensuring that data stored and computation performed by third parties is secure, reliable, and what was asked for.

Finally, **continued research into architectures for emerging technologies is needed to make them possible successors to conventional technologies whose improvement is stalling** (Section 5). Architects play a vital role in the advancement of new technologies by bridging the gap between theory and practice. Computer architects can devise use cases for upcoming technologies and articulate the constraints these technologies must work under to be deployable.

These research initiatives are best tackled through academic research. The breadth and depth of change to fundamental architectural precepts exceeds the comfort zone of any company; indeed, it exceeds the comfort zone of many academics. But there are academic researchers who can take a radical departure

from the status-quo; who can question the basic precepts of our field. To do so, however, requires significant and sustained investment by the federal funding agencies. This report outlines specific recommendations to these agencies in order to make progress on the research thrusts described in this report and prevent them from "dying on the vine" (Section 6).

The workshop attendees submit this report to Federal funding agencies and to their own research community, and exhort them to study it carefully. We must rebuild the foundation of our field or we will be trapped beneath the bungalow foundation we laid a half century ago: Advancement in conventional computer architectures is over!

1 Introduction

The degree to which information technology has impacted society cannot be understated and computer architecture has played a foundational role in enabling that revolution. Computer architecture has been the engine for translating Moore's Law technology improvements into improved software performance for nearly 50 years. Increasing transistor budgets have enabled increasingly sophisticated processor implementations, and by virtue of an abstraction layer between the software and the hardware – the instruction set architecture – the resulting performance benefits have been available with little, if any, effort from the programmer or end user.

With computer architecture being so central to IT's success, it is important for it to envision future needs and trends and respond to them. Computer architecture currently faces a turning point that calls for dramatically new techniques and approaches. This turning point is caused by several key trends:

- **Technology Drivers**: On the one hand, technology trends will change the shape of future computer architectures. In particular, the progressive divergence between real scaling and classical (or Denard) scaling makes it increasingly difficult to extract more computer performance by employing more transistors on-chip. Power limits and reduced semiconductor reliability will make scaling more difficult to leverage in the future. At the same time, there are emerging new technologies for logic devices, storage, and interconnects that can be game-changers in the future.

- **Application Drivers**: Computer architectures must also respond to the changing nature of applications and software. Recent years have seen the emergence of rich multimedia applications and data-intensive computing, as well as extensive growth in both mobile/embedded computing at one extreme, and warehouse-scale data center computing at the other. All these call for rethinking the simple architectural interfaces (load, store, compute) that have remained largely unchanged for decades.

- **Metrics and Goals:** A final important trend calling for architectural attention concerns the metrics and goals by which computer systems are evaluated. While computing has until recently been almost exclusively performance-focused, it is now abundantly-clear that other metrics demand attention. In addition to performance, computer systems are expected to be reliable, to be secure, and to operate with manageable and malleable power budgets. Achieving these goals requires extensive hardware and software cooperation, and thus it becomes vital that the computer architecture be designed with appropriate mechanisms and interfaces to facilitate this cooperation.

Drawing an analogy to the architecture of buildings, information technology/computing cannot continue to add floors to a skyscraper with a foundation designed for something much smaller. Rather, society's computing needs call for us to create new computer architectures designed from the ground up to support the diverse and challenging requirements of current and future computer systems. The foundations of this next-generation computer architecture include:

- First-class support for security, programmability, and reliability.

- New fundamental mechanisms to support performance that is *scalable* (back to Moore's Law), *understandable* (for applications where predictability is vital), and *boundable* (for real-time requirements).

- New mechanisms to support execution that is likewise *efficient, predictable*, and *boundable* in its use of energy as well.

Many research thrusts can be funded to help make the above vision a reality. In particular, this report focuses on three we find especially compelling.

- Beyond Traditional HW/SW ISAs: While traditional ISAs have served us well, they have become increasingly confining and have stagnated innovation in computer architecture. They lack the richness to express programmer and/or compiler intent, requiring the hardware to re-discover even a fraction of what was known at compile time. Furthermore, they force decisions to be made statically that would be better made at run-time. By re-defining the HW/SW boundary, and perhaps separating a logical ISA (the interface for distributing programs) from the implementation ISA (the instructions that the machine actually executes), we provide the opportunity for new levels of efficiency and for growing the role of the processor. Section 3 identifies our research agenda in this area.

- New Goals and Metrics: With computer systems in wide use throughout business and critical infrastructure, a performance-only focus is hopelessly inadequate. Rather, computer systems security and reliability are paramount, and computer architectures must play a role in supporting this. Likewise, energy-efficiency and predictability are also central goals. Section 4 outlines research opportunities in this area.

- Harnessing Technology Trends: As CMOS semiconductors—the traditional sources of technology-driven performance scaling—lose ground, it falls to the computer architects to seek out new opportunities and harvest from the rich set of technology alternatives to find options that will offer appropriate performance, reliability and other attributes. Computer architects can then integrate them into systems that take advantage of their best performance, power, and reliability attributes, while shielding higher system layers from accompanying deficiencies. Section 5 identifies our research agenda related to these topics.

Why Academic Research into Computer Architecture?

We believe that key advances in these areas are likely to come from academic research because:

- Many large industry players have significant economic incentives for retaining existing interfaces, which prevents the kind of revolutionary re-thinking of these interfaces required.

- The holistic kind of thinking required for the above research thrusts generally spans boundaries between companies.

- Academic research can take a longer-term view than industry.

- As we get fewer and fewer, larger and larger players in the computing industry, the diversity of research agendas from academia becomes a more important complement to the advanced development work done in industry.

Recommendation for Federal Funding

This report includes specific recommendations for the funding agencies, in order to make progress on the research thrusts described (Section 6).

Societal Impact

The research plan we discuss in this report has the potential for sweeping societal impact. With computer architecture serving as the keystone of computer systems hardware and software, rethinking it to treat performance, power, security and reliability as equal first class design goals, has the potential to ignite a second IT revolution. In particular, providing a stable foundation to build the next-generation of

embedded, mobile, desktop, and warehouse-scale computers will allow IT infrastructure to rely on high-performance computer systems that are also reliable, secure and energy-efficient.

To our own field, it is exciting to envision the significant impact of removing the HW/SW ISA restrictions we have too often placed on ourselves. One can only imagine the sorts of creative and high-impact ideas that could follow. With this, the educational impact can also be significant, as our students are trained to think big (without the traditional ISA constraints) while also maintaining computer architecture's long-standing traditions of finding simple, elegant, and effective solutions to pressing problems.

2 Applications

In this section we outline the applications and features that motivate the need for research into the areas outlined in the rest of this report.

Single-Thread Performance

Despite the availability of commodity multicore processors, software that utilizes them remains far from widespread. Not only is there a large installed base of single-threaded components, Amdahl's law will ultimately limit the performance scalability of even those applications tuned to utilize parallel cores. For this reason, single-threaded performance is still of the utmost importance to the future of the IT industry and scientific advances that rely on computation.

Additionally, some applications are very difficult to parallelize or contain a significant inherently-sequential component. Specifically, many graph-theoretical problems have significant sections where single-thread performance is crucial. For instance, determining the length of the shortest paths between all vertices in a graph (i.e., the all-pair-shortest-path problem), is difficult to parallelize on multicore architectures. Similarly, problems such as the circuit value problem and the related Boolean formula satisfiability problem are not parallelizable in general --- i.e., they are P-complete.

Security

Many different types of applications wish for security and privacy. Electronic voting is one example that has received considerable attention. On mobile platforms, users are increasingly concerned that potentially-sensitive information such as their current location (identifiable by a cellphone's GPS, for example) is only used for the intended purpose, and is not shared broadly without their consent.

Fundamentally, users and programmers want to have a good understanding of the answers to the following four questions:

- Where do the input bits come from?
- What modules have access to them?
- Is my code unchanged? Do the instructions I intended to execute, actually get executed?
- Where do my output bits go?

System support for some aspects of these problems has already been explored. Much of security employs cryptography, requiring hardware accelerators, which are currently under investigation. Likewise, proof-carrying code is also being explored, although, potential opportunities remain in the area of accelerating the phase that checks that the proof applies to this code before execution.

Fundamentally, however, significant novel challenges and opportunities remain. For example, taint analysis can track the flow of data in some instances, and hardware support for it can make it more efficient. Memory technologies that ensure a destructive read operation can form the foundation for assurances to software that data will be unreadable after a certain point in the program. New memory instructions and operations that better certify the flow of data may also support taint tracking.

In the extreme, side-channel attacks that detect system behavior based on performance counter readings or power dissipation represent particularly challenging threats. Hardware and system responses could help mitigate these threats through careful isolation and/or by inserting "white noise" activity that equalizes execution signatures seen by performance counters or power dissipation.

Cloud Computation

A fundamental barrier to the widespread adoption of cloud computing, with its accompanying benefits of business flexibility and power efficiency, is the issue of trust – that a cloud platform provider (e.g. Amazon Web Services, Google App Engine, or Microsoft Azure) will not read or corrupt a customer's data or examine or affect a customer's computation. Businesses are nervous trusting their corporate assets in the absence of technological mechanisms to protect them. Threats to these assets come less from the platform providers, currently large companies with little incentive to violate their customers' trust, than from third parties running on the same services. This threat may also shift as cloud computing becomes more of a commodity and new parties start providing this service. Computer architecture can provide a hard level of isolation to ensure that a computation executes in complete isolations from both the underlying system software and other users.

The key to achieving this goal is an execution model for a user-level process that is isolated from the system software (hypervisor, OS, applications) as well as other concurrently executing processes and applications. This is a fundamentally new security construct that current architectures were never designed to handle. Traditional systems trust the operating system, but not user processes. Architectures and systems will have to be built quite differently in a future where neither can trust each other. As a starting point, the processor must provide trusted computing hardware, similar to today's TPM, capable of computing a secure cryptographic hash of the process's state, to ensure that the customer's code and data start execution unmodified. The system, software and hardware, should not be able to examine or modify the state (memory, register, processor internal) of the user process after it starts executing. Finally, the processor must provide a secure, isolated location for storing cryptographic keys, accessible only to the application that uses the keys.

None of this functionality can be provided in software, both because the system software is controlled by the service provider, but also because software is mutable and subject to attacks that could compromise the integrity of the guarantees. Conversely, while burning such functionality into hardware is a requirement, the potential for errors exposes such designs to attack. Note the multiple unsuccessful attempts in industry to try and do so. A more basic research approach is required in this area in order to make progress.

Reliability

To the highest extent possible, the hardware should provide abstractions that hide any underlying unreliability of hardware systems, rather than exposing the unreliability to the programmer. Moreover, the hardware should provide support for robust debugging to increase the reliability of programs. For instance, the effects of a bug can often be far removed from its source. Fine-grained tracking of program state enables debugging tools to find the source of these kinds of bugs. However, this kind of tracking imposes a tremendous performance penalty. Consequently, direct hardware support for tracking information flow and logging control flow can tremendously speed up this kind of debugging and improve programmer productivity.

Emerging Programming Environments

In the past decade, the quantity of programming languages in general use has expanded far beyond the C/C++ and Fortran of 15 years ago to encompass a large number of managed (e.g., Java, C#, etc.) and scripting (e.g., Python, Perl, Ruby, Javascript, etc.) languages. Moreover, systems are often constructed from several different languages, increasing code reuse, but causing problems with divergent language semantics and challenging debugging of mixed code. Because of a desire for portability and the difficulty of compiling dynamic language constructs,, scripting languages are typically interpreted, leading to poor performance and inefficient use of hardware resources.

Computer architecture changes could improve the performance of interpreters, particularly the large dispatch loop that is the heart of most interpreters. The switch statement is usually implemented with an

indirect branch, whose behavior is unpredictable. Moreover, the ubiquitous operations of type checking, bounds checking, and garbage collection might benefit from hardware support.

Direct hardware support could improve both the performance and energy characteristics of interpreted languages. For instance, the ISA could be augmented with instructions that support the dispatch loop. As another example, managed languages with garbage collection require write barriers. Hardware support for write barriers could eliminate a significant performance penalty of managed languages.

3 The End of the Road for Conventional ISAs

The interface separating hardware and software (the ISA) and the corresponding levels of abstraction are critical to well-engineered computer systems. For example, they allow hardware and software to be developed at different times, in different locations, by different design groups, working for different companies.

However, the ISA as used today was developed decades ago. The important characteristics of ISAs were well-established in the 1960s (IBM 360). The ISA that is most widely used today (x86) was initially developed over 30 years ago, and has been repeatedly extended since then with backward compatibility being maintained with each extension.

Modern processor designs are a marvel of capability: they re-order instructions to achieve higher degrees of instruction-level parallelism, use speculation to ignore potential dependences that fail to materialize, and learn program behavior to apply speculation judiciously. For the past 30 years, processor implementations for commercially important ISAs have been refined through academic research and industrial development to the point where a lot of the potential for these ISAs has been realized, pushing current ISAs beyond their feasible design envelope

While ISA compatibility has served an important function, there are significant reasons to believe that these CISC and RISC ISAs developed in the 1970s, 80s and 90s are not appropriate for modern computer design. They were defined at a time where the resource challenges were different (area was at a premium, power wasn't) and before the consideration of security, reliability, and efficient execution of managed-code environments were first class goals. A significant amount of information that is known at compile time is lost and has to be re-discovered by the hardware, and there are limited means for communicating information back from the hardware to the software. Hardware needs to allocate resources and track state at the granularity of individual instructions, when a larger granularity would be more efficient. Finally, existing ISAs preclude some of the most energy efficient implementations like accelerators and reconfigurable logic. All of these represent opportunity costs that will continue to grow.

Today, we find ourselves with ISAs that are ill-suited for both the software and hardware. Current ISAs do not reflect the needs of current applications for dynamic data sharing, security, and reliability. Furthermore, software expressed in current ISAs requires energy intensive hardware for mapping ISA specifications to hardware resources (e.g., through register renaming). Current ISAs fail to provide an efficient means for providing performance-critical information to hardware. This means that hardware must uncover this information on its own, often doing the same work repeatedly as the same instruction sequence is executed. Again, this tends to be energy-intensive to perform in hardware. Current ISAs also require the support of a precise process state model that is defined at the assembly language level. This places a severe constraint on hardware while not providing useful information to the high level language programmer.

Taken together, these observations clearly indicate that we have reached the end of the road for conventional ISAs. Consequently, we recommend the complete overhaul of the current ISAs, in a way that makes the difference between RISC and CISC look like syntactic sugar. What follows is just a small sampling of the research directions that we envision should be pursued with the goal of developing a new ISA model to carry the IT industry forward into the next half century.

Research Thrust Description

New ISAs

The space for future interfaces and implementations is immense. Exploring this space involves identifying appropriate layers of abstraction along with efficient implementations that meet the design goals. In addition, interaction with the memory model and exception semantics must be carefully thought out.

One possible solution is to identify two new layers of abstraction in the system stack, a logical ISA and an implementation ISA. The logical ISA is the ISA that plays a role similar to today's conventional ISAs and is the one to which software is statically compiled. The implementation ISA is concealed from all conventional software and is the target of dynamic compilation and optimization via a concealed runtime; the hardware implements the implementation ISA. The runtime is a form of virtual machine that is designed concurrently with the hardware --- a so-called co-designed VM --- and it is concealed from all conventional software.

What the logical and implementation ISA should be are open questions, and may vary across application domains and/or design targets and constraints. The logical ISA could be an existing ISA, potentially with additional annotations; an existing high-level language bytecode representation (e.g., Java bytecode, Microsoft's Common Intermediate Language) or a compiler's enhanced intermediate representation (such as that used by LLVM); or a completely new abstraction with specific support for communicating software characteristics and programmer intents to the underlying layers, or the logical ISA could be specifically tailored towards a specific application domain (e.g., distributed computing in the cloud on large data sets). The implementation ISA is completely unconstrained by compatibility requirements and can potentially be different for every hardware implementation. The implementation ISA can be chosen such that it optimizes a particular set of design constraints (e.g., energy efficiency) and/or it may be specific to an application domain (e.g., mobile, cloud, etc.).

One particularly compelling means of re-defining the hardware/software interface is co-designed virtual machines, where the notion of the instruction set architecture is divided into two interfaces: the logical ISA, which is an implementation independent interface by which programs are distributed, and the implementation ISA, which is the actual instruction set implemented in hardware. By this separation we achieve three main goals: 1) we permit a rich interface (the logical ISA) that enables the programmer/compiler to communicate intent as well as expose high-level characteristics such as concurrency (independence) as well as data flow (dependence) among instructions, 2) chips can include a wide range of heterogeneous cores and accelerators, and 3) we allow the implementation ISA to be tailored specifically to the microarchitecture with no forward compatibility concerns. By placing a (hidden) software layer to translate between the logical and the implementation ISAs, we can use the rich program information coupled with the dynamic feedback from the hardware to select the appropriate cores for a region of code and optimize it appropriately.

Infrastructure Development

Conducting this kind of radical ISA research is not easy. Research in identifying new interfaces and implementations requires substantial infrastructure investment. It requires a re-targetable static compiler that compiles application code to the logical ISA, and it requires a re-targetable dynamic compilation framework that dynamically translates and optimizes logical ISA code into an implementation ISA that is run on the hardware (or simulator). A simulator allows detailed evaluation of the hardware design and hardware/software interaction. As suggested above, a key feature for a successful research infrastructure is that it is easily re-targetable so that different logical ISAs, implementation ISAs, and hardware implementations can be explored quickly.

Along with a re-targetable simulation infrastructure, new simulation methodologies are needed to be able to understand performance phenomena over long periods of time (say, minutes of run time), the reason being that the runtime (the co-designed VM) optimizes code dynamically. The runtime will perform online performance/efficiency monitoring in order to drive optimizations of the software and hardware during run time. Ideally, rather than reinvent common needs across multiple research thrusts, investment in a common infrastructure (similar to the LLVM compiler, for example) is likely to foster research in this area.

Supporting Legacy Software

Devising a runtime that translates and potentially optimizes legacy ISA code into the implementation ISA will enable support for legacy code. This legacy code runtime could be viewed as transient technology. The legacy code runtime can be abandoned once the logical ISA is widely adopted, enabling the legacy code to be recompiled or binary translated to the logical ISA and combined with all the new software that gets compiled to the logical ISA.

Cross-Cutting Issues: Dynamic Hardware and Software Optimization

The plethora of transistors afforded by Moore's law provides future systems with unparalleled hardware resources, but power and thermal limits dictate that only a subset of these resources may be simultaneously active. Future systems are likely to consist of a sea of structured or unstructured resources that must be efficiently composed based on the global resource allocation policies, local application requirements, and other constraints. To maximize quality of service metrics (e.g., system level efficiency) or meet service level agreements (e.g., application responsiveness) subject to power and resource constraints, future systems must dynamically adapt to changing application and environmental requirements. For example, as an application or run-time environment changes, the underlying hardware can change, in order to reduce power, improve performance or provide more functionality. Specialized components, such as special-purpose functional units, interconnect, and memory structures, may be selectively enabled and composed for general-purpose tasks. Whether in a single application or multi-program environment, dynamic microarchitecture adaptation enables application- and system-specific optimization based on the current constraints. Adaptability provides a framework that can be leveraged by the hardware/software interface to simultaneously coordinate global and local resource management. This framework requires mechanisms to communicate up and down the hardware/software stack. By facilitating information flow down to the hardware, the microarchitecture can adapt to the specific application or system requirements and constraints, like the power budget. With information flow up the stack, from the hardware to the software, we enable global resource control and feedback based on the current and future system constraints.

Application Characterization

Research is needed in application characterization to understand their properties and commonalities, so that these get reflected in the new interface(s). For example, security is a key attribute --- the ubiquity of networked devices calls for robust security policies. As another example, data-intensive datacenter-level workloads may benefit from constructs that enable specifying parallelism and the order in which the computation is done is not important. Yet another example may be the ability to communicate (in) dependences between reads from and writes to memory to the underlying layers.

Security and Reliability Properties

Existing protection models are fairly heavy-weight and do not address current application needs --- new protection models are needed. For example, fine-grain protection is useful in enforcing protection boundaries among modules within a single application. In addition, security must be architected in order to make it a first-class property that applications can manipulate. Similarly, support for identifying and exposing both hardware and software failures will enable reconfiguration and improve reliability. Understanding the ISA implications on these properties is an important research direction.

Why is this Research Transformative?

By moving away from the traditional ISA as the sole hardware/software interface, we enable transformative research. Existing ISAs severely constrain research into new microarchitecture paradigms and limit the role that processors can play in providing system-level support for security, reliability, debugging, and performance analysis. By re-drawing and re-defining the hardware/software boundary, we permit a more holistic approach to addressing every low-level aspect of computing, and enable an inter-disciplinary effort to re-consider what hardware primitives should be provided in each of these areas.

The decoupling of a portable format for programs (the logical ISA) from the instructions implemented on the machine (the implementation ISA) greatly relaxes constraints on the implementation ISA (and microarchitecture). Most notably this enables unlimited heterogeneity both across and within platforms transparently translating applications to a given form factor and the available set of processors/ accelerators to maximize energy efficiency. Furthermore, deferring many decisions to run-time (when dynamic information is available) rather than being restricted only to static compile time will revolutionize both compiler optimization and resource allocation.

Finally, new implementation ISAs and better information provided from the logical ISA or the intermediate software layer enable complete re-thinking of the microarchitecture. Rather than making the microarchitecture fit the architecture, we can allow the microarchitecture to drive the implementation architecture, with the implementation architecture being coupled to the logical architecture via the concealed runtime. A redefinition of the granularity of the work unit coupled with the provision of an execution plan that reduces decision-making and mis-speculation by the hardware could lead to significant improvements in energy efficiency.

What Are the Deliverables?

If successful, the deliverables from funding this line of research would include:

- A new HW/SW interface, broadly in use in both research and industry, that scales, supports modern computing requirements, and provides a platform for conducting future architectural, programming language, and systems research.

- An infrastructure supporting flexible HW/SW interface design, analysis, and understanding. Such an infrastructure would provide for both legacy and emergent application support.

- A deeper understanding about the nature of computation. The current acceptance of how processors must be built is stuck in a 1960's mindset: rigid ISAs, precise state, monolithic and uniform memory, etc. This mindset has pigeon-holed the type of architectural research envisioned and carried out, and held back industrial computing enhancements.

Research Disciplines Involved

Developing new interfaces requires a multi-disciplinary approach. Architects must closely collaborate with application domain experts, language designers, systems researchers, and technology experts, in order to develop interfaces that (a) meaningfully capture program intent that can be utilized for architectural optimization, as well as (b) provide the necessary feedback to the system and application that allows dynamic property tailoring.

Risks to Industry and Society If Not Pursued

The risks to industry if this research is not carried out, or is unsuccessful, is a dramatic shift in the IT business. Already, today, we see that the time taken by individuals and businesses before they upgrade their systems is double the time in the 1990's . If this were to double again, and at the moment, the momentum is in that direction, then the "life time" of a computer would have gone from approximately 2

years to 8 in less than two decades. This would have enormous implications on IT spending and an associated impact on the wider economy.

Benefits of Success to Industry and Society

The research direction described here, if successful, promises to return the industry back on a virtuous cycle of continuously better -- faster, lower power, more feature rich -- devices. Not only will that reignite the engine at the core of the IT industry -- translating Moore's law exponential decrease in transistor cost into exponential economic growth -- it also promises to unlock new levels of performance, enabling ever more insightful scientific investigation.

Why Not Let Industry do it?

The established computer industry has a strong vested interest in maintaining the current ISAs. It yields a proprietary advantage and it preserves a large part of a company's design infrastructure, e.g., CAD tools, validation and diagnostic software. Furthermore, designs have evolved to become extremely complicated so that even minor changes tend to be avoided (if it works, don't fix it). What is required is a major re-design of the abstraction layers separating hardware and software. Academia, without vested interest in the status quo is very well-suited to undertake this effort.

In order for academics to be successful at pursuing this research agenda several things must happen. First, funding has to be available to carry it out. This report (Section 6) includes specific funding recommendations to federal funding agencies that will make conducting this kind of research possible. Second, the academic ecosystem (infrastructure, students, conferences, etc) needs to be conducive to encouraging this type of research. It goes without saying that it is easier to carry out and publish incremental architecture research. This will change through leadership and the building of a significant body of like-minded researchers.

Likelihood of Success

There is enormous risk in pursuing research in this area. There is enormous risk in not doing so. The prospects for success are unknown, but several historical events show promise for the future. First, the concept of separating the logical from physical ISAs has been successful already in industry. The two most widely known examples are the switch to the Pentium 6/Pro microarchitecture and the processor devices from Transmeta. Both occurred in the 1990's and in both cases the underlying microarchitecture executed a different, but related ISA, from the exposed logical ISA. All x86 chips built today separate the logical from the implementation ISA.

This report argues research and industry must go much further than these attempts. Concepts such as precise state and monolithic, uniform memory, must be questioned. Researchers have explored around these concepts decades ago (dataflow, software controlled caching), but such attempts were conducted in an entirely different technological environment, and much has been learned in the intervening years. And of course, much can be learned from studying these earlier attempts.

4 Beyond Power/Performance

The foundation of our IT infrastructure today is an architecture developed by mid-20th-century pioneers and to a large extent stabilized with the IBM 360. This architecture focused on the efficient use of the modest resources available during that time and facilitated many valuable products, such as mainframes, minicomputers, and personal computers. Today, roughly a half-century later, this 20th-century architecture hobbles efforts to attack our nation's most important IT problems, including security and privacy, reliability, and programmability. For example, software which is certified at EAL6, the second highest level of assurance in the Common Criteria, costs on the order of $10,000 per line of code, compared to $1 to $3 for un-assured code. A large portion of this cost difference is due to the fact that modern machines completely lack the mechanisms necessary to manage information flows effectively, to recover from hardware and software failures, and to provide understandable and predictable performance. There are several major problems. First, current architectures make sharing the default, which makes even keeping a secret effectively nearly impossible. Second, they allow a single bit flip and errant pointer to potentially corrupt vast state, which increases the challenges of providing reliable systems by not containing inevitable errors. Finally, they promote a tower of fragile software that raises the cost of getting systems working.

In many dimensions, the computers of the 21st century are more than 1000 times better than their mid-20th-century ancestors. Moreover, society has come to depend on IT for education, government, commerce, science, and entertainment. Due to these changes, a new foundation is needed: ***We cannot and should not build IT skyscrapers on a bungalow foundation.*** This 21st-century architecture must facilitate what matters now and in the foreseeable future: Security and Privacy, Reliability, Programmability, and Predictability.

Research Thrust Description

To attain our goals, we propose to (1) develop a new class of architectural security and privacy primitives, (2) establish the new architectural mechanisms necessary to build truly reliable computing devices, (3) uncover the methods necessary to create understandable and predictable systems.

New Architecture Foundations for Security and Privacy

Within the thrust of architectural foundations for security and privacy we identify three sub goals: (1) the creation of new architectural mechanisms capable of strongly containing secret data and operations; (2) the rethinking of architectures within the context of modern datacenter use, where a single core may be utilized by hundreds or thousands of individuals; (3) architectural techniques in support of ensuring safe operation in which the participating parties (user, OS, applications) are mutually distrustful.

Secret Containment

Critical systems require tight guarantees on the flow of information, for example when handling social security numbers, secret cryptographic keys, health records, or critical avionics data. The ability to strongly protect a secret is at the heart of most secure systems, a task for which modern machines are exceptionally poorly suited. Instead, software developers are forced to constantly compensate for the failings of these machines. The problem is that every operation performed in a modern machine leaks some form of information. Secrets can be inferred from changes in the cache behavior, the timing, the electro-magnetic emissions, and many other logical and physical sources. In many domains, software developers are willing to go to remarkable lengths to minimize the amount of leaked information, for exampleby flushing the cache before and after executing a piece of critical code, attempting to scrub the branch predictor state, normalizing the execution time of loops by hand, or by attempting to randomize or prioritize the placement of data into the cache. While these techniques make it more difficult for an adversary to gain useful knowledge of sensitive information, in the end these heuristics have tremendous costs in terms of both time and money, yet do not bring our systems any closer to a strong notion of security or privacy. We need new architectures that, with information containment and side channel mitigation, support the core cryptographic operations on which the next generation of secure systems can

be built. If successful, such techniques will drastically reduce the effort necessary to build the secure systems on which our nation depends on for both commerce and national security.

Multi-Tenancy from the Ground Up

In addition to the problems with secret containment, the architectures we have today were simply not built with the massive multi-tenancy we find in modern data centers. Data centers today provide exceedingly complex services, everything from massive queries spanning hundreds of thousands of machines, to raw computation as a service through cloud computing. The fundamental structures of the machines that they are built on were envisioned originally to be used by a single user at a time, but in this context it is not uncommon for a single processor to contribute to the data and compute needs of tens of thousands of different users within a single second. In addition to the many opportunities this presents to optimize for the multi-tenancy case, it also potentially opens up many new avenues for attack. A single compromised node can have a devastating effect on both the integrity and availability of those services. Supporting this scenario requires that the ability to strongly isolate individual users from one another without giving up the performance and manageability gains that aggregation provide.

Operating Under Mutual Distrust

The enforcement of information flow policies is one of the most important aspects of modern computer security, yet is also one of the hardest to get correct in implementation. Unfortunately, even understanding the true flow of information through a traditional processor is difficult because executing an instruction affects so much internal state: the program counter, the memory system, forwarding and pipeline logic, and countless other bits throughout the machine. This problem is exacerbated even further when the operating system, applications, and user are mutually distrustful of one another. The modern realities of software are such that such distrust is not uncommon. Operating systems can be compromised by attackers, applications are pieced together from parts created around the world by people with complex or unclear intentions, and users are subject to both manipulation and confusion which may lead to poor decisions. A next generation architecture needs to support powerful primitives for isolation, the ability to verifiably virtualize resources, strong and non-bypassable methods of monitoring and enforcement, and rich methods of information flow control. One potential direction in this vein is the work on dynamic dataflow tracking architectures that has led to many clever new ways of detecting everything from general code injection attacks to cross-site scripting attacks.

Strong Architectural Support for Reliable Systems

While many strive for perfection in the systems they develop, the reality is that systems fail. They can fail for many different reasons, including device failure, hardware faults, software bugs, malicious behaviors, or any of a host of other problems. These imperfections can never be fully eliminated, and failure is always an option. However, systems today are built on top of a hardware / software stack that is overly optimistic leading to exceedingly brittle systems. Simple failures, such as the flip of a bit or an errant pointer can set in motion a devastating chain of events, leading to anything from loss of data, to loss of life, depending on the situation. We need to develop new architectural methods to radically transform the problem of reliable system development, allowing the creation of robust and gracefully degrading systems. Towards this goal, we identify three sub goals: (1) new lightweight and trustworthy methods of identifying different classes of faults at the architecture level; (2) strong, even provable, architectural methods by which the effect of those faults can be isolated and contained; and (3) new architectural mechanisms to support efforts to mitigate the effect of those faults.

Error Detection

The first, and perhaps most fundamental problem in reliability is simply recognizing that a fault has occurred. As we attempt to push the underlying circuit technology to its breaking point, and as we add new technologies with complex fault models, the ability to detect errors is of critical importance. If failures can be reliably identified, we can use aggressive feedback driven optimization, pushing technology to the bleeding edge. Of course faults can also occur above the device level in the architecture itself. As we

attempt to squeeze performance out of our systems at the end of scaling, we will need the ability to rapidly design and deploy aggressive new machines. The ability to detect architectural faults, perform core-level self-check, and dynamically verify results will provide a safety net for the tight wire act now required. Finally, we need new architecture mechanism to help efficiently and continuously identify errors at the full system level, as problems such as atomicity violations and concurrency errors become increasingly important.

Error Containment

The second problem is that once a fault has been detected (especially if the fault is detected late enough that it is not transparently reversible), the effect of that fault must be properly contained within a portion of the system. This problem is made worse by the fact that the outcome of a hardware failure is not immediately clear. For example, if a single bit of a movie frame is modified, it may have no consequences to the operation of the system as a whole, while if another bit is modified, it might have dire consequences. Currently there are no methods by which we can determine the true impact of either hardware or software failures. Such methods are a necessary first step towards strong fault containment as they can be used to measure success. In terms of containment, the final piece that we need is lightweight, configurable, and strong information firewalls. The ability to protect critical components from even the most devious failures would be a powerful new primitive in the creation of reliable systems.

Error Mitigation

The third problem is the ability to mitigate the effects of any, preferably properly contained, errors. While many methods for gracefully handling errors exist, these methods are often times not accessible or performed in the domains in which they are now sorely needed. For example, while transactions are a well-established way of bounding the set of possible states that a system may fail into, most computation today is not structured in a transactional way. Architectural mechanisms that will enable such mitigations include lightweight replay, transparent failover and adaptation, and fault-aware reconfiguration.

Predictable and Programmable By Design

While reliability and security are critical end properties of many systems, it is important to not forget that most systems are built incrementally, starting with small simple pieces and then extending and composing those pieces. Programmers and engineers need ways of creating new systems easily, identifying and eliminating potential problems in both correctness and performance, and in extending and maintaining those systems far into the future. In support of these goals, we identify three sub goals: functional models of architectures understandable to software, performance models of architectures understandable to software, and lasting/extensible architecture interfaces.

Strong Models of Function

One of the most pressing problems in computer architecture is the lack of good models of processor function that can be used by higher level programming systems. Currently, if one wishes to perform a formal analysis of a piece of software at the assembly level, the first thing that needs to happen is that the semantics of the processor need to be defined formally. Typically this happens by reading the processor architecture documentation and making approximations as appropriate. However, there is often a large gap between the documented architecture and the actual implementation that contains updates, bugs, undocumented features, and secret control instructions. We propose that architectural innovation is needed to address this growing problem, with particular respect to determinism and semantics. Solutions in this space should include the definition of formal semantics at the architecture level, work across the formal programming language and ISA boundaries, and ensuring the safe and deterministic execution of microprocessors under a variety of external factors.

Strong Models of Performance

In addition to correctness specifications of these systems, a significant amount of programming effort and perceived reliability is performance. Often when a system fails to perform as expected the reasons for

this behavior lie in some complicated cross-level interaction, such as the network stack interacting with paging and interrupt handling. These cross level issues are difficult to diagnose, especially so when they appear exclusively in live production machines but not in the test suite. These performance failures can lead to everything from loss of income to, in hard real time systems, loss of life. In this direction, we need new methods for debugging performance problems, performance isolation, and hardware real-time where appropriate.

Lasting Interfaces

Across all of the topics mentioned above, including relating to security and reliability, we need to build systems with lasting interfaces. Architectures that solve one-off problems are capable of moving the needle, but what is really needed are new abstractions that go beyond this and will carry far into the future. Commercially, x86 in this regard has been a wild success. Originally specified for the 8086 in 1978, it has proven extensible for over 30 years. We need to determine the interfaces necessary to ensure another 30 years of progress while enabling the development of predictable, secure, and reliable systems.

Cross Cutting Issue: Adaptability

Future systems must provide mechanisms and policies to balance user requirements for emergent capabilities against the constraints imposed by limited hardware resources. For example, critical financial applications require higher levels of security and reliability than applications that simply enhance the user's experience. Resource allocation to support these capabilities may depend upon the perceived error rates and threat levels, which may vary due to environmental and other factors. In addition, policies must account for the competing demands of other applications and reach globally, not simply locally, optimal allocations. To enable adaptation for security and reliability requires, at least, novel adaptable mechanisms for detection, containment, and mitigation.

To adapt to higher reliability requirements, a system may enable N-modular redundancy and stronger error correction codes. For security, it may enable strong information and control flow tracking mechanisms. Limiting information flow and error propagation require strong mechanisms to contain information within limited domains. These mechanisms must be reconfigurable to adapt to changing trust relationships and application failure models. Transparent recovery from detected anomalies may require the reassignment of remaining resources.

Why is this Research Transformative?

A 100-fold decrease in the cost of developing high assurance software will transform the problem space entirely, as it would provide new opportunities to offer value to society, especially in scenarios where current architectures are lacking. For example, applications in medicine benefit from systems for which security, privacy, and reliability are foundational. Moreover, as history has shown, better IT technology, allows creative people to develop whole new genres of activity that are hard to envision beforehand.

What Are the Deliverables

In 3 years, this research can produce papers that provide alternative visions for achieving aspects of the aforementioned goals. The work in these papers will sometimes compete, sometimes expose synergies, and sometimes motivate new approaches and refinements.

In 5 years the research should produce test-bed systems that illustrate the effectiveness of research ideas and how multiple aspects interact. For example, one could build a digital "shoebox" which would preserve and keep private pictures and movies for a decade.

In 7 years, this research and initial test bed should enable preliminary products, with an emphasis on problem domains where security, privacy and reliability are paramount (e.g., medical devices).

In 10 or more years, we expect to see new classes of products enabled by this new foundation, much as the personal computer facilitated spreadsheets, word processing, and web access that were not envisioned by its creator.

Research Disciplines Involved

Architecture and architects do not solve the aforementioned problems alone. Architecture should provide mechanism that, on one hand, support the policies, and, on the other hand, are efficiently implementable. To develop such mechanisms and policies, architects must first work with domain experts. Security work must be done in conjunction with security experts; reliability work should be done with researchers in databases and fault tolerance. Programmability work must involve language, application, and sometimes real-time experts. All of the above will require architects to interact with experts on system software and runtime systems. Finally, hardware implementation issues will require close interaction with device and circuit scholars.

Risk to Industry or Society If Not Pursued

IT has been a valuable engine of growth for the USA, allowing us to compete by innovating rather than compete by lowering cost and wages. Given that scaling of performance-power will not continue at its historic pace, the IT industry and the innovation it fosters in society are at risk of stalling. In contrast, the aforementioned research enables new ways of providing value through security, reliability, and more efficient software development.

Benefits of Success to Industry and Society

In the last decades, performance, cost, and power scaling of computers has been phenomenal. This allowed hardware vendors to develop and sell machines, software vendors to create and sell new experiences, and society has found value in both, as exemplified by how people buy these products.

A new architecture will enable new ways for industry to create value. If successful, society will buy and benefit from this value. For example, it could create computers as reliable and simple to use as modern automobiles. It could create computers safe enough to permit placing bank accounts and children's pictures in the cloud without undo concern for theft or corruption.

Why Not Let Industry Do It

Industry has focused on performance to the exclusion of many other important properties. While it may focus on some of these new properties going forward, industry is best suited for local optimizations that improve a system while changing only a small part of it. This is true in part because one company rarely controls the whole ecosystem for applications, system software, and hardware. In contrast, the new architecture we advocate is a global rebuilding of the computing foundation that will take many years to get right.

Moreover, successful companies often have a disincentive to radically innovate (or publish such innovation) when they are currently dominant and new technology enables competing players. For example, 30 years ago the broad community did not learn about IBM RISC technology until Berkeley and Stanford published their RISC ideas.

Likelihood of Success

We see three avenues to success. First, new architectures are most easily successful in a new market that (a) benefits from new architectural properties and (b) does not a have a large installed base demanding backward compatibility. Mobile devices and cloud computing follow this avenue, as do embedded devices such as medical and automotive / avionics.

Second, success in research and new markets will motivate work to retrofit ideas from new architectures into old ones. Perhaps someday a current laptop environment will run in a safe sandbox enabled by the new, more-secure architecture.

Finally, the new architecture can replace the old architecture in established markets. This avenue is made more possible when successes in the first avenue enable the new technology to mature. Nevertheless, success on this avenue is harder -- and perhaps less important -- then the other two, since the old architecture has been sufficient for success up until recently.

5 Leveraging New Technologies for Computer Architecture

New technologies provide an opportunity to extend the historical rates of computer performance improvement previously delivered by Moore's Law for the foreseeable future. This research thrust focuses on identifying some of these technologies and showing how they can be used to redesign computer architectures to get us back to the exponential performance improvements while reducing power consumption. Our ambitious goal is to be able to provide two to three orders of magnitude in performance improvement, energy efficiency increase, memory capacity increase, and reduction in manufacturing cost.

Research Thrust Description

There are several emerging technologies that can potentially provide major improvements in the frequency of compute logic, power consumption, density and capacity of storage, communication bandwidth (in particular, for non-local communication), communication latency, reliability, non-volatile storage, and cost of manufacturing. However, these technologies often change the bottlenecks in a computer system, and can possibly add new bottlenecks. In addition, many of them have reliability challenges. Finally, in many cases, understanding them requires multi-disciplinary research and partnership with researchers in physics, electronics, chemistry, and material science.

In this thrust, we identify some of these technologies. We group these technologies into short term (up to 10 years, approximately), and long term (for technologies that will take more than 10 years to be commercialized). For each of these technologies, we need to work with experts to understand and advance the technology. Then, we need to identify the positive and negative impacts of it on computer architecture. We need to exploit the opportunities and fix or minimize the shortcomings. We also need to identify the implications of the technology on the application domains. For example, some technologies may enable large working-set or data-intensive applications, while others may enable fine-grain synchronization. Finally, we need to identify the complementary technologies and the incompatible ones, and identify how to integrate them together in different parts of the system. In the following, we describe the research agenda based on short-term technologies, long-term ones, and the infrastructure needed to pursue this agenda.

Short-Term Technologies

We believe that several new technologies, such as 3D stacking, optics, supply voltage management, and non-volatile memories will significantly impact the computing stack in the short term. We briefly describe specific promising research directions in each of these areas.

3-D Stacking for Memory or Logic

3D stacking provides the ability to bond multiple dies together in the same package using silicon connectors instead of pins, thereby reducing the footprint and increasing the density of placing multiple dies together.

The advantages of this technology include incorporating heterogeneity into the design, reducing the latency and increasing the bandwidth between components that would otherwise be connected via pins, and reduced power consumption due to tight connection between the components. The current disadvantages of this technology are increased thermal pressure and cost of manufacturing.

There are several ways in which future computer architectures can be designed to take advantage of this technology.

3D die stacking can enable more compact memories by stacking memory dies on top of each other. Large improvements in system performance can be attained by placing processors and memories in the same stack, reducing the latency and increasing the bandwidth to main memory. Likewise, we also increase the capacity of the memory that is close to the processor. Moreover, 3D die stacking can enable ways to design a heterogeneous system that consists of components implemented in different technologies.

There are several specific research directions to pursue. One example is how to partition the entire processor and memory hierarchy across the multiple dies. Another one is how to design an effective heterogeneous architecture. Yet another one is how to manage heat dissipation to ensure that 3D stacking is successful.

Optics

Recent advances in optoelectronics have made it possible to place optical components/interconnections both on-chip and off-chip. The advantages of optical interconnection technology include high bandwidth, low power, and low latency communication which can facilitate both on-chip and off-chip communication. The disadvantages include high overhead (latency and power) of optoelectronic conversion and reliability and yield issues in manufacturing.

Optical communication can benefit on-chip interconnections, off-chip interconnections within a board or platform, and large-scale interconnections such as those in a data center and supercomputer. An important topic is to investigate the benefits in each of these domains in terms of performance and power. For example, how much can we reduce the memory bandwidth pressure by employing optics on chip and off-chip within a platform? Can we integrate photonics in a 3D setting to improve bandwidth and latency further?

Supply Voltage Management

A potential approach to design an energy-efficient architecture is to operate circuits at a low supply voltage, for example by employing near-threshold voltage design. The key advantage of this technology is a cubic decrease in dynamic power while the primary disadvantages are slower circuits and an increased error rate caused by process variation. One of the main challenges caused by low-voltage operation is detecting and correcting errors. Solving this challenge will usher in future computer architectures which are significantly more energy efficient. One promising research direction is architectural, system-level, and software techniques to correct or tolerate errors caused by low-voltage operation.

There are other areas of research around supply voltage management. One of them is to design circuits that have multiple supply voltages. Another research area is the design of efficient on-chip voltage regulators. This approach can enable very power-efficient and fine-grain management of the voltage. Research into how to provide this control in hardware or software or through hardware/software cooperation is promising.

Non-Volatile Memory Technologies

Several non-volatile memory technologies, such as Phase Change Memory (PCM), Magnetic Memory (MRAM), memristors, and Flash can potentially provide large improvements in memory system design in the face of ending DRAM scaling. These technologies have the potential to improve memory system capacity, density, and power efficiency. They also enable non-volatility, which can translate into performance and robustness improvements in the overall system. The current problems with these technologies are wearout due to erase cycles, large write latencies and large write power compared to SRAM and DRAM.

Research issues include how to design the architecture and system stack to overcome current disadvantages of these technologies, where to place different technologies (e.g., placement of MRAM on chip), how to exploit low-leakage characteristics to build a very energy efficient system, how to exploit non-volatility, and how to minimize wearout. These technologies can enable a rethinking of the entire memory hierarchy design.

A common problem with some of the above technologies, such as low-voltage design, is the increased error rate, variability and noise. Enabling these technologies requires research into architectural mechanisms to address these challenges.

Long-Term Technologies

We believe several emerging technologies, such as non-silicon substrates, new silicon transistors, and quantum computing can have a potentially large impact on the computing stack in the long term. We describe briefly specific promising research directions in each of these areas.

Advanced Silicon and Non-Silicon Process Technologies

While classical transistor scaling no longer brings exponential power and performance gains, new transistor structures and non-silicon substrates have the potential to turn back the clock and return the industry to another decade of exponential scaling. Research into these technologies is primarily driven by industry, but their realization would give computer architectures more time to adapt to the still-inevitable end of transistor scaling.

Non-planar transistor structures, particularly FinFETs and multi-gate FETs, allow for thicker gate oxide layers, the limiting factor in classical transistor scaling. Furthermore, tunneling transistors use quantum-mechanical electron tunneling to conduct electrons at very low power. The advantage of these technologies is better scalability and very low power consumption. The disadvantage is the manufacturing difficulties.

Different substrates also have the potential to delay the end of scaling. Extremely thin silicon-on-insulator (ETSOI) is the least disruptive as it still uses silicon, but it also has the least potential to return processors to exponential scaling. Non-silicon substrates, such as graphene, carbon nanotubes, germanium, and III-V semiconductors would enable at least another decade of scaling. The main advantages of these substrates are high density and, like non-planar transistor structures, extremely low power consumption. Their disadvantage is that their viability for manufacturing is fairly low in the short term.

A new plethora of computer architectures can be explored by exploiting the very low power consumption property of these devices. For example, very high frequency or very wide-issue microarchitectures that were not possible before due to power constraints can now be investigated. The ISA design can be re-thought to exploit the underlying low-power and high-frequency substrate. These microarchitectures can also enable very fast sequential execution, thereby easing programmability.

Quantum Computing

In addition to employing novel and non-CMOS technologies for more traditional computing approaches, quantum computing (QC) approaches also bear exploration. While it is unlikely that QC approaches will be usefully applied to general purpose computing in the near future, their applicability to certain cryptographic and search applications makes them worthy of attention. QC approaches have been the focus of considerable research by physicists exploring underlying device characteristics. At the other extreme, mathematicians and algorithm experts have likewise devoted significant effort to "applications" for these unique platforms. Severely lacking, however, is computer architecture research in between these two extremes, to consider possible organizations and architectures for implementing QC. For example, comparing several possible quantum bit (qubit) implementations, one finds there is significant variance in terms of how complex the qubit control mechanism will need to be, and in terms of the

latencies for qubit transportation and computation. So while physicists can explore qubits from a devices perspective, architects must weigh in regarding which devices offer the most promise for organizing into well-controlled and programmable computing devices. Such work can form the basis of our most forward-looking efforts in the area of technology-driven architecture.

Bio-Inspired Devices

Biological computing is another emerging field where computing (and computer architecture) can play a major role. DNA and RNA-based logic gates have already been constructed, showing the potential for doing computation in an entirely new domain – within cells. The potential for biological computing is astounding, but "programming" these circuits requires deep knowledge of the underling circuits, chemistry, and mechanisms. Computer architects have a big role to play to define the right interfaces and abstractions to make biological computing accessible to a much broader audience.

Infrastructure

This research is unique in that it requires a certain infrastructure. To attain this infrastructure, we propose several avenues. The first one is to partner with faculty in physics, chemistry, electrical engineering, and materials science from our university departments. We can write joint interdisciplinary proposals to funding agencies (e.g., NSF Expeditions). We can then share infrastructure with them, such as photonics networks, or new device technology. We feel that graduate students can greatly benefit from this collaboration.

We can also ask the funding agencies to provide funds for a nation-wide shared infrastructure that everyone can use. As an example, we can have a shared facility to fabricate 3D chips, or on-chip optical circuitry, similar to the model provided by MOSIS. We can also collaborate with industry, government laboratories or other institutions such as the Semiconductor Research Consortium (SRC).

Above all, we would like to encourage computer architecture researchers to share infrastructure, in the form of hardware and software artifacts. It is especially important that computer architects build models of how a given technology impacts computer architecture, and distribute them widely.

Cross-Cutting Issues with Adaptable Systems (Mark: number these subsections)

Ultra Fine-Grained Adaptivity

In adaptive and reconfigurable systems, the appropriate granularity for reconfiguration is determined, in part, by the characteristics of the underlying manufacturing technology. The smaller the reconfigurable elements are, the greater the relative overhead, in both area and delay, of the reconfiguration itself is.

Post-CMOS technologies will have different characteristics, creating new opportunities to evaluate the appropriate granularity for reconfigurable and adaptive hardware designs. Moreover new technologies may have fundamental properties that are particularly amenable to adaptation. For example, a memristor is a tiny passive circuit component that intrinsically recalls dynamic data value histories. This capability could be used to implement very tight, tiny feedback loops within a circuit, enabling substantially finer-grained reconfiguration, likely at substantially lower overheads.

Online, Adaptive Speed and/or Feature Binning

As new technologies emerge and scaling passes the limits of CMOS, we anticipate continued difficulties with process variation during manufacturing. For example, while carbon-based structures such as graphene ribbons and nanotubes have many desirable electrical and scaling properties, it is particularly difficult to finely control their placement. Static reconfiguration can help homogenize a variable population of chips by speed- or feature-binning the components on each chip. After this process, the on-chip diversity would be sorted into a set of bins that is comparatively homogeneous from chip to chip.

Online adaptation can even go one step further, continually re-evaluating what are the fast components and what are the slow ones as a system operates. This is particularly advantageous as applications change: a module that is fast or provides more features for one application may be slow or even useless for another. Adaptability makes it possible to consistently match the available hardware components to the software's most immediate needs.

Why Is This Research Transformative?

With the end of CMOS scaling, computers may become a low-margin commodity, possibly leading to the stagnation of the entire computing industry. Enabling and exploiting new technologies for logic and memory can enable computing to continue the performance and power scaling that has enabled new applications over decades.

The research is transformative because some of these technologies can provide orders of magnitude improvements in frequency, power/energy efficiency, logic and memory density and capacity, and communication bandwidth and latency. In addition, previously unavailable non-volatile on-chip or main memory storage can largely change the way applications are designed and can enable new applications.

The proposed research directions can also enable computer platform designs that were previously too complex or hard to build. For example, new technology may enable a 1000x1000 on chip optical crossbar design and/or very large on-chip non-volatile memory.

What Are the Deliverables?

The main deliverables are threefold. First, the research will result in a model for each of the new technologies that is relevant to and usable by computer architects and system designers. Second, the research will provide measurements of the impact of these technologies on existing traditional architectures, providing performance and power consumption evaluations. Third, the research will develop new architecture concepts and implementations that enable and take advantage of these technologies.

Research Disciplines Involved

This thrust is very interdisciplinary. It requires computer architects to partner with physicists, chemists, electrical engineers, materials scientists, and applications researchers.

Risk to Industry and Society if Not Pursued

With the imminent end of CMOS technology scaling, it is essential to investigate new technologies to continue the performance-power scaling trends that have enabled innovations in the entire computing industry, which in turn has transformed society.

If performance-power scaling does not continue, the entire computing industry will likely stagnate and the pace of innovation in sciences, technology, and society is likely to stall. For example, new applications that drive innovation but require larger performance will likely not be enabled. New technologies can provide new paradigms to enable going back to the traditional performance-power scaling curve

Since it is not clear which of these new technologies will be commercially viable, we have to invest in a multitude of them.

Benefits of Success to Industry and Society

New technologies will provide new paradigms to enable going back to the traditional performance-power scaling curve. This can enable the computing industry to flourish and the pace of innovation in the entire industry and society to expand by enabling new applications. Furthermore, by enabling very high

performance single-thread execution with non-silicon substrates, programming can become easier, enabling the industry to innovate faster and be more productive.

Why Not Let Industry Do It?

While many of the new technologies we have identified are developed primarily in industry, companies have little incentive to explore applications of the new technology to problem domains in which they have no business interest. This potentially limits the scope and impact of the new technology on society. Industry may also miss opportunities to integrate disparate technologies because they might focus on one or two near-term commercially viable ones. In addition, academic exploration of “what if?” scenarios can help identify potential problems and drive important future research directions before industry has fully developed the technology. Next, openly publishing models of new technologies allows other researchers and corporations to evaluate the impact of new technologies beyond the company at which the technology is being developed, speeding innovation and impact. Finally, industry has little incentive to share the results and parameters of technology for open research and broad-based exploration.

Likelihood of Success

While the likelihood of success of any particular technology varies wildly, there is little doubt that some of these technologies will radically change the options available to computer architects. Until new transistor structures and semiconductor substrates are ready for production, architects must rely on optimizations and new technologies to scale performance and power, driving the sales that keep industrial research funded.

The likelihood of success in applying computer architecture concepts to emerging domains depends on the domain itself. Yet if the new domain provides robust underlying "hardware", architects must find the right abstractions to let developers harness the power of the new domain without being domain experts.

6 Recommendations for Federal Funding Agencies

This report was formed by bringing together a diverse set of researchers in the computer architecture domain, representing substantial combined experience. The charge was to diagnose the present, and articulate the future. The result, embodied in this report, is a clear statement about the perilous state of the bedrock foundation of the IT economy -- the computing devices themselves. This report articulates a bold new agenda aimed to put architecture research, and ultimately computer systems design, back on a growth track. This agenda will flounder, however, without sustained commitment from the funding agencies.

As outlined throughout this report, industry will not pursue this agenda, or similarly bold agendas, on its own. There are heavy investments in the status quo. Even though there is enough market data demonstrating the slow-down in new computing technology investments by businesses, industry does not appear poised to accept, yet, that existing technologies and methods will not bring the industry back to exponential performance improvements. Only funding agencies, and academic researchers, divorced from near-term market dynamics, can take the “long view”, and pursue this research agenda.

This report contains three broad research themes: (1) rethinking the hardware/software interface; (2) enabling new computing capabilities; and (3) matching new technological progress to practical devices and systems. We recommend the funding agencies to take these themes and make them priority thrusts in their funding efforts --- either by allocating new funds or reallocating existing funds. We recommend the following funding strategy:

- **Fund ambitious, larger proposals for 5 years.** Many NSF-funded projects are too small and incremental to make much impact on the large problems that this report outlined. Rather than funding designs that are not too dissimilar from those we build and sell today, we recommend that NSF give priority to truly bold and far-reaching projects. These projects tend to require longer research periods (5 years) and require larger sums ($1--3 million). When needed, NSF program managers should have the flexibility to over-rule the recommendations of conservative panels.

- **Fund more proposals**. Despite effort, review panels will be conservative. Not all funded proposals will pan out. The only real hedge against failure is to simply fund more long term ideas. It is not enough for NSF to place strategic bets on one or two radical architecture ideas. Obviously, the need to fund more proposals runs counter to the need to fund them at a larger amount. We recommend NSF only pursue truly transformative research and maintain a strict adherence to this mission.

- **Fund cross-cutting proposals with interdisciplinary teams that are willing to change the architecture.** The problems described require teams that include computer architects and experts of other domains such as security, systems, material science, and electrical engineering. Proposals from such teams that assume that the architecture is changeable (rather than a “solved problem” that should remain fixed) have the highest chance to make an impact on the problems outlined. NSF panels evaluating these proposals should be carefully set-up so that the architecture game-changing aspect of the proposal is given high priority.

- **Coordinate NSF and DARPA/DOE efforts.** The NSF mission is to fund longer-term, basic research, while DARPA and DOE efforts have more emphasis on technology-transfer to industry and operate under closer, milestone-driven oversight. To maximize the impact of the funding, NSF must continue to focus on longer turn, truly transformative, research. DARPA/DOE must be engaged throughout the life cycle of these efforts, however, to step in and bring the most promising ones from scientific workbench, to reality. This will require stronger communication and tighter coordination between these agencies than has occurred in the past.

7 Conclusion

Contrary to the perception that some might have about computer architecture, it is not a “solved problem”. The emphasis of processor manufacturers on evolutionary technologies and methods is unlikely to broadly put computer systems back into exponential performance growth. The period between system upgrades by consumers, for hardware, and consequently most software, has doubled from historical rates, and is on track to double again. The impact from not funding innovative computer architecture research is keenly being felt.

The ACAR II workshop brought together a diverse group of researchers to help form an agenda for the future of architecture research. The result, embodied here, is a bold and invigorating agenda for computer architecture research that is far reaching, deep in its intellectual curiosity, and truly transformative.

The workshop attendees submit this report to funding agencies and to their own research community, and exhort them to study it carefully. We urge funding agencies to fund academic research in the research thrusts in this report. Only through deep and sustained funding of truly transformative academic research in computer architecture will we see many of the important challenges outlined in this report, overcome. Furthermore, we urge that small, but specific changes to the funding and review process for existing programs be made.

We urge the architecture research community to question the fundamental precepts of our field. The existing hardware/software interface has reached the end of the road for creative and impact-full computer architecture research. In addition, computer architectures must support new capabilities, including significantly more security and reliability, from the ground up rather than as an add-on. Finally, continued research into architectures for emerging technologies is crucially needed to make them possible successors to current technologies whose improvement is stalling.