Report for the NSF Workshop on Cross-layer Power Optimization and Management

Massoud Pedram, David Brooks, and Timothy Pinkston, Editors

July 31, 2012

Contributors:

Mohamed Allam, David Andersen, Murali Annavaram, Rajeev Balasubramonian, Kaustav Banerjee, Sankar Basu, Luca Benini, Christopher Batten, Keren Bergman, David Blaauw, Paul Bogdan, Pradip Bose, David Brooks, Naehyuck Chang, Pai Chou, Jason Cong, Brian Davidson, Jeffrey Draper, Eby G. Friedman, Rajesh Gupta, Sandeep Gupta, Sudhanva Gurumurthi, Lei He, Payam Heydari, Mark Hill, Jon Hiller, Charles J. Holland, Engin Ipek, Bill Joyner, Brucek Khailany, Hyesoon Kim, Eren Kursun, Benjamin C. Lee, Peng Li, Per Ljung, Ahmed Louri, Radu Marculescu, Margaret Martonosi, Renu Mehra, Rami Melhem, Jose Moreira, Trevor Mudge, Onur Mutlu, Farid Najm, Vijaykrishnan Narayanan, Kunle Olukotun, Michael Orshansky, Massoud Pedram, Timothy Pinkston, Viktor Prasanna, Qinru Qiu, Karthick Rajamani, Parthasarathy Ranganathan, Vijay Janapa Reddi, Kaushik Roy, Karu Sankaralingam, Sachin Sapatnekar, John Shen, Mani Srivastava, Mircea Stan, Steve Swanson, Michael B. Taylor, Josep Torrellas, Tom Wenisch, Lin Zhong

Disclaimer

The material in this document reflects the collective views, ideas, opinions and findings of the study participants and contributors only, and not those of any of the universities, corporations, or other institutions with which they are affiliated. Furthermore, the material in this document does not reflect the official views, ideas, opinions and/or findings of the National Science Foundation, or of the United States government.

Organization and Contributions

Organizers

Massoud Pedram, USC

David Brooks, Harvard

Timothy Pinkston, USC

Core Working Group (Area) Leaders

Kaushik Roy, Purdue

Jason Cong, UCLA

Margaret Martonosi, Princeton

Luca Benini, University of Bologna

NSF Sponsors

Sankar Basu, NSF CISE/CCF

Ahmed Louri, NSF CISE/CCF

Government and Industry Liaisons

Frederica Darema, AFOSR

Charles J. Holland, DARPA MTO

Bill Joyner (SRC)

Workshop Participants

For full list, see the Appendix.

Foreword

This document reflects the views of a group of researchers from universities, industry, and research laboratories on the need for potential new avenues of research that can change how power optimization and management is addressed in critical computing infrastructure. The report was generated from an NSF-sponsored Cross-layer Power Optimization and Management (CPOM) workshop on February 10-11, 2012. The general findings of the report agree with many from other strategic visioning studies and documents recently made available to the public, including the NSF Cyberinfrastructure for 21st Century Science and Engineering (CIF21) Advanced Computing Infrastructure Vision and Strategic Plan, the CRA CCC 21st Century Computer Architecture White Paper, and the NITRD Program 2012 Strategic Plan. Energy-efficient computing solutions that cut across multiple layers of the computing stack and infrastructure are needed for sustaining advancements in information technology to address critical societal challenges.

The goal of this study was to identify approaches and means by which cross-layer approaches to power optimization and management can be investigated, developed and adopted by the research community at large. The report was put together by area leaders based on input by the workshop participants. There was general agreement about the key challenges that surfaced from the study and the significant potential value of cross-layer optimizations to enable design and operation of power-efficient computing platforms. Several areas for targeted funding of important research directions have been identified.

We are grateful to have worked with so many dedicated people who put in many long hours to help the study along. Kaushik Roy, Jason Cong, Margaret Martonosi, and Luca Benini, led the focus area discussions. We thank our NSF sponsors, Sankar Basu and Ahmed Lori, for making this workshop possible, their active participation in the workshop, and their follow-up actions in addressing recommendations from the workshop.

We are privileged to have been part of this study, and wish to thank the CPOM workshop attendees for their time, their effort, and their insight.

Massoud Pedram, David Brooks, and Timothy Pinkston

Contents

Executive Summary

There is a current and growing crisis in power management for the entire range of power-performance computer infrastructure design points—from mobile to enterprise to cloud computing. The *NSF Cyberinfrastructure for 21st Century Science and Engineering (CIF21) Advanced Computing Infrastructure Vision and Strategic Plan*[1] documents the need for energy- and power-efficient computing. Other recent strategic visioning studies, including a white paper by the Computing Community Consortium on 21st Century Computer Architecture[2] and the NITRD Program 2012 Strategic Plan[3], emphasize the importance of energy-efficient computing for sustaining advancements in information technology and addressing critical societal challenges. The exploration of holistic power optimization and management solutions that cut across multiple layers of the computing stack and infrastructure will better enable available opportunities for maximizing energy efficiency to be fully exploited.

It is natural to ask: Why is cross-layer power management promising? Without sacrificing performance and reliability, what is missing between the various layers that inhibit effective power optimization and management? What are key research challenges? An NSF-sponsored workshop on Cross-layer Power Optimization and Management (CPOM) was held at the University of Southern California on February 10-11, 2012, to address these and other important questions. It was organized into four parallel tracks:

- Area 1 - Technology, Circuits and Beyond
- Area 2 - Circuits, Micro-architecture and Beyond
- Area 3 - Micro-architecture, Systems and Beyond
- Area 4 - Systems, Applications and Beyond

This timely workshop provided a forum for leading experts, including representatives from government funding agencies, to discuss methods and means by which cross-layer approaches to power optimization and management can be investigated, developed and adopted by the research community at large.

Several high-level findings emerged.

1. Energy efficiency has become a first-order design goal. Designing for both high performance and low energy is something companies would claim to have been doing for years already. But the design process is still not fully adjusted for this. For example, how does the design process change if one starts from the following viewpoint, "Everything is turned off; what should be turned on for each computation?"

[1] NSF Cyberinfrastructure for the 21st Century Science and Engineering (CIF21) Advanced Computing Infrastructure Vision and Strategic Plan, www.nsf.gov/pubs/2012/nsf12051/nsf12051.pdf, February 2012.

[2] CRA CCC 21st Century Computer Architecture White Paper, http://cra.org/ccc/whitepapers.php, May 2012.

[3] NITRD 2012 Strategic Plan, http://www.nitrd.gov/pubs/strategic_plans/2012_NITRD_Strategic_Plan.pdf, July 2012.

2. New extreme-scaled CMOS devices are being introduced—these devices are subject to significant variability and aging effects and are expected to operate reliably and with high switching speeds at very low supply voltages. Current design tool methodologies and tools are not capable of achieving the full potential of these devices in VLSI circuits and systems.
3. Power efficiency is about not only low leakage currents and small switched capacitances but also the efficiency of the power distribution network, power conversion circuitry, and heat removal. A holistic approach to power optimization and management that considers all this is in great demand.
4. A general shift away from CPU-centric design thinking is taking place. In mobile systems, the focus should be on display, radio, and sensors. In enterprise systems, memory, storage, and networks are increasingly more important in terms of their power usage. To go beyond the incremental energy efficiency gains possible from component-wise optimization, one must consider the coordination and control of storage, networking, memory, compute, and cooling infrastructure in a system. By tackling the optimization problem as a whole, one can then develop solutions at one layer that will be exploited at other layers.
5. An energy-efficient system must exploit component heterogeneity and dynamic adaptation. Heterogeneity would allow energy-optimized components to be brought to bear as application characteristics change. Dynamic adaptation would in turn enable the system to adapt and provision hardware components to meet varying workload and performance requirements, which in turn could eliminate resource over-provisioning and energy waste.
6. Existing techniques are sometimes localized and sometimes cross-layer, but often/frequently ad hoc. There is a need to invent techniques that can be orchestrated and analyzed in terms of how they compose together and interact.
7. It is essential to understand and properly model the emerging application and mobile user behavior and develop metrics for user experience that allow a (mobile) system to decide how much power/energy to allocate to a given computation.
8. Power-efficient systems of the near future will comprise of heterogeneous multi-core designs, with embedded accelerator "sub-cores" connected via heterogeneous interconnect elements. It is important to develop the programming model and software task scheduling support as well as low power/thermal management solutions for such a fabric. An important step in this direction is the development of System-Level Instruction Set Architectures that allow coarser-grained chunks of computation to be managed and scheduled, thereby enabling longer-term planning of communication and energy across large multi-core chips.
9. Other issues stand out as critical cross-layer power optimization and management design challenges, including (i) statistical variability and/or uncertainty about the workloads, circuit parameters, and operating environment of a target system, (ii) lack of standard interfaces that enable bi-directional information flow across different layers of the design stack and cross-layer optimization solutions, (iii) absence of a formal framework to capture and explicitly present power, latency,

bandwidth, reliability tradeoffs to the designers, and (iv) lack of standard benchmarks and evaluation techniques to enable realistic evaluation of proposed approaches and uniform comparison between approaches.

In the following, a more detailed description of the findings and recommendations of the four working groups (focus areas) are summarized.

Technology and Circuits Area Summary

To kick off the discussion, the working group considered a list of key questions, including the following. How will technology transition from 22 nm to 11 nm technology nodes (e.g. the International Technology Roadmap for Semiconductors roadmap) affect the design of power-efficient integrated circuits and on-chip modules? What are the impacts of 3D integration (using through-silicon vias), advanced on-die interconnect (including thinned silicon and fine-pitch silicon-silicon interconnections), and new memory devices (including Spin-Torque-Transfer RAM and MRAM) on the chip's power efficiency and reliability? What mix of super-threshold and near-threshold computing devices is desired for good power efficiency vs. circuit speed tradeoffs? The following is a summary of the group's findings and recommendations.

Motivation and Trends—With shrinking CMOS minimum feature sizes, higher chip densities, and lower operating voltages, it is the case that process, voltage, and temperature (PVT) variability have become significant challenges for designers of power-efficient integrated circuits. New methods for power optimization and management are needed that are robust against these sources of variability and uncertainty. Steeper subthreshold-slope switches will also be necessary to address the leakage problem and allow ultra low-voltage operation. Asymmetric devices (such as a drain-underlapped FinFET; or an asymmetrically source/drain doped FinFET) can lead to simultaneous improvement in both performance and reliability of key circuit elements such as static memory cell. On-chip non-volatile memory technologies will emerge and get embedded with the logic circuitry to achieve higher power efficiency. Heterogeneous components (digital, analog/RF/microwave, and optical) will be integrated on the same chip, most likely by employing 3-D IC (TSV-based) technology platforms. Near threshold operation being one of the most energy efficient design points, various low voltage CMOS circuit and architecture options at the near threshold region of computing will emerge and be used to achieve significant improvement in power efficiency when device/circuit co-design is considered for near-threshold operations.

Challenges and Opportunities—CMOS technology still has a lot of life in it. We expect that a new breed of multi-gate transistors such as FinFETs and Tri-gates will allow CMOS scaling to 10 nm or below. For this to happen, however, one must develop analysis and simulation tools to characterize properties such as delay, power/energy efficiency, and variation/aging tolerance of the new devices. In addition, we must also explore how new multi-gate transistor devices can be mixed with traditional CMOS devices for designing memory and logic cells that can operate at very low voltages.

Any suitable steep-subthreshold slope device will dramatically reduce the chip's leakage power, thereby making chips more power/energy-efficient and reducing thermal

problems. In the future, steeper subthreshold-slope switches (such as tunnel-FETs (TFET), NEM-FETs, IMOS, etc., will be necessary to address the leakage problem at the most fundamental level and to allow ultra low-voltage operation. Corresponding circuit design challenges will also need to be addressed for such non-CMOS devices.

For CMOS circuits, there are several important aging mechanisms that must be taken into consideration, such as bias temperature instability, time-dependent dielectric breakdown in the gate oxide and interlayer dielectrics, hot carrier injection, and so on. There are numerous opportunities for cross-layer optimizations to manage circuit aging. It is, however, vital to provide appropriate "hooks" to enable information transfer, from the point of view of modeling, real-time sensing, and real-time adaptivity, that enables these cross layer optimizations. Other important research problems are the development of models of appropriate complexity and accuracy at each layer and enabling easy cross-layer information transfer and strategies for controlling the amount by which a circuit may age without causing circuit failures, while ensuring that system-level performance metrics are optimized.

With respect to near threshold computing, a key challenge is the development of a cross-layer energy-delay optimization framework that spans technology advancement at the device layer and optimizations at the circuit layer. In particular, key elements of such a framework include methods for delay/energy/variation tolerance characterization of new multi-gate transistors such as FinFETs and Tri-gates, a multi-layer simulation hierarchy that can answer the question of how new multi-gate transistor devices can be mixed with traditional CMOS devices to produce memory and logic cells that can operate seamlessly between super-threshold and near-threshold voltages, and optimal provisioning of heterogeneous circuit elements and logic families that can operate in the near-threshold or super-threshold regimes.

Power delivery, regulation and power management circuits provide a foundation for delegating dynamic power/thermal management directives and must be optimized with cross-layer considerations. At the circuit level, the power distribution and DC-DC converters must be optimized to optimally tradeoff between power efficiency, supply noise, stability and transient response time. New materials and process technologies can significantly impact key characteristics of on-chip voltage regulators, and hence must be considered as part of the optimization process. The achievable system power efficiency is a joint function of the power loss in power delivery and the energy savings achieved through dynamic power management that runs on top of it.

Synthesizing circuits with different quality-energy efficiency values via voltage scaling, gate-level, or algorithmic transformations can achieve significant power savings. This is a promising possibility for systems that can tolerate imperfect results, such as signal processing, data mining, and learning. Cross-layer approaches are essential in this domain.

Research on non-Si devices for a better switch or for highly dense memory elements will continue and accelerate. Novel devices/technologies include carbon electronics, tunnel FETs, spin transistors and spin memories, and so on. It is expected that proper co-design and exploration of novel circuits suitable for such new devices are required to determine how they replace or serve as add-on to CMOS technology. Power models are available for

CMOS, but must be extended for emerging beyond-CMOS technologies. The bigger need, however, in both today's CMOS and future beyond-CMOS technology, is for true architectural level power models. Many attempts have been made, but the problem remains open and must be solved in order to allow design of the runtime environment and software level power management. In addition, there are no good abstractions today for expressing the power demands at high levels of abstraction that are usable in the currently practiced design methodology.

A fundamental way to bring about a revolutionary change in energy efficiency of future circuits is to move to a "non-thermionic" carrier injection-based device that can significantly lower the leakage power which is projected to be beyond acceptable levels for deep nanoscale CMOS devices. Thermoelectric devices and materials can be used to convert the heat flux from a hot spot into electrical energy that could be used to power peripheral circuits. An appropriate simulation framework needs to be developed that can accurately estimate the scavenging performance of these embedded thermoelectric devices, including parasitic losses due to the interfaces.

Circuits and Micro-architecture Area Summary

To kick off the discussion, the working group considered a list of key questions, including the following. What analytical models or simulation tools are needed to evaluate the power/energy efficiency, performance, and thermal issues for a system that is being designed? How do the circuit-level power optimization knobs such as multi-VDD and multi-VTH designs affect decisions at the micro-architecture level? What type and degree of heterogeneity (including general purpose cores with different EPI and IPS values as well as special purpose hardware accelerators) should be provisioned to achieve a good tradeoff between power, latency, and design cost? With the rising noise levels and dynamic faults in the circuits, what micro architectural techniques are needed to combat these transient errors? What about the aging effects (NBTI, EMI, etc.)? The following is a summary of the group's findings and recommendations.

Motivation and Trends—Impressive progress has been made in the past two decades on power optimization, which has made it possible to have billion-transistor-cellphones operated by battery power without recharging for more than a hundred hours. Instrumental to such success stories has been the introduction of standards such as the Advanced Configuration and Power Interface (ACPI). While ACPI has served the needs of technologies to date, newer CPOM approaches (such as supply voltage level optimizations that are independent of the operating frequency control) may not fit the abstractions laid out in the ACPI.

In today's multi-core processor chips, per-core dynamic frequency scaling capabilities, along with global voltage scaling, provide knobs that are tightly integrated with a collection of hardware sensors and with the power management firmware. By leveraging on-chip hardware counters, zone-based temperature sensors, and critical path monitors, chip power and performance metrics can be monitored and managed at run time for maximum performance, energy efficiency, and reliability. The cross-layer power optimization and management framework is, in general, a way for a lower-level layer to provide a feature that enables the control knobs, and an upper layer to utilize the control knobs. This

approach can achieve a power saving closer to the optimal than within-layer power optimization because a higher layer has more information from the applications and/or users and is aware of more accurate information about the system idleness. A well-defined standard interface specification between layers is crucial for cross-layer power optimization.

Architectural simulation research has a long history, but by no means is a solved problem. The previous simulation speedup techniques while very successful for single-core processors, are not easily extended to multi-core, parallel simulation due to the complications of inter-core communication, synchronization, and modeling of shared resources. Moreover, the introduction of heterogeneous components further complicates the simulation requirements. The opportunity for cross-layer design and optimization, for example, considering the use of near-threshold circuits and on-chip non-volatile memory technologies, also complicates the simulation models. Finally, technology scaling to 10nm and below and related PVT-induced issues require an increasing cross-layer optimization and management focus jointly for performance, power efficiency, and reliability in the coming years. Today's simulation infrastructures that model power and performance lack reliability modeling and overlook the existence of control systems that can dynamically alter operational conditions affecting quantities of interest. Multi-timescale operation of these control systems also increases the modeling and simulation challenges.

Challenges and Opportunities—As technology scales beyond 22nm, reliability of CMOS devices is decreasing. This has traditionally been overcome with guard bands. There are tremendous benefits in simplifying the design process and reduce costs if reliability can be relaxed. Furthermore, reducing the requirements of correctness can provide energy benefits because circuits can operate at common case operating points and reduce margins for guard band, etc. The implication then for the higher layers is that the hardware is not always correct. This introduces a fundamental *cross-layer problem* on how to expose the hardware's lack of correctness up through to the software.

Architectural heterogeneity means including cores of various computing and storage capabilities (e.g., small vs. big cores, SRAM vs. eDRAM vs. NVM) and extensive use of hardware accelerators in a system-on-chip design. Technological heterogeneity means including logic blocks composed of different device and interconnect technologies (e.g., CMOS vs. non-Si based devices, metal interconnect vs. RF or optical channels, near-threshold vs. superthreshold operating logic.) Heterogeneity provides a multitude of performance/power/reliability tradeoff opportunities. However, it is difficult to manufacture such heterogeneous systems at low cost (with high yield) in spite of promising path based on 3D integration. Other key challenges include (static) provisioning of this heterogeneity and making it reconfigurable are two Challenges of designing heterogeneous circuits. Furthermore, recent studies indicate that the potential performance/power/reliability improvement associated with the emerging device technologies and heterogeneous architectures are significantly reduced without careful design planning. This clearly highlights the need for modeling and tools infrastructures for effective adaptation of such systems.

Platform-specific optimizations (embedded systems): New abstractions and interfaces have to be investigated to coordinate the management of energy harvesting units, electrical

energy storage modules together with the management of power consuming modules. Some of the embedded systems, such as environmental monitoring systems, are natural candidates for approximate computing. Other embedded systems, for example embedded controllers of airplanes or automobiles, have a nearly zero tolerance for error. Reliability constrained energy optimization at design time should be investigated for different types of embedded systems.

Platform-specific optimizations (mobile computing systems): The microarchitecture of a mobile computing system should allow the trade-off of reliability for energy efficiency so that it satisfies the QoS and other SLA requirements. To enable correct operation of a system that can tolerate low-level errors in its building blocks, error detection, corrections, and recovery mechanisms should be investigated. Display and wireless interface are two major power-consuming components in a mobile computing system. Cross-layer power optimization techniques that target power minimization in both LCD and OLED-based displays should be considered. Another hardware trend of mobile systems is the use of multiple antennas for improved network speed and overall network capacity. This suggests an important cross-layer power optimization mechanism that power-manages the multi-antenna transceivers based on communication requirement.

Platform-specific optimizations (servers): We need to focus more on cross-layer channels to exchange power models and workload requirements between microarchitecture and upper-level applications in server systems. More accurate power models and highly standard constraint languages are believed to facilitate the cross-layer power management. New memory technologies such as NVM should be investigated. Heterogeneity, including cores of various computing capabilities and extensive use of accelerators, should be introduced to increase energy efficiency at this level.

Micro-architecture and Systems Area Summary

To kick off the discussion, the working group considered a list of key questions, including the following. What micro-architecture is needed for effective implementation and exploitation of CPOM? How should global interconnect be designed in systems with a large number of diverse and interacting IP blocks in order to power-efficiently support the required traffic patterns? What system architecture (memory hierarchy, computational resources, network-on-chip, communication protocol, etc.) is optimal for CPOM while meeting the application-level requirements? What advances are needed to support performance isolation (physical or virtual) in systems with shared resources? What is needed at the micro-architecture and system software interface to improve power efficiency in the cloud and to enable energy-proportional computing across various other platforms? The following is a summary of the group's findings and recommendations.

Motivation and trends—Environmental pressures and government rating systems have resulted in initial experimental R&D in the area of "green" or "zero-emission" data centers. In such approaches, the heat dissipated from data centers is reused to serve as energy sources for city heating and water desalination projects. Recent research has proposed the use of renewable energy sources (specifically solar power) to ease the power burden of servers. These ideas call for cross-layer optimization and modeling across energy supply, workload-driven demand and heat recycling.

3D packaging is an emerging technology that facilitates close integration of processor units and memory units. Thus, there is an opportunity to access DRAM main memory without traversing expensive off-chip interconnects. This helps overcome the power wall associated with achieving high bandwidth access to memory. However, we need new programming models to facilitate application development, new compiler strategies for the low-power cores on a 3D stack, and runtime systems that can manage co-ordination between many threads. Furthermore, the tradeoffs between capacity and bandwidth at different levels of the memory hierarchy could change significantly, and will require cross-layer optimization between architectural, programming systems, and application layers.

Writing parallel programming has been one of the major challenges in programming. Parallel programming is fundamentally hard, but we could also argue that the lack of tools to help programs is another cause. Assisting parallel programming is important both for programmer productivity and program efficiency in this heterogeneous multicore era. Hence, developing software tools that can help programmers and/or other software systems is a pressing need. Tools to provide which algorithms might be more efficient in the underlying hardware can also improve the quality of the code.

Not all computational threads are equally important and critical. Knowing criticality can provide many opportunities in various layers. For example, non-critical work can be easily sent to low-performance but energy efficient cores/memories. We can also control DVFS to improve energy efficiencies for non-critical work. Unfortunately, right now there is no good way for hardware to get this criticality information from upper layers. There should be some generic ways of transferring this information from software systems to the underlying hardware.

Instruction set architecture (ISA) has been a great way of hiding the complexity of hardware or software by providing a concrete and stable interface between two layers. Consequently, quite a lot of information from software is lost on the way. Examples are criticality, data-flow information, data locality, data movements etc. As a result, hardware must regenerate information at runtime, such as finding critical threads, critical data etc. Hence, this is the time to re-think the granularity of ISAs. If software can pass hardware more summary information about its characteristics and requirements, hardware can use that information to improve performance and power.

Challenges and Opportunities—Lower levels of abstraction in the design hierarchy have better coupling between layers, because their specification is based on established models, for example, physical circuit models, Boolean algebra, register-transfer languages, etc. One of the things that inhibit cross-layer optimizations from the system level to the (micro)architectural layer and downwards is the lack of a formal specification language. This area is in its infancy in terms of experimental research or commercial systems such as BlueSpec and SystemC. Significantly more investment is needed to bridge the gap between the system architecture level and the physical implementation layers.

In addition to design-time specifications across the architectural boundaries, there is also a need for higher abstraction layers to assist in system-level mapping and scheduling choices, particularly when dynamic and heterogeneous parallelism are involved. We argue for System-Level Instruction Set Architectures (ISAs) that allow coarser-grained chunks of

computation to be managed and scheduled. By avoiding per-instruction handling, energy overheads can be greatly reduced, and coarser-granularities also assist longer-term planning of communication and energy planning across large multi-core chips.

The cost of communication is now significantly higher than the cost of computation. In addition, there is an order of magnitude difference in communication energy depending on where data is found on the chip. Similarly, main memory organizations are also highly distributed. The average cost of communication will again vary by an order of magnitude depending on the quality of data placement in main memory. Currently, higher levels of the system stack (application, OS, compiler) are largely unaware of data placement in caches and memory modules. This disconnect leads to an order of magnitude increase in communication energy on average. Communication distance can be minimized by not only placing data in appropriate regions of the memory space, but by also moving computations to the memory when appropriate. . Such computation migration requires much more extensive support from the operating system and programming models than what currently exists.

Power optimization needs have driven the system architecture community towards heterogeneous multi-core designs, with embedded accelerator "sub-cores" connected via heterogeneous interconnect elements. However, the programming model and software task scheduling support still lack the formalism required to exploit the full potential of this emerging paradigm shift. Significant new research investment is needed to facilitate co-design and co-optimization across the application, system software and architecture layers.

Systems and Applications Area Summary

To kick off the discussion, the working group considered a list of key questions, including the following. What will be the dominant workloads (i.e., computational load and memory/network traffic pattern) for different application classes in 5-10 years? What are the applications/workloads that are likely to place special requirements on CPOM? How do we achieve system-level power or energy efficiency under latency, bandwidth, thermal, or cost constraints? What level and amount of re-configurability (both at the circuit and architecture levels) are needed to enable power efficient operation of a target system under various workload conditions and reliability/fault tolerance requirements? What is the support needed at the hardware and system software level to enable such reconfiguration? How is one to achieve optimizations that combine dynamic control, algorithmic transformations, and compiler optimization? The following is a summary of the group's findings and recommendations.

Motivation and Trends—All types of computing devices are now energy-constrained either due to performance limitations, battery life, or electricity cost. While it is hard to find a silver bullet that can address energy issues across the space of computing applications, parallel operation has been heavily explored in recent years to address energy-scaling challenges by exploiting simpler, more parallel components. Continued work is seen in this area especially as new programming models allow programmers to exploit heterogeneous processing systems. In addition to parallelism, the trend towards mobile computing places increased burden on system designers to consider cross-layer, holistic solutions that are cognizant of the users' preferences in addition to the application, system software, and

architecture. Recent trends in this area include energy process monitors, hybrid power knobs to enable longer and deeper use of sleep states, and increased used of reactive wake-on events. Coordinated, cross-layer tuning of applications and architecture is another ongoing trend that has seen success in embedded system, HPC platforms, and in datacenter scale computing. There is potential for such approaches to be applied more broadly to computing systems.

Challenges and Opportunities—Complexity is increasing at all layers of the computing stack, and power-management techniques must efficiently navigate this complexity. Cross-layer optimizations such as algorithmic transformations, dynamic runtime systems, and static compiler transformations provide a large design space that must be navigated to reduce system-level energy consumption. Application programmers are not likely to want to be involved in power-management, and we must develop middleware layers that provide the interface between the applications and the system.

Given these challenges, there are significant opportunities that can be exploited with cross-layer approaches for deeply embedded, mobile, and server systems.

***Deeply Embedded Systems and Sensors*:** Deeply embedded systems and sensors will proliferate with the development of 3D stacked sensors, solar cells, batteries, and microcontrollers. As the capabilities of these platforms expand, opportunities exist to partition applications between the client and the cloud allowing the projection of huge amounts of compute resources into the traditionally low-performance embedded system domain. Balance between local compute, communication, and compression strategies can target large improvements in energy-efficiency and application capability.

***Mobile System*:** Understanding of emerging application and mobile user behavior can provide an important lever on system development. The challenges of cloud computing also arise in mobile systems, but user experience is paramount, and metrics for user experience must be developed that allow the system to decide how much power to allocate to a given computation. Mobile systems that interact with the cloud must balance radio communication latency and power costs with the ability to leverage the cloud to reduce compute power on these platforms.

***Servers*:** Compute density is a primary design metric for large datacenters, and this is limited by thermal density. Operating cost (energy) is also a primary motivation for energy-efficient server design. There is a tremendous opportunity to exploit heterogeneous compute infrastructure with processing elements that have varying power/performance characteristics for compute, memory, interconnect, and storage. Exploiting these opportunities require identification of bottlenecks and runtime systems to leverage this heterogeneity, and designers must also address potential system costs such as multi-version compilation and tuning.

Putting It All Together

Cross-layer approaches to energy efficiency have the potential to deliver low power with significantly lower performance overheads than current intra-layer approaches. By distributing power/thermal efficiency concerns across the full design stack, these approaches can take advantage of the information available at each layer of the design to

improve system efficiency while tolerating variation, aging, and even errors. The best of these approaches can adapt to varying application characteristics and requirements, operating environments, and changing state of the hardware components.

The work necessary to achieve energy-efficient systems crosses the entire computing ecosystem—from CMOS devices and VLSI circuits to computer architectures and platforms to system software and user applications. In other words, no one player can cause change by herself. Cooperation across many disciplines and organizations is necessary to drastically change the computing system design paradigm in the direction of energy efficiency.

The United States', and indeed the world's, economy depends critically on energy efficient operation. The government and industrial leaders play key roles in funding new research areas, providing foundations and bricks for infrastructure advancement, and in advocating and enforcing standards to enhance sustainability across the board. The research challenges and opportunities described in this report provide US industry and government funding agencies with directions for developing many of the technologies and approaches that are needed to develop energy-efficient systems of the future and ensure sustainability of the information technology ecosystem.

In the following, we present the detailed area reports. Detailed pre-workshop position statements from various CPOM workshop invitees and attendees may be found under "Position Statements" links on each of the Area Overview pages at http://atrak.usc.edu/cpom/ .

Area 1: Technology, Circuits, and Beyond

Area Leader: Kaushik Roy

Co-Editor: Peng Li

Other Area Members: Mohamed Allam, Kaustav Banerjee, Keren Bergman, David Blaauw, Eby Friedman, Lei He, Payam Heydari, Farid Najm, Michael Orshansky, Sachin Sapatnekar

Our sub-group considered the different challenges, opportunities, and possible solution paths for designs in sub-15nm technology nodes. The technology options were divided into different application domains classified in terms of power consumption – Embedded/Mobile Computing (Milli-watts), Servers (Watts), and Sensors and networks (Micro-watts.) The sub-group also felt that cross-layer designs, in particular with the availability of new technologies such as spin memories, III-V devices, tunnel FETs, and other non-CMOS devices, will become even more important. Heterogeneity in terms of device and interconnect technologies will lead to the possibility of very complex and low power designs.

The following are the different research areas and research needs that the sub-group identified.

Success/Failure Stories

Power delivery – Circuit design

In a mobile platform, power delivery includes power generation, regulation, and distribution from the battery through several levels of voltage regulation to the billions of transistors (or loads) on integrated circuits. This complex system includes a hierarchical combination of power regulators and decoupling capacitors, decreasing in current and response time. Each stage of the power delivery system regulates the current and voltage required at that level of the power chain. The choice of topology for the regulators is strongly dependent on the efficiency objectives and area constraints. Furthermore, the local impedance between stages also deeply affects the ability of the power delivery system to accomplish the design goals. This system therefore requires great care in design and analysis to ensure that the locally distributed voltages are properly designed to meet the needs of the loads while not expending excessive resources.

The efficient delivery of on-chip current is required by every integrated circuit. If the load varies more quickly than what the nearby local regulator or decoupling capacitor can provide current, the transistor loads will not perform as required. If the supply voltage level provided to the loads is not as expected and needed, the active devices will again not perform as desired. The input and output of each regulator needs to be properly regulated. If not, the power line will oscillate, affecting the circuit behavior. Furthermore, these power lines are global in nature, and noise can propagate through the nearby lines as interconnect

coupling noise and/or through the substrate as substrate noise. Finally, certain topologies can be used such as switching buck converters which use a passive inductor-capacitor to filter out the voltage ripple, while maintaining high efficiency and providing significant current. These switching converters, due to the large inductors and capacitors, require significant silicon area, greatly increasing cost.

	Milli-Watts Designs (Emb/Mobile)	**Micro-Watts Designs (Sensors)**	**Watts Designs (Servers)**
Challenges	Temp./cooling Battery (mobile) Reliability Parameter variation Power delivery	Energy harvesting Variability Energy efficiency Ruggedness Longevity	Temp. cooling Reliability Power dissipation Parameter variation Power delivery issues (on/off chip)
Solutions & Needs	(1)Device/circuit/arch. co-design (2) New generation of universal memories (3) Tech. heterogeneity Context-aware design (4) Approx. computing (5) Aging management (6) Reconfigurable circuit/system design (7) Enable adaptability	(1) Device/circuit/arch. co-design (2) New generation of universal memories (3) Energy harvesting (4) Tech. heterogeneity (5) Context-aware design (6) Approx. computing (7) Aging management (8) Enable adaptability	(1) Device/circuit/arch. co-design (2) New generation of universal memories (3) On-chip power generation, regulation, and delivery (4) Active cooling (5) Tech. heterogeneity (6) Context-aware design (7) Approx. computing (8) Aging management (9) Enable adaptability

There are many important and interesting ways for circuits to fail from these power delivery-related mechanisms. A classic manifestation of a failure mechanism from noise produced by a power delivery system is phase jitter in an oscillator. Noise induced by the power network propagates into an oscillator, for example, a phase locked loop, causing the oscillating output system to shift in time. Other common examples of power deliver design issues causing circuit failure are increased levels of delay uncertainty due to the power supply noise as well as other, sometimes related, noise sources (such as interconnect and substrate coupling noises.) Probably, the most common example of a power delivery system causing a system to fail is the system not providing sufficient current within the proper time for the circuit to behave as designed. Essentially, the voltage is lower than expected due to IR and L di/di induced transient voltage drops, which delays the circuit response while adding to the delay uncertainty. All of these cases exemplify the importance

of the power delivery system on system functionality and behavior. There are many examples of industrial circuits failing due to these mechanisms. Power delivery remains as one of the most important and omnipresent issues within the entire circuit design process.

Reliability

Traditional reliability analysis for gate oxide TDDB uses the "area-scaling" model at the transistor level, which predicts circuit lifetimes under TDDB stress, based on the assumption that any transistor-level failure causes a circuit failure. However, based on a cross-layer analysis at device/circuit levels, it has been observed that not every device failure causes a circuit failure, due to inherent resilience in the circuit where, for example, DC paths to a supply node will successfully fight off a weak leakage path to ground. An improved model that accounts for these effects across the device/circuit layers has shown that the actual circuit lifetime is 5-8x longer than the prediction from the traditional area-scaling model.

Cross-Layer Research Needs

Energy harvesting and energy recycling

Energy harvested from the environment such as solar, pressure, vibration and communication-cables may be used as the primary energy source for ultra-low power platforms including sensor nodes. It could be a significant power source for mobile devices and desktop computers to prolong battery life or save energy. In terms of energy recycling, we should look at the entire system since the primary power consumers could be communication components and displays. It is important to collect and reuse energy from displays and communication components. There has been preliminary research on harvesting energy from backlighting of LED displays and more studies on zero-power switch for DC/DC converters and power amplifiers. While novel and cost efficient circuits need to be developed for energy harvesting and recycling, cross-layer optimization could be performed to maximize the opportunity of energy recycling, and embedded energy storage means could be allocated to account for temporal variation of power dissipation.

Memory, device, tech. heterogeneity

The most fundamental way to bring about a revolutionary change in the power consumption and energy efficiency roadmap of future (beyond 10 nanometers) integrated circuits is to move to a "non-thermionic" carrier injection based device that can significantly lower the "leakage" or "wasted power", which is projected to be beyond acceptable levels for deep nanoscale CMOS devices. Energy-efficiency can be achieved by lowering both dynamic and leakage power consumption. Lower power also leads to lower operating temperatures of electronic devices resulting in improved reliability, since most reliability mechanisms tend to degrade with increasing temperature. However, any significant lowering of power using traditional techniques will become non-trivial beyond the 10-nanometer technology node, although some avenues for incremental improvements may exist. This is due to the fact that in such nanoscale devices, the most effective knob used for lowering power dissipation, namely the power supply voltage, cannot be scaled as rapidly as in earlier technology generations without incurring significant performance

penalty arising from the fundamental inability to simultaneously reduce the threshold voltage. Simultaneous scaling of threshold voltage, which is essential for maintaining a certain ON to OFF ratio of the device currents (that is essential in digital circuits where the transistors are used as switches), leads to a substantial increase in the subthreshold leakage (OFF state) current (Fig. 1(a)), owing to the "non-abrupt" nature of the switching characteristics of MOSFETs (Fig. 1(b)), thereby making the devices very "energy inefficient".

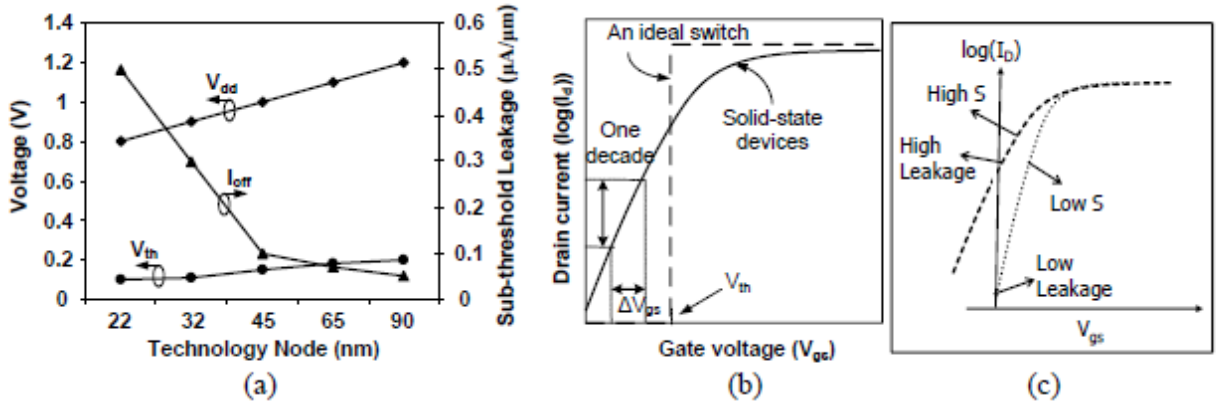


Fig. 1. (a) CMOS technology scaling trends as predicted by ITRS, (b) Comparison between switching characteristics of an ideal switch and that of solid-state devices, (c) Relation of subthreshold swing to leakage current.

The metric that is used to capture the abruptness of this switching behavior is known as the subthreshold swing (inverse of the subthreshold slope = $dlogI_d/\ dV_{gs}$ in Fig. 1(b)), which has a fundamental lower limit of 2.3kT/q = 60 mV/decade for MOSFETs, indicating that in order to reduce the source-to-drain current by one decade, one must reduce the gate voltage by 60 mV. This value is essentially due to the "thermionic emission" of carriers from the source to the channel region in MOSFETs. Practical nano-scale CMOS devices have even higher subthreshold swings (80-90 mV/decade) (Fig. 1(c).) Hence, although lowering of the subthreshold swing below 60 mV/decade is a non-trivial task, it is extremely critical for achieving any significant improvements in energy-efficiency of electronic products.

Any suitable steep-subthreshold slope device will dramatically reduce the chip's leakage power, thereby making them more power/energy-efficient, and lower the thermal problems. In future, steeper subthreshold-slope switches (such as tunnel-FETs (TFET), NEM-FETs, IMOS, etc. will be necessary to address the leakage problem at the most fundamental level and to allow ultra low-voltage operation. Corresponding circuit/system design challenges will also need to be addressed for such non-CMOS devices. Device-circuit co-design will be vital.

For memory technologies, ultra-dense, low-power as well as high performance (SRAM or SRAM-like) on-chip memory banks will need to be integrated in close proximity to the logic. Non-volatile memory technologies including R-RAMS, PC-RAM, STT-RAM and so on will also need to be co-integrated with the logic circuitry. Integration of Analog/RF/microwave and optical components would be needed to realize ultimate

power/energy optimized "smart systems" in the future. Such heterogeneous integration of technologies would most likely be realized via the 3-D packaging or 3-D IC (TSV based) technology platforms.

Device/Circuit/Architecture Co-design

It is expected that the transistors in the sub-10nm would primarily be Multi-gate FETs, having much reduced short-channel effects than their bulk counterparts. However, effective usage of such technologies would require proper understanding of circuit behavior and subsequent optimization of devices to meet the performance and power efficiency requirements. Such co-design, for example in FinFETs, would consider fin orientation of devices, gate underlapping, width quantization to come up with the best logic and memory architecture. Let us consider an example: standard 6-transistor SRAMs have conflicting requirements for read stability and writeability. This becomes even more important in deeply scaled technologies where parameter variations are expected to be larger. Asymmetric devices (such as a drain-underlapped FinFET; or an asymmetrically source/drain doped FinFET) used as access transistors in 6T SRAMs can lead to simultaneous improvement in both readability and writeability, while having better short channel effect, improved edge direct tunneling current; albeit with slightly increased variability.

The need for low voltage CMOS has the design community explore various circuit and architecture options at the near threshold region of computing (near threshold operation being one of the most energy efficient design points.) However, designing circuits in the subthreshold or near threshold region, using standard "high performance" transistors, may lead to energy inefficiency. It is an established fact that for scaled super-threshold transistors it is essential to have halo and retrograde doping to suppress the short channel effects. The main functions of halo doping and retrograde wells are to reduce drain induced barrier lowering (DIBL), prevent body punch-through, and control the threshold voltage of the device independent of its sub-threshold slope. However, in sub-threshold operation, it is worthwhile to note that the overall supply bias is small (in the order of 200mV~400mV.) Consequently, the effects of DIBL and body punch-through are extremely low. Further, as long as we have a fixed I_{OFF}, the steeper the sub-threshold slope *(S)* is, the higher the device performance will be. Hence, it can be qualitatively argued that halo and retrograde doping profiles are not essential for sub-threshold device design. The absence of the halo and retrograde doping has the following implications:

(a) A simplified process technology in terms of process steps and cost,
(b) A significant reduction of junction capacitances, resulting in faster switching speed and lower energy consumption.

It should, however, be noted that the doping profile in the optimized devices should have a high-to-low doping profile. It is thus necessary to have a low doping level in the bulk of the device to achieve the following:

(a) Reduce the capacitance of the bottom junction,
(b) Reduce substrate noise effects and parasitic latch-up problems.

Initial results of analysis suggest more than 50% improvement in power efficiency when device/circuit co-design is considered for sub-Vt operations.

Research on non-Si devices has also started in earnest in quest for a better switch or for highly dense memory elements. Such devices/technologies include carbon electronics, tunnel FETs, spin transistors and spin memories, etc. It is expected that proper co-design and exploration of novel circuits suitable for such new devices are required to determine how they replace or serve as add-on to CMOS technology.

Active cooling

High performance processors of the future are expected to have hot spots, exhibiting power density close to 400W/cm^2. Effective cooling of such hot spots would require innovations in thermoelectric devices and materials. Such devices can also be used to convert the heat flux from a hot spot into electrical energy that could be used to power peripheral circuits. The thin-film thermoelectric devices are capable of handling heat flux in the order of 100-300 W/cm^2 and are suitable for on-chip energy scavenging/cooling applications. These materials are usually super-lattices of Bi2Te3 and Sb2Te3 reaching a ZT of 2. Such devices when embedded in a chip-level package could lead to heat recovery for energy efficient computing as well as possible application for hot-spot cooling in high performance processors. An appropriate simulation framework needs to be developed that can accurately estimate the scavenging performance of these embedded thermoelectric devices, including parasitic losses due to the interfaces.

Aging management

Reliability has emerged as a major challenge in future technologies, and it is projected that its impact will become even more serious in the future. Aging variations, caused by stresses experienced by a circuit, can be manifested as (a) parametric failures, corresponding to shifts in circuit behavior over time, or (b) catastrophic failures, which cause a circuit to become nonfunctional.

For CMOS circuits, there are several important aging mechanisms that must be taken into consideration, such as bias temperature instability (BTI), time-dependent dielectric breakdown (TDDB) in the gate oxide and in interlayer dielectrics, hot carrier injection (HCI), electro-migration (EM), and thermo-mechanical stress. These variations can be acutely sensitive to factors such as the supply voltage level, changes in environmental parameters (notably temperature), and process parameter variations. Given the relationships between power, temperature, supply voltage changes, and reliability, it is essential to treat reliability as a “first-class citizen” in future designs. Along with power management (which controls the total power to a design) and temperature management (which controls the thermal hot spots), aging management must be a crucial component of future designs.

Aging management primarily entails two aspects:

(a) Developing models of appropriate complexity and accuracy at each layer and enabling easy cross-layer information transfer that enables aging analysis at all levels.

(b) Determining strategies for controlling the amount by which a circuit may age without causing circuit failures, while ensuring that system-level performance metrics are optimized. Such strategies include, but are not restricted to, pre-silicon methods such as the use of redundancy or guardbands to account for temporal performance degradation, and post-silicon methods such as real-time sensing and adaptation.

The precise aging mechanisms associated with post-CMOS technologies are likely to become more apparent as the technologies mature, but it seems likely that any new technology will also experience failures in time, and a research agenda similar to that for CMOS technologies may be extrapolated.

There are numerous opportunities for cross-layer optimizations to manage circuit aging. The most effective optimizations can be made at higher levels—microarchitecture, system, and applications. It is vital to provide appropriate "hooks" to enable information transfer, from the point of view of modeling, real-time sensing, and real-time adaptivity, that enables these cross layer optimizations and implements them at the circuit level. This information transfer will enable capabilities at various levels—e.g. activating/deactivating various units and controlling the level of permissible errors at the system/application level. The latter approach may leverage redundancy/speculation, tolerating low-level errors that are never expressed at the application level, or enabling energy-efficient approximate computing, as outlined in a next section.

On-chip power delivery/regulation

The ability to convert and deliver power to on-chip devices in an efficient manner is an integral part of system power/thermal management. Power delivery, regulation and power management circuits provide a foundation for delegating dynamic power/thermal management directives and must be optimized with cross-layer considerations.

There has been an increasing awareness that the entire power conversion, regulation and distribution chain shall be considered as a complete electrical sub-system and optimized for targeted applications (e.g. mobile vs. desktop.) At the circuit level, the power distribution and DC-DC converters must be optimized to optimally tradeoff between power efficiency, supply noise, stability and transient response time. Switching and switched-capacitor DC-DC convertors, linear voltage regulators, power grids and decoupling capacitors and their proper composition shall be considered as key components of the power delivery network. New materials (e.g. magnetic materials for low-loss on-chip inductors) and process technologies (e.g. deep-trench capacitors) can significantly impact key characteristics of on-chip voltage regulators, and hence must be considered as part of the optimization process. In addition, limits on the amount of delivered power and passive decoupling must be considered during the circuit-level design optimization.

Other cross-layer challenges and opportunities exist due to the fact that on-chip power delivery and regulation shall be optimized synergistically with high-level power management schemes to maximize overall system performance. Supply noise shall be controlled through circuit design with a proper understanding of workloads for a given implementation platform (e.g. servers vs. embedded systems.) Power management policies interact with power delivery differently and have different response time requirements in order to manage the chip power at an optimal temporal granularity. The achievable system

power efficiency is a joint function of the power loss in power delivery and the energy saving achieved through dynamic power management that runs on top of it. Co-optimization of the two is the key to optimize the interactions between the power delivery and dynamic power management. Key circuit and policy level parameters need to be determined jointly and optimized with workload awareness. Cross-layer optimization also need be targeted to provide adaptability to the workload, device variability, and aging phenomena.

Enabling adaptability by predicting the system workload

Digital and analog circuits may be designed with different degrees of freedom to change circuit behavior. For example, the voltage of the circuit can be controlled dynamically to enable operation at higher or lower frequencies as in dynamic voltage and frequency scaling (DVFS) techniques. Although DVFS has been discussed in the literature for a long time, some challenges are still stalling broader adaption of DVFS in the industry. To date, the ability to predict computational workload at the system level is a challenge. Therefore, DVFS is not being used in a truly dynamic manner as it implies in naming. Consequently, power reduction capabilities at the circuit level are not being completely utilized due to the lack of workload prediction schemes at the system level.

Approximate computing and designing with unreliable components

An efficient way to improve computation power is to relinquish the requirement for perfect correctness of every operation. That may come in two forms: intentional trading of achievable computation quality for energy, and relaxation of protection guarantees for certain types of transient/intermittent errors.

Synthesizing circuits with different quality-energy efficiency values via voltage scaling, gate-level, or algorithmic transformations can yield significant power savings. This is a promising possibility for systems that can tolerate imperfect results, such as signal processing, data mining, and learning. Cross-layer approaches are essential in this domain: one needs to know where – from the algorithmic perspective – approximate computation is permissible. For example, when implementing control logic of a finite state machine no errors can be permitted. At the same time, a block performing an image compression algorithm, such as IDCT, can permit errors. Thus, in order to enable such importance-driven or significance-driven computation, hooks in software must be made available to identify operations admitting imprecise/approximate evaluation.

It may be possible to reduce energy by discarding algorithm steps that contribute less to the final quality of results. Alternatively, adaptively setting the precision of the arithmetic unit output may be used to save energy. Joint use of error-prone and simpler error-correcting blocks can reduce the overall power dissipation. Algorithmic transformations, such as identifying and skipping the unnecessary computations, can also be effective. Demonstrations of such measures on signal-processing circuits, including IDCT/DCT blocks have been made. More research at the level of RTL/logic optimization is needed to generalize such constructions.

A related principle is that when certain blocks can tolerate errors due to their relative significance, less energy must be dissipated for soft error protection. Such soft errors are

transient in nature and may be due to power supply noise, coupling noise, and/or high-energy alpha particle strikes.

Power estimation and modeling

Cross-layer power optimization and management depends on the availability of robust power estimation and modeling capabilities. In the context of chip design, these capabilities must span the levels of abstraction from devices and technology to circuits and architectures, as well as the variety of heterogeneous components and technologies. At the level of circuits and devices, power models are available for CMOS, but must be extended for emerging beyond-CMOS technologies. The bigger need, however, in both today's CMOS and future beyond-CMOS technology, is for true architectural level power models. Many attempts have been made, but the problem remains open and must be solved in order to allow design of the runtime environment and software level power management. The modeling difficulties relate both to the difficulties in comprehending the circuit timing context and the wide range of possible implementations, but they also relate to the fact that there are no good abstractions today for expressing the power demands at high levels of abstraction that are usable in current design methodology. These problems must be overcome so as to enable optimization of power across multiple layers of the design hierarchy.

Area 2: Circuits, Microarchitecture, and Beyond

Area Leader: Jason Cong

Co-Editor: Lin Zhong

Other Area Members: Christopher Batten, Naehyuck Chang, Sandeep Gupta, Engin Ipek, Bill Joyner, Eren Kursun, Renu Mehra, Vijaykrishnan Narayanan, Qinru Qiu, Karthick Rajamani, Karu Sankaralingam, Mircea Stan

In this section we first review the progress and success in the circuit and microarchitecture areas for power optimization, and also lessons learned by the research community and industry in exploring various directions and techniques for power optimization. Then we discuss the challenges ahead and opportunities for cross-layer power optimization, with emphasis on heterogeneity, specialization, standardization, memory power minimization, cross-layer co-optimization, platform-specific optimization, and exploration of bio-inspired systems.

Success/Failure Stories

Impressive progress has been made in the past two decades on power optimization, which has made it possible to have billion-transistor-cellphones operated by battery power without recharging for more than a hundred hours. We think it is worthwhile to review the past successes in power optimization, especially at the circuit and microarchitecture levels, before we map out new research directions.

Success Stories

We think that the following achievements have been instrumental for power management and optimization in the past two decades.

Success in level of abstraction for power modeling

Industry has partially succeeded in providing various levels of abstraction for power modeling. In particular, very accurate gate-level power models for both dynamic and leakage power dissipations are available from silicon foundries and are integrated into the commercial tool flow. Characterizations are available at multiple corners and at different supply voltages. This allows tools to make informed optimization decisions that cover different corners, voltages, and different conditions on the chip.

Standardization

Standardization is critical for cross-layer power optimization. The Unified Power Format (UPF) and Advanced Configuration and Power Interface (ACPI) are good examples. UPF enables users to bridge the gap between the system level and the microarchitecture level. It allows users to specify power domains (at a fairly fine-grained level), the power network

including power supplies and switches, characteristics of the power domains (which elements need to be state-retained during power gating), and the interaction between the power domains (including isolation and level shifting interfaces.) In addition, the state combinations between power domains are defined. This allows the synthesis tools to implement hardware that respects the power intent of the user. System-level domain requirements that are captured in ad hoc formats, like Excel spreadsheets, can now be converted into a design specification at the next lower level of abstraction.

The Advanced Configuration and Power Interface (ACPI, in use on x86 systems) is the foundation for DVFS and processor idle state management on x86 systems. ACPI helped bring together advancements at the operating system level (and by extension, at the application-level), processors (micro-architecture, circuit designs), and devices layers with firmware support. An aspect of this related to processor micro-architecture is the support for *pStates* (voltage-frequency control abstraction) and *cStates* (core, thread, cache, chip idle mode control abstraction.) OS (or suitably empowered applications, or hypervisors), directly or indirectly through firmware, direct power management actions for the processor, memory system, and I/O devices where supported. Power-aware applications and runtime systems can also leverage the capabilities through extensions provided by the OS for tapping the mechanisms. While it served the needs of technologies to date, newer approaches (such as supply voltage level optimizations that are independent of operating frequency control) may not fit the abstractions laid out in the ACPI. Extensions or newer standardization frameworks might be needed to support the new power optimization capabilities.

Cross-layer optimization in mobile systems

The initial success of cross-layer power management techniques has provided a significant improvement in the overall efficiency of mobile systems. One good example is the advancements made in the mobile space. We now have the equivalent compute power at our disposal that a supercomputer had a couple of decades ago, or a desktop computer had a decade ago. But today that compute power is battery-powered and in the form factor of a cell phone. At the same time, in the mobile platform space there are many variations, with some products having better within-layer figures of merit (e.g. highest performance, largest screen, highest power efficiency, best radio, etc.), yet the overall most successful solution (iPhone) is one that uses a holistic design approach with tight vertical integration and cross-layer optimization. The iPhone trades features for user friendliness. While it is not the best on any individual metric, it excels in usability.

Judiciously exposing microarchitecture resources to the compiler can often result in system-level energy optimization opportunities that take advantage of both the runtime information possessed by the hardware and the global information possessed by the compiler. Scratchpad memory is a successful example of a hardware feature where the compiler's knowledge of future memory access patterns can be leveraged to exploit locality with lower energy than a hardware-managed cache. Loop stream detectors and buffers, which detect loops in hardware and buffer the decoded instruction stream, are another example where cross-layer optimization is useful: if the compiler is aware of the loop stream buffer, it can adjust the number of times a loop is unrolled, making sure that the number of instructions within the loop body is smaller than the buffer's capacity.

Cross-layer power optimization in server designs

Recent microprocessor chips (e.g. IBM's Power7 processor chip) illustrate a full-stack example of the corresponding improvement that can be achieved through cross-layer power optimization. By leveraging deep trench capacitors in 45nm SOI processes, the Power7 processor incorporates an eDRAM-based L3 cache design to realize a highly energy efficient cache hierarchy. This has resulted in a 5x lower standby power compared to the SRAM-based L3 cache designs. A range of processor idle/sleep modes was employed in the design for maximum energy efficiency in the 8-core/SMT4 design. Per-core dynamic frequency scaling capabilities, along with global voltage scaling, provide knobs that are tightly integrated with a collection of hardware sensors and with the power management firmware.

By leveraging on-chip hardware counters, unit-level temperature sensors, and critical path monitors, chip power and performance metrics are monitored and managed at run time for maximum performance and energy efficiency. Another example of full-stack management is shown between the packaging and architecture-level designs in the P7 chip. On-chip digital thermal sensor readings are dynamically monitored, and the measurements are used for dynamic fan speed control; this results in an additional 21% power savings.

In recent years IBM also demonstrated hardware characterization-based resource and power management for improved power efficiency. By passing core-to-core variation characteristics to the system hardware and software management, on-chip controllers can intelligently select the voltage and frequency levels (as well as customizing the power management techniques.)

Lessons Learned

In the process of searching for efficient power optimization techniques, the research and industry communities learned a number of lessons from the unsuccessful attempts. We list several in this section, since we think it is important to learn from these experiences.

Overexposure of lower layer

When microarchitecture exposes hardware features to software so that the software can leverage them for efficiency, it may make the interface (i.e. ISA) fat, leading to innate inefficiency. Although VLIW remains the dominant micro-architectural paradigm in today's DSP processors due to its potential for extracting high levels of instruction level parallelism (ILP) with low overhead, the application of such a technique in the general-purpose domain has been tried and largely rejected by the market. The problem stemmed partially from an overly ambitious cross-layer development agenda that relied heavily on compiler support to extract ILP in general-purpose codes. Also, the extra hardware support that was added to EPIC ISAs to ease the compiler's job has, in turn, imposed additional energy overheads.

Overuse of Top-down methodology

While attractive in shortening design cycles, top-down design methodologies require caution and proper cross-layer hooks to keep inefficiencies under control. In processor design, except for some embedded small soft core examples, higher performance processor

solutions require different levels of customization—all the way from pure hard cores to more intermediate firm cores. The main reason is that aggressive power and performance metrics need either accurate cross-layer hooks, or extensive low-level optimization.

Binary translation

Binary translation for energy efficiency has been tested and tried as an effective way of improving the efficiency of x86 binaries. The technique did not meet with wide success due to the overhead of binary translation hardware and lack of ways to accurately evaluate the impact of high-level optimization.

Transmeta is an example in our discussion of "binary translation for energy efficiency." The idea that one can use a very simple core design for energy efficiency and do binary translation to enable execution of legacy software just never seemed to really pan out. The energy overhead of binary translation always seemed to outweigh the benefit of using a simpler core design. We believe that the big low-power win for Transmeta comes from its aggressive use of DVFS.

Misuse of power gating in overly fine-grain power management

Circuit-level optimizations such as power gating can be managed potentially at different levels. But they also often need information from multiple levels. Depending on which level the decisions are taken, appropriate input must be made available from other levels; e.g. if fine-grain power gating control is to be implemented at the circuit/micro-architectural level, the right abstractions from software to the hardware logic might be needed to initiate power gating at the right moment to make it somewhat proactive and not just purely reactive. Conversely if coarse-grain power gating is to be managed from the software layers, they need to be adequately informed of the delays and energy overheads of invoking power gating levels at the lower layers. Evaluation infrastructure (simulators) should incorporate delays, and overheads at each of the concerned levels (circuit, microarchitecture, software) to accurately capture the benefit/cost of the power gating technology.

More generally, research studies sometimes ignore the modeling of the impact at each layer or the need for the right communication interfaces. This results in incorrect recommendations on the value and/or implementation approach needed for a successful incorporation of the technologies. Similar observations can be made regarding other circuit-level optimizations such as the employment of multi-V_{DD} capabilities for combating variation issues with technology scaling/low-voltage operation.

Moving Away From Overclocking and performance-driven design

In the 2003-2004 time frame, there were multiple reports on high temperatures (operations close to thermal limits) and performance degradation due to throttling in a microprocessor design. Significant changes were made in architectures to address the power efficiency and power density challenges (clean slate approach.) This also marked an important transition in the industry—that is, moving away from overclocking and moving more towards power-aware design. The positive outcome was the recognition of power/power-density aware design, which also resulted in a wave of research on

temperature-aware design/temperature-sensing infrastructure that eventually improved system power efficiency further.

Coupling microarchitecture innovation with programming models and compiler support

Some success has been demonstrated in coupling architecture and microarchitecture development together with programming models and compilation tools with good cross-layer coordination. One example is recent GPU products, which were originally targeted for the computer graphics domain. With the availability of CUDA and OpenCL programming models and associated compilation tools, as well as the efforts by Nvidia and other companies to build a user community around such programming models, the use of GPUs has spread into many other domains. Many of the top-100 supercomputers use GPUs for high-performance computation. In comparison, the Cell processor is less successful in terms of being accepted by other applications beyond its initial intended use (support of computer gaming); this is due to the lack of efficient programming models and a well-trained user community.

Cross-layer Power Optimization Challenges

In order to further cross-layer power optimization at the circuit/microarchitecture level and beyond, we need to overcome a number of challenges. Here, we highlight a few major challenges.

Models and Tools for Effective Energy-Based Design-Space Exploration

One of the biggest challenges in evaluating the energy efficiency of system research ideas is the lack of a fully integrated methodology to enable accurate energy-based design-space exploration at various levels of abstraction. Traditionally, researchers working on micro-architectural techniques use one of two approaches, depending on whether they are approaching the problem from an architecture perspective or a VLSI perspective. From the architecture perspective, one augments a standard cycle-level simulator with first-order event-based energy estimates and then possibly generalizes these estimates into an even more abstract system-level model. From the VLSI perspective, one uses either RTL or gate-level bit-accurate energy estimates based on abstracting detailed circuit-level models.

There are two Challenges to be addressed to support effective energy-based design space exploration: (1) Development of first-order, event-based energy modeling tools in evaluating revolutionary architectures that differ significantly from the architectures used to validate the original models; and (2) Tight (and preferably automated) vertical integration of the higher-level, first-order, event-based energy modeling tools with the lower-level, gate-level, bit-accurate energy modeling tools.

Modeling revolutionary architectures

There are many interesting research directions to take when exploring revolutionary architectures; these might include: custom accelerators, reconfigurable logic, near threshold operation, and emerging storage/interconnect technologies. It is not clear how well traditional first-order event-based energy modeling tools capture the energy

implications of these revolutionary architectures, since these traditional tools were validated against traditional architectures. Revolutionary architectures might include components not modeled in traditional modeling tools (e.g. reconfigurable logic, application-specific functional units, advanced vector units), and these architectures might not adhere to traditional assumptions about which portions of the system have negligible energy consumption (e.g. unstructured control logic, local/semi-global wires.)

Vertical integration

There is no well-established and integrated methodology for connecting the gap between first-order event-based energy models and gate-level bit-accurate energy models. Currently, researchers must either "abuse" the available first-order event-based energy modeling tools, or manually write and characterize a specific RTL implementation. Ideally, it would be possible to automatically generate first-order event-based energy models (or even higher system-level models) from gate-level bit-accurate energy estimates based on the RTL of new architectures.

Efficient Simulation and Architecture Exploration Support for Heterogeneous Architectures and Emerging Technologies

Architecture simulation research has a long history, but by no means is a solved problem. Cycle-accurate simulation of multi-core processors takes days to weeks, as the number of cores increases. The previous simulation speedup techniques—such as statistical sampling and hardware emulation—while very successful for single-core processors, are not easily extended to multi-core, parallel simulation due to the complications of inter-core communication, synchronization, and modeling of shared resources (such as L2 cache and network-on-chips.) Moreover, the introduction of heterogeneous components, such as small vs. big cores and accelerators, further complicates the simulation requirements

The opportunity for cross-layer design and optimization, for example, considering the use of sub-threshold circuits, bio-inspired components, will definitely further complicate the simulation models. In particular, technology scaling and related variability induced issues, as well as low-voltage solutions for energy efficiency, will need an increasing cross-layer optimization and management focus jointly for power and reliability in the coming years. Incorporation of control systems spanning circuit, micro-architecture and software layers would be integral to power-reliability management solutions addressing these issues.

Today's simulation infrastructures that model power and performance lack reliability modeling while ignoring the existence of control systems that can dynamically alter operational conditions affecting quantities of interest. Multi-timescale operation of these control systems, as well as affected characteristics (thermal - large time constant, timing failure - small), also increase the modeling and simulation challenges. Even the question of whether a single integrated modeling/analysis approach or some hierarchical modeling/analysis approach is better is unclear.

Akin to SCM for memory applications, RF and optics on chip for communication are interesting candidates for embedded and server platforms for both performance and efficiency reasons. Plenty of cross-layer design and management opportunities exist across technology, circuits and microarchitecture for these options. Modeling methodologies that

analyze the implications of adopting a particular technology characteristic with a certain circuit implementation adopted for a particular microarchitecture design are mostly manual today. A generalized framework for quick exploration of such multi-layer design options will be needed for accelerated adoption of newer technologies.

Finally, the design space exploration is a huge problem. There are many design variables associated with cores, accelerators, on-chip and off-chip memory systems, and on-chip and off-chip interconnects. The number of valid design points is easily in the orders of billions and trillions. Moreover, the design space is highly discrete, and usually non-convex. How to search effectively in such a design space efficiently for an optimal solution is extremely challenging.

Reliability

As technology scales beyond 22nm, reliability of CMOS devices is decreasing. This has traditionally been overcome with guard bands, where conceptually the paradigm is to *mask* the imperfection of devices and continue creating the illusion of perfection at the circuit-level and higher. While this idea of masking imperfection was well suited for previous technology nodes, as reliability gets increasingly worse, the paradigm of masking errors is causing enormous overheads at the circuit level and below. There are tremendous benefits in simplifying the design process and costs if reliability can be relaxed. Furthermore, reducing the requirements of correctness can provide energy benefits because circuits can operate at common case operating points and reduce margins for guard band, etc. The implication then for the higher layers is that the hardware is not always correct. This introduces a fundamental *cross-layer problem* on how to expose the hardware's lack of correctness up through to the software. It includes exposing reliability in at least three ways: i) building well-understood fault models that correlate with physical phenomenon; ii) building circuit and microarchitecture models that build upon the fault models to expose mechanisms to the ISA; and iii) building language models that programmers can leverage.

Ultimately there are several possible scenarios. Device reliability may get to be such a large problem that program energy efficiency will be very low unless programmers use the language hooks to expose places where less reliability is sufficient. For example, multimedia processing is well known to be error tolerant. Another scenario is that programmers may use the reliability hooks and exploit this as an optimization for energy tradeoffs—i.e. intelligent structure and exploitation of program error tolerance to obtain performance or energy improvements. In particular, in mobile environments or cloud environments, such reliability requirement modeling could be easily exposed to end users, and be embedded into standard frameworks like MapReduce or the Android SDK, etc.

These language extensions will then have to translate through compiler optimizations and exploit the microarchitecture mechanisms. Simple examples include aggressive voltage scaling, tolerating aging failures, etc. Micro-architectural and circuit techniques could exploit the relaxed reliability requirement to build highly efficient data paths that are unreliable. This could also enable and synergistically cooperate with techniques like analog computing. Also, fundamental architectures and organizations are made possible by revisiting the design of microprocessors if we assume errors are allowed. The research thrust is fraught with the inherent challenge in determining which portions (or which

applications) cannot tolerate errors and the best way to guarantee one's choice. Some promising research has begun to appear in this direction. They can be categorized as quantifying error tolerance, device-focused techniques, architecture-focused techniques to expose reliability, compiler-focused techniques to expose reliability, and language-focused techniques to expose reliability. While a vibrant area of research, much work needs to be done to understand interaction with circuits and technology scaling.

Cross-layer Power Optimization Opportunities

Despite significant challenges, we see great opportunities ahead in cross-layer power optimization to achieve an order-of-magnitude energy efficiency improvement through circuit and microarchitecture innovation coupled with software optimization. Here, we highlight some major opportunities.

Heterogeneity

Heterogeneity means including cores of various computing capability and extensive use of accelerators. Digital CMOS will still be the main implementation technology, but some of the accelerators can be analog/mixed-signal modules (such as bio-inspired components), and/or post-CMOS devices.

Architectural (e.g. accelerators) and technological heterogeneity (such as 3D integration of disparate technologies and nodes, nanophotonics, non-volatile memories) provide a multitude of performance and power improvement opportunities. Power efficiency advantages of heterogeneous systems were demonstrated by a number of studies. Accelerator-rich architectures provide significant performance and energy improvement for domain-specific computing. 3D integration has been shown to provide high-bandwidth low-latency interconnect, increased packaging density, and heterogeneous integration capabilities with resulting power efficiency advantages at the architecture/system levels. Similarly non-volatile memories have been shown to provide significant static power reduction and ways to transform various levels of the memory hierarchy. However, the increased complexity of such systems, as well as the lack of modeling and tool infrastructures, limits the effectiveness of microarchitecture techniques (and exploration of architectural trade-offs.)

While the energy efficiency of heterogeneous architectures that leverage accelerators has been largely acknowledged by both academic and industry studies, traditional power management techniques are not effective in such architectures. New cross-layer approaches that leverage the capabilities of both system software and hardware are needed.

Tools and models that specifically target heterogeneous architecture and heterogeneous technology designs are essential in order to guide microarchitecture decisions. As the specific challenges (such as temperature sensitivity, variability, reliability) vary significantly across emerging technologies, flexible tool infrastructures will be key in providing the customization and right levels of abstraction and speed/accuracy/complexity trade-offs, as well as the interactions among the heterogeneous components.

Recent studies indicate that the potential performance and power improvement associated with the emerging technologies and heterogeneous architectures are significantly reduced without careful design planning. This clearly highlights the need for modeling and tool infrastructures for effective adaptation of such systems.

The post-CMOS era, and the adoption of non-silicon technologies (such as graphene, carbon nanotubes), along with silicon, will potentially bring disruptive changes in the microarchitecture design as well as the associated modeling/tool infrastructures. While such disruptive changes are still in the long-term outlook, forward-looking infrastructure development and exploration are key in making the adoption less challenging.

Specialization

Specialization is a well-known technique that often results in a two to three orders-of-magnitude improvement in energy efficiency over a general-purpose system. These energy benefits are realized by optimizing data movement by directly passing operands from one hardware unit to the next rather than writing to memory structures, and eliminating instruction processing overheads that are present in a general- purpose system. To harness the large improvements in energy efficiency that are possible with custom hardware, embedded system and SoC designers have been using fixed-function hardware blocks in their systems for decades. Embracing specialization in a general-purpose computing environment, however, requires broadly increasing the scope of applications that can be supported via customization. In particular, one needs to consider efficient on-chip architecture and software support for accelerator scheduling, sharing, and virtualization.

One attractive way of improving the generality of customized hardware is through reconfigurable computing. Reconfigurable hardware platforms such as FPGAs and CPLDs allow the hardware designer to customize the system's functionality after fabrication, and are more cost-effective than ASICs at low volume. FPGAs are not a panacea, however, as they suffer from significant performance and energy overheads due to the reconfigurable interconnect that ties together configurable look-up tables. One promising avenue to address these shortcomings is to leverage emerging resistive memory technologies such as phase change memory (PCM) or spin-torque transfer magneto-resistive RAM (STT-MRAM.) Resistive memories hold the potential to replace the SRAM-based lookup tables and configuration registers in existing FPGA chips; the significant density advantage of these novel technologies over SRAM should translate directly to shorter interconnects, and thus lower interconnect power than current-generation reconfigurable systems. Other interconnect technologies, such as RF or optical interconnects, are other research directions with significant potential for making FPGAs more energy efficient. Another direction is to consider programmable fabrics of different levels of granularity—in particular, developing new programmable fabrics that are more compute-friendly. Finally, compilation time for reconfigurable computing needs to be greatly shortened. The time for logic synthesis and physical design of a modern FPGA can be tens of hours, far exceeding the compilation time for general-purpose CPUs or GPUs. Substantial effort and investment need to be made to achieve two or more orders of reduction of compilation time to make reconfigurable computing practical for wider adoption.

Further Advances on Standard Interface

The cross-layer power optimization and management framework is, in general, a way for a lower-level layer to provide a feature that enables the control knobs, and an upper layer to utilize the control knobs. This approach can achieve a power saving closer to the optimal than within-layer power optimization because a higher layer has more information from the applications and/or users and is aware of more accurate information about the system idleness. In view of systems theory (cascaded control), the lower layer corresponds to an inner control loop that ensures stability, and the upper layer corresponds to an outer control loop that determines the set point (supervisory control.) The interface between the lower layer and the upper layer must guarantee stability of the system as well as the maximum effectiveness of power management. An appropriate level of abstraction is the key to the success of cross-layer power optimization and management.

Transmeta pioneered DVFS for microprocessors based on its code morphing software (CMS.) The CMS idea was successful, and is also currently widely used. However, they tried to apply DVFS within the CMS framework, but their method does not fully consider the upper-layer software architecture and behavior when DVFS is applied. The CMS framework does not provide enough long-term software idleness information, which significantly reduces potential gain from DVFS. Consequently, DVFS results in more performance loss compared to energy gain. The inner-layer DVFS is still used today. It may achieve power gain but may have negative energy gain, which contradicts the DVFS philosophy.

A proper interface specification between layers is crucial for cross-layer power optimization. A well-defined standard interface between the microarchitecture and its upper-layer interface can be fully aware of the functionality and limitation of the power management feature provided by the microarchitecture. For a given workload and a deadline, a hypervisor should be able to determine the clock frequency of the CPU and memory and voltage level of the CPU, for example. Companies typically have a proprietary interface between the microarchitecture and hypervisor. However, a systematic implementation requires an industry standard description of the control register (for example) to read and modify the system states, power consumption, die temperature, and so on, that ensures safe system operation as well as maximum power saving.

Most of all, an industry standard between the microarchitecture and hypervisor may provide a great change for application programs running on mobile platforms, which need more aggressive cross-layer optimization. As mobile platform applications are often based on real-time image and audio processing and communication, such a standard interface enables the optimal DVFS with exact knowledge of the system idleness. With a well-designed frame-rate control of the built-in camera, DVFS of the CPU and acceleration hardware, memory power management, and radio power management/control, the standard interface may achieve a 2-10x additional power saving for Android applications.

Memory Power

Off-chip memory power (DRAM) is an increasing fraction of system power and increasing limitation to computing capability growth. Many core approaches to processor performance growth further exacerbate this issue. A promising solution is the integration of large amounts of DRAM (or SCM looking to the more distant future) onto the same package, leveraging silicon interposer technology or 3D stacking obtaining lower-power,

and high bandwidth connectivity between processors and memory. A cross-stack design effort is necessary across packaging, circuits (e.g. wiring across 3D, power delivery), microarchitecture (e.g. changes to memory array designs suitable for cache-line and sub-cache-line access rather than page-level access, techniques to address wear-out, write-current issues for SCM) to reach a feasible design point and, potentially, runtime and application-level design changes to take better advantage of this increased integration. New power delivery, management and cooling techniques again spanning circuits, microarchitecture and software components are needed to optimize capabilities in this tightly integrated solution.

Co-optimization

A key opportunity for cross-layer power optimization and management is to use co-optimization at design time and at run time. In the sensor platform domain there are opportunities to come up with plug-and-play solutions that can use any of a number of various sources of energy (e.g. battery, energy harvesting, etc.), various computation engines that tradeoff performance for power, and diverse communication channels depending on availability—all while globally optimizing for desired overall metrics. In the mobile platform domain there are rich opportunities for device/cloud co-optimization. Sometimes, but not always, it is better to offload some computation to the cloud if that saves energy and/or increases performance on the mobile platform—even at the expense of extra energy in the cloud. Offloading to the cloud, however, may not be desirable under some conditions, e.g. when the savings on the mobile are minimal, or when the overhead on the cloud is too large. Yet even in that case, offloading might need to be made at run time when batteries get depleted. In the server platform domain there are significant opportunities for co-optimization of the traditional compute platform in conjunction with the cooling solution and the power delivery solution. For example, allowing for the operating temperature to get slightly warmer may allow for overall power and energy savings when the compute and cooling power are considered together. Some of the co-optimization opportunities include:

- Sensor: plug-and-play with global metric optimization
- Mobile: mobile/cloud co-optimization
- Server: compute/cooling/delivery co-optimization (e.g. higher temperature unregulated operation)

Exposing on-chip communication from the microarchitecture to the higher levels of the stack may also present an important opportunity for cross-layer optimization. But this idea needs to be further researched and validated.

Platform-Specific Optimization

Embedded system

Embedded systems are highly specialized circuits. Most of the embedded systems are designed for real-time applications such as sensing and controlling. The hardware and software are usually customized for specific applications. They provide very little programmability, which enables closely coupled hardware-software co-optimization at

design time. A design environment that enables seamless vertical integration will be critical and most efficient to achieve energy optimization.

From a system point of view, an embedded system is an integration of computation, storage and communication components, as well as energy storage (i.e. battery, super capacitor) and energy harvesting units. Microarchitecture designers should provide hardware channels to efficiently monitor and manage these devices during runtime. New abstractions and interfaces have to be investigated to coordinate the management of energy harvesting/storage units together with the management of energy consumption units (e.g. embedded systems.)

Some of the embedded systems, such as environment monitoring systems, are natural candidates for approximate computing. Techniques such as a relaxed guard band for improved power efficiency could be applied in these systems. Other embedded systems, for example embedded controllers of airplanes or automobiles, are mission-critical. They have a nearly zero tolerance of error. Reliability constrained energy optimization at design time should be investigated.

Mobile computing system

Mobile computing systems have a large amount of specialized circuits. They provide some degree of customization and reasonable support for programmability. The software running on a mobile computing system is a mixture of large amounts of application-specific computing, such as gaming, video/audio encoding/decoding, etc., with heavy user interactions. For typical mobile computing systems, reliability is not as critical as security. It is acceptable to have jitters in video playing as long as it is within the user's tolerance. However, access to a protected memory region, which would lead to a security breach, should definitely be prevented. The microarchitecture should allow the trade-off of reliability for energy efficiency up to certain degree, such that it satisfies the QoS and security requirement. Error detection and fault recovery mechanisms should be investigated.

Display and wireless interface are two major power-consuming components in a mobile system. While backlight dominates the power consumption of a traditional LCD, emerging organic light-emitting diode (OLED)-based displays do not require backlight and the RGB value of a pixel determines its power consumption. This architecture difference between OLED and LCD brings new opportunities in OLED display power optimization. Recent work has demonstrated the effectiveness of content-aware supply voltage scaling on OLED displays [1] and the effectiveness of color transformation [2]. This suggests new cross-layer power optimization opportunities in designing the control circuits and architecture of OLED displays, as well as making applications aware of their display energy cost through an OLED display energy model.

Another hardware trend of mobile systems is the use of multiple antennas for improved network speed and overall network capacity. For example, spatial multiplex multiple input multiple output (MIMO) technology, adopted by 802.11n and LTE, sends independent data streams through antennas to leverage spatial channel diversities for enhanced speed. In another example, recent research has suggested that a mobile device form a focused radiation pattern through digital beam forming in order to conserve transmission power

and improve network capacity. The use of multiple antennas along with their transceivers increases the circuit power consumption but has the potential to significantly reduce the radiation power consumption. The most energy-efficient number of antennas to use depends on the communication requirement—such as throughput and range as demonstrated by recent work [3]. This suggests an important cross-layer power optimization mechanism that power-manages the multi-antenna transceivers based on communication requirement.

Servers

Servers have the least amount of specialized circuits and the maximum programmability. They have the highest requirement for both reliability and security. Runtime cross-layer power optimization is critical to achieve the energy efficiency of servers.

It is agreed that we need to focus more on cross-layer channels to exchange power models and workload requirements between microarchitecture and upper-level applications. More accurate power models and highly standard constraint languages are believed to facilitate the cross-layer power management. New memory technologies such as NVM should be investigated. Heterogeneity, including cores of various computing capabilities and extensive use of accelerators, should be introduced to increase energy efficiency at this level. However, we need new techniques to efficiently and accurately estimate the benefits and the overhead of introducing heterogeneity. Compiler support should also be provided to utilize the heterogeneity.

Another dimension of cross-layer optimization at the server level is computing, cooling and delivering co-optimization. Chip-level interconnect and packaging technology advancements, combined with micro-architecture and circuit-level innovations, can turn this challenge into a tremendous opportunity.

Natural/Biological Systems

Understanding other natural/biological systems that achieve better energy efficiencies at non-deterministic tasks can serve as an inspiration for designing more efficient domain-specific accelerators. The primate brain is considered to be more efficient in processing several tasks such as visual processing, speech recognition and reasoning. For example, the energy-efficiency of a human brain is estimated to be 10^6 times better than the IBM supercomputer that competed in a quiz competition. What can be learned from the efficiencies in algorithmic, encoding, signaling, and implementation fabric used in a brain? What are the synergies across these different elements of the brain's operation that achieve these energy efficiencies? Can it help to identify similar cross-layer optimizations across hardware-software designs to achieve similar energy efficiencies? Can there be inspiration from the structure of the communication fabrics in the brain for designing on-chip communication fabrics? Can these principles influence the design of domain-specific accelerators? Accelerator implementations of algorithms based on brain-based visual models such as Convolutional Neural Networks and HMAX [4][5] demonstrate 10-20x improvements over execution on existing general-purpose fabrics such as GPUs and CPUs. These accelerator fabrics achieve energy efficiencies of 1 TeraOPs/Watt for visual processing [6]. How can these efficiencies be further enhanced using emerging device and circuit primitives? Early prototypes are emerging of dense crossbar arrays using non-

volatile memory technology to achieve low-power, low-leakage, dense fabrics for processing vision algorithms [7]. Energy-efficient microarchitectures for visual perception can help a variety of critical applications in health (aiding visually impaired) and homeland security (pervasive smart sensors.)

Area 3: Micro-architecture, Systems, and Beyond

Area Leader: Margaret Martonosi

Co-Editor: Sudhanva Gurumurthi

Other Area Members: Murali Annavaram, Rajeev Balasubramonian, Pradip Bose, Jeffrey Draper, Brucek Khailany, Hyesoon Kim, Rami Melhem, Trevor Mudge, Onur Mutlu, Mani Srivastava, Michael Taylor, Josep Torrellas

Rajeev Balasubramaniam, Pradip Bose, Sudhanva Gurumurthi, Brucek Khailany, Margaret Martonosi, Onur Mutlu, Trevor Mudge

There is a current and growing crisis in power management for a range of power-performance design points for mobile to enterprise computer systems. In this section, wr review examples of success and failure stories in cross-layer power optimization approaches followed by opportunities and possible approaches to address the micro-architectural and system level design challenges via cross-layer approaches.

Success Stories

The interface between hardware and software has seen significant attention for cross-layer power optimizations over the years. From a very expansive point of view, even things like general compiler optimizations and some cache hierarchy optimizations can be viewed as power optimizations, because many such performance-aimed techniques also improve power dissipation.

Focusing more narrowly on cross-layer power optimizations, the research community has proposed and evaluated many ideas, of which some have made it into real systems and others have seen less widespread adoption.

We discuss here two iconic examples of the power of cross-layer power optimization: a mobile scenario and one from enterprise-class systems. After describing the success and potential of such cross-layer approaches, we will follow with more detailed thoughts on why such successes are so limited, what research can do to help such successes be more widespread, and why we need to fund such research now.

Example 1: Mobile scenario:

Mobile phones are clearly an example of a power-constrained system, and the increasing computational demands on smartphones mean that increasingly-aggressive power optimization and management is needed across all hardware and software levels.

Example 2: BlueGene-Q

At the enterprise level, IBM's BlueGene-Q effort represents success with cross-layer power optimization for a high-end supercomputer. Blue-Gene-Q instances have dominated the Green500 list which enumerates the most energy-efficient supercomputers in the world.

One example of a cross-layer optimization was the design of the basic compute node: an SoC with embedded DRAM + 16+1 compute cores + network switch. Integrating the network switch onto the SoC allowed for high-performance at lower-power. High integration and SoC approaches have helped reduce power needs. In addition, using lower-voltage circuit design techniques (relative to the 45nm technology node) allowed for further savings.

While BlueGene-Q's hardware integration may be considered a power success, there are also aspects of the hardware-software interface where even more cross-layer optimization may have been possible. For example, while the hardware design was aggressive, the hardware-software connection was not particularly strong. The operating system is a ported version of Linux, but aggressive power optimization has not been included in it yet.

This sort of staged adoption of power optimization in operating system and applications is fairly common in both mobile and enterprise computing. In some ways, it points even more strongly to the need for appropriate cross-layer information flow and abstractions, so that power optimizations can be integrated into systems in a staged fashion.

Summary: For both the mobile and enterprise examples above, there are a few clear characteristics that allowed these cross-layer success stories to happen. A major enabler of these success stories was that the same corporate organization controlled all or most of the system design layers. This means that when more information was needed to flow between two layers, the decision about how or if to implement it did not need to involve two separate companies.

Another characteristic that is common to these two success stories is that the aggressive system goals required concerted power management innovations beyond what would have been possible without cross-layer attention. In the case of mobile phones, battery life and thermal issues so directly influence consumer experience that aggressive low-power goals are universal. As a result, the mobile space aggressively employs heterogeneity in processors, storage, and other system modules. At the enterprise level, scaling goals mean that very-large-scale very high-performance compute engines can ONLY be built if their power dissipation is minimized in all possible ways.

Opportunities and Possible Approaches to address the challenges via cross-layer approaches

Green Computing

Environmental pressures and government rating systems (e.g. EnergyStar) have resulted in initial experimental R&D in the area of "green" or "zero-emission" data centers (e.g. work done by IBM Zurich Research Lab, in collaboration with EPFL [8].) In such approaches, the heat dissipated from data centers is reused to serve as energy sources for city heating and water desalination projects. Recent research from University of Florida [9] has proposed the use of renewable energy sources (specifically solar power) to ease the power burden of servers. These ideas call for cross-layer optimization and modeling across energy supply, workload-driven demand and heat recycling. Significant new research investment is required to expedite R&D, with quick technology transfer in this crucial new emerging area of green computing.

Processing-Near-Memory

3D packaging is an emerging technology that facilitates close integration of processor units and memory units. Thus, there is an opportunity to access DRAM main memory without traversing expensive off-chip interconnects. This helps overcome the power wall associated with achieving high bandwidth access to memory. Some emerging applications are excellent fits for processing-near-memory abstractions. For example, MapReduce applications are embarrassingly parallel during the Map phase and can be effectively mapped to a processing-near-memory architecture. To exploit such architectures, we need new programming models to facilitate application development, new compiler strategies for the low-power cores on a 3D stack, and runtime systems that can manage co-ordination between many threads. Furthermore, the tradeoffs between capacity and bandwidth at different level of the memory hierarchy could change significantly, and will require cross-layer optimization between architectural, programming systems, and application layers.

Tools

Writing parallel programming has been one of the major challenges in programming. Parallel programming is fundamentally hard, but we could also argue that the lack of tools to help programs is another cause. Assisting parallel programming is important to both programmer productivity, and program efficiencies in this heterogeneous multicore era. Hence, developing software tools that can help programmers and/or other software systems is pressing needs. For example, if a profiling tool can identify energy inefficiencies of computations, programmers or software system can optimize and tune the identified section. Tools to provide which algorithms might be more efficient in the underlying hardware can also improve the quality of code.

We also need more energy and performance monitoring tools. Even though many hardware performance counters are available, not many easy programs/tools that provide energy and performance information to programmers exit. This kind of feedback information can be also directly connected with dynamic compilation system to optimize programs more efficiently.

Criticality

Not all work is equally important and critical. Programmers or compilers can easily know critical tasks from non-critical ones. Knowing criticality can provide many opportunities in various layers. For example, non-critical work can be easily sent to low-performance but high energy efficient cores/memories. We can also control DVFS to improve energy efficiencies for non-critical work. Another example can be found from UI threads vs. computation threads. OS can expose this information or priority to the application level.

Unfortunately, right now there is no good way for hardware to get this criticality information from upper layers. There should be some generic ways of transferring this information from software systems to the underlying hardware.

New Abstraction Layers

ISA has been a great way of hiding the complexity of hardware or software but providing a concrete and stable interface between two layers. Consequently, quite amount of information from software is lost on the way. Examples are criticality, data-flow information, data locality, data movements etc. As a result, hardware tries to regenerate information using only transistors such as finding critical threads, critical data etc. Hence, this might be the time for us to re-think about the granularity of ISAs. If software can pass the characteristics of software, the hardware can use that information to improve performance and power. Then, what will be the right way of passing the information? What should be the right granularity?

Example Topic Areas

This section begins by describing a set of three key topic areas that will be instrumental to supporting cross-layer power management and the high-leverage power savings it can lead to.

Modeling, formal specifications, and abstraction layers

Lower levels of abstraction in the design hierarchy have better coupling between layers, because their specification is based on established models, for example: physical circuit models, Boolean algebra, register-transfer languages, etc. One of the things that inhibit cross-layer optimizations from the system level to the (micro)architectural layer and downwards is the lack of a formal specification language. This area is in its infancy in terms of experimental research or commercial systems like: BlueSpec, SystemC, SystemVerlilog, etc. These specification languages are still at a level of abstraction that is too low level for architects to convey to and receive information from levels below the system abstraction layer. Significantly more investment is needed to bridge the gap between the system architecture level and the physical implementation layers.

Examples of the capabilities we envision as a result of inventing such an executable specification layer for system architecture are:

- Linkage between IPC and cycle time aspects of system performance, which would allow architects to work with logic and circuit designers in co-optimization mode with the goal of maximizing *net* power-performance efficiency metrics (e.g. energy-delay.)
- Automatic creation of cycle-accurate and latch-accurate architecture-level simulators, that facilitate cross-validation across the architecture and RTL layer models of the machine.
- Specification and modeling of embedded sense-and-actuate power/thermal control loops that allow workload-driven benefit analysis with accurate physical energy-related metrics included in the overall analysis; also this allows the architecture-logic-circuit cross-layer team to make more optimal choices regarding the timing and extent of applied power control actuation knobs.

In addition to design-time specifications across the architectural boundaries, there is also a need for higher abstraction layers to assist in system-level mapping and scheduling choices, particularly when dynamic and heterogeneous parallelism are involved. We argue for System-Level Instruction Set Architectures that allow coarser-grained chunks of

computation to be managed and scheduled. By avoiding per-instruction handling, energy overheads can be greatly reduced, and coarser-granularities also assist longer-term planning of communication and energy planning across large multi-core chips.

Accounting for and Minimizing Communication Distances

The cost of communication is now significantly higher than the cost of computation. A single 64-bit double-precision, floating-pint multiply-accumulate operation only consumes 50 pJ. Fetching a 256-bit value from a cache bank that is 1 mm away costs 31 pJ. Fetching the same value from a distant cache bank costs 1.2 nJ. Thus, there is an order of magnitude difference in communication energy depending on where data is found on the chip.

Similarly, main memory organizations are also highly-distributed. Data may be found in a memory DIMM on a local channel, or may require multiple hops through inter-socket interconnects (e.g. Intel QPI), or buffer chips (e.g. Intel SMB), or off-board interconnects (if data is allocated in a memory blade.) The average cost of communication will again vary by an order of magnitude depending on the quality of data placement in main memory. Currently, higher levels of the system stack (application, OS, compiler) are largely unaware of data placement in caches and memory modules. This missing link leads to an order of magnitude increase in communication energy on average.

Communication distance can be minimized by not only placing data in appropriate regions of the memory space, but by also moving computations to the memory when appropriate. Such computation migration requires extensive support from the operating system and programming model, artifacts that do not currently exist.

Supporting heterogeneity

Power optimization needs have driven the system architecture community towards heterogeneous multi-core designs, with embedded accelerator "sub-cores" connected via heterogeneous interconnect elements. However the programming model and software task scheduling support still lack the formalism required to exploit the full potential of this emerging paradigm shift. Significant new research investment is needed to facilitate co-design and co-optimization across the application, system software and architecture layers. One potential solution would employ a menu or library of different computing units (i.e. cores and accelerators) that can be deployed statically (during design) and invoked dynamically during power management. Again, progress towards an executable specification language (as mentioned before), with suitable extensions to cater to the multi-core abstraction layer, will enhance our ability to solve the current impediments to exploiting the paradigm of pervasive heterogeneity.

Why now?

In the last decade, successive generations of semiconductor technologies have deviated more and more from classical scaling, resulting in unsustainable levels of energy density. Perhaps the most recognizable contributing factor of this condition is the slowing pace of voltage scaling. In computer platforms ranging from the sensing, portable, and server domains, power has become the most important constraint. As a result, researchers and developers in the hardware and computer architecture areas have engaged in a relentless

effort to optimize designs for power and energy efficiency. It is a fact that major advances have been made in the last few years, ranging from highly power-efficient semiconductor devices and circuits, microarchitecture structures, on-chip architectural organizations, and integrated systems. Each layer is using all the tools, techniques, and approaches it has available to provide the most power-efficient solution to the other layers of the computing stack.

It is crucial to continue to make progress in the area of power efficiency. Two clear drivers are the need for exascale-level computing and the need to reduce the carbon footprint of computing. As noted in the DARPA–sponsored report, ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems, timely advances in extreme scale computing cannot be achieved without a holistic approach to reducing energy. Even with aggressive liberal assumptions beyond straight-line ITRS projections, an exascale system in the 2018 timeframe is projected to dissipate 67 MW, far beyond the tolerable budgets of interested organizations. Coupled with resilience issues that are prevalent at such system scales due to the laws of reliability concerning such significant numbers of components, overcoming the hurdles to achieving exascale computing will require a cross-layer system approach with regard to several factors, energy being the predominant one but certainly not the only one. In some sense, this challenge has already been acknowledged in the execution model research that has been undertaken in the exascale community, but the research is very preliminary.

Additionally there are societal factors that are becoming increasingly more important. The advent of "green living" has brought the notion of carbon footprints to the forefront of most aspects of life, and computing is no exception. The amount of power dissipated by office PCs and data centers is often highlighted in news articles on the environment. With the sheer volume of electronics in computing platforms from handheld mobile devices to large-scale data centers, significant energy reduction in global computation can result simply by addressing cross-level power optimization for a few key platform types.

Unfortunately, local optimization within individual layers is not enough. There is only so much that each individual layer can do to improve. We are approaching rapidly-diminishing returns in this area. It is, therefore, necessary to optimize across the device, circuits, microarchitecture, architecture and software systems layers.

Luckily, such cross-layer optimization offers major opportunities for power savings. At a high level, this is because, classical designs, by focusing on optimizing each layer individually, end up with a globally sub-optimal design. What is needed is an approach that provides the correct interfaces across layers, exporting the needed information to other layers and taking in information from other layers. The result is that, while individual layers are sub-optimal, the whole multi-layer design is optimal.

Note that, currently, we are addressing the power constraint problem with parallelism and heterogeneity. Both approaches already require breaking the traditional layers and optimize across them. Consequently, we are already starting to make progress in this area. What is needed, however, is an aggressive agenda for cross-layer optimization.

In this report, we present an agenda for cross-layer power optimization. We start by giving a few examples and then outline the areas that need to be researched.

Quantitative Targets

We use data from the HPC space in our quantitative sketch here. In a mobile, sensor, or datacenter type environment, the exact numbers will be different, however the themes will be similar.

An aggressive estimate of the quantitative opportunity at the micro-architectural and systems layer for power optimization is 3-5x from energy efficient chip micro-architecture and 2-4x from efficient system design (6-20x total.)

At the micro-architectural level, our opportunity is to eliminate per-instruction energy overheads. In 40 nm technology running at nominal Voltage (0.9 V), the energy of executing a double-precision floating-point multiply-add (DFMA) is 50 pJ. However, the energy of executing a single instruction on a high-performance CPU in similar technology is 1.7 nJ, meaning the actual computation is only 5% of the per-instruction energy and for each instruction, scheduling and data movement provides a 20x tax on the actual computation. An energy-efficient throughput-optimized core in similar technology is 450 pJ per DFMA or 225 per floating-point operation, 7.5x more energy efficient, but still a 9x tax on the actual computation energy. Our opportunity at the micro-architectural level is to attack this 9x tax and build more energy-efficient processors. We estimate that by removing these per-instruction energy overheads, we could improve energy efficiency by around 3-5x compared to these current heterogeneous systems, higher if compared to CPU-only systems. Additional energy efficiency gains beyond this factor of 3-5x are possible by exploiting fixed-function hardware specialization.

For large-scale systems, at the system layer, we can improve energy efficiency by removing overheads and providing energy-proportionality. The systems layer today has significant energy overheads above and beyond what is provided by micro-architectural components. For example, while GPU-based supercomputers in the Top 500 contain energy-efficient GPUs at 225 pJ/flop (4.44 GFLOPS/Watt), full systems only run around 1 GFLOPS/Watt. This >4x system power overhead can be attributed to the lack of energy-proportionality throughout all components in the system, power-inefficient heterogeneous system architecture (discrete CPUs and GPUs), power dissipated in interconnect between compute nodes, and other overheads associated with power delivery and cooling. Although the exact system overhead will vary from system to system (e.g. mobile processors typically have lower overheads), this >4x system power overhead is exemplary of a problem that arises from not treating power as a cross-layer problem. Integrated systems which have taken a more cross-layer approach (BlueGene, Anton, etc.) have lower system power overheads.

Cross-Layer Examples

Example #1: Low-Power Design With Emerging Non-Volatile Memories

Emerging Non-Volatile Memories (NVMs), such as PCM and STT-RAM, offer a clear opportunity for cross-layer optimization. At the device level, key figures of merit include the retention time, endurance, energy required for reads and writes, and access latencies and asymmetry between the read and write latencies. Based on the level in the memory

hierarchy (caches, main memory, or storage), the device properties will need to be adapted. For example, an NVM for a storage application will require long retention times, perhaps at the cost of access latency, whereas an NVM for use in cache or memory requires low access times but can sacrifice retention time (as demonstrated by [10].) Managing the physical limitations of the NVM (e.g. limited write endurance) will require optimizations at the circuit and microarchitectural levels. These include the use of techniques such as coding, sparing, and wear leveling, which have to be designed in a power-conscious manner.

NVMs can also influence cache management and memory bandwidth management policies. For example, one may choose to use a cache management policy that minimizes writes rather than one that minimizes the total number of accesses. The use of NVMs in layers of the memory hierarchy above storage provides a new property – non-volatility - that can be exploited by the microarchitecture and software. For example, non-volatility can facilitate an "instant power on" capability that can be used for fine-grained power management, either at the microarchitectural level or through software control.

Non-volatility can also lead to a blurring of the distinctions between memory and storage and consequently influence the design of file systems, databases, and virtual machines, as recent work in this area (e.g. [11][12]) have demonstrated. For example, in a hybrid memory system that uses a combination of volatile memory and NVM, the file system can manage the placement of data objects based on their access and update frequencies or data longevity requirements (e.g. placing temporary files in volatile memory vs. allocating NVM space for persistent data or logs.)

Example #2: Designing with unreliable silicon

As a second driving example of the potential of cross-layer optimization, consider scenarios and implications around the possibility of designing with unreliable silicon. Looking forward, upcoming technology generations may see less reliable transistors. In some cases, silicon reliability will tradeoff fairly directly against power dissipation. For example, higher supply voltage levels or increased device or circuit-level redundancy can often be used to overcome unreliability in individual transistors, but at some power cost. In general, such scenarios may reduce guardband requirements.

With underlying silicon unreliability as a possible technology trend, this case study explores how exposing reliability to the higher levels of the stack may result in power-saving leverage. At the same time, it often requires cross-layer approaches for percolating characterizations of device behavior up the stack, and information about software/application needs down the stack.

At the circuit level, device unreliability may call for increased redundancy in circuit designs, or different strategies for encoding data. For data storage, parity and ECC strategies can be tuned based on reliability expectations. For both data storage and computation, reliable and unreliable versions of modules might be offered, with higher-level architectural or compiler approaches selecting which to use based on software needs. In on-chip interconnect, transmission unreliability can be tolerated through protocols with checksums, retransmissions and other strategies, including many drawn from other larger-scale networks.

Percolating aspects of device unreliability to the microarchitectural level offers further leverage. As previously mentioned, we envision that in addition to "fully-reliable" units, one could have less-reliable versions that are offered at lower power cost for error-tolerant portions of a computation. For example, video and streaming applications may have some tolerance for errors in their data calculations, as long as errors do not exist or percolate into memory address pointers, program counters or other non-error-tolerant portions of the code. (Language support for such approaches is discussed next.) Another microarchitectural approach to underlying unreliability might be techniques to launch several copies of a computation down the pipeline and allow for "voting" to gauge result correctness. When reliability is most important, many redundant copies will be used, albeit at a higher power cost. When power matters more than 100% calculation correctness, fewer redundant copies can be used.

At the Architectural and language level, further adjustments can be made to exploit this power vs. reliability tradeoff. Following on the example from the preceding paragraph, one will probably want to "mark" aspects of error tolerance at a coarser granularity than individual instructions. For example, in an image processing code, the programmer may be able to express a desired SNR or quality of result metric for an overall calculation more easily than they can express reliability requirements for individual lines of code. This can be compiled down to reliable or unreliable threads, or mapping onto different ALU units accordingly.

At the Language and compiler level, the programmer may benefit from programming environments that are less general than C or C++, in ways that preclude the arbitrary mixing of data and control. For example, while it is often tolerable to have a bitflip at an arbitrary point in some image data, the same program cannot tolerate a bitflip in an address pointer. One could imagine hardware implementations where program counters and address pointers are stored in "hardened" or higher-reliability circuits while image data is stored in less reliable circuits that save power.

In summary, the overall purpose of this example has been to articulate how exposing aspects of a power tradeoff from the lowest system levels---namely power/reliability tradeoffs in future silicon transistors---may lead to power-saving leverage and tradeoffs at several other system layers up to and including software.

Example #3: Energy-Reducing Cores

As larger fractions of a chip's transistors become dark transistors, power consumption becomes the dominant limiting resource for chip design and operation, more so than other resources such as total silicon area. This shift often calls for new architectural techniques that "spend" area to "buy" energy efficiency. One approach is to use this dark silicon to implement a host of heterogeneous coprocessors, each of which is attuned to the computation.

In a future heterogeneous-focused system, execution hops among these coprocessors, executing where it is most efficient. At the same time, the unused coprocessors are power- and clock- gated to keep them from consuming precious energy. If the specialized coprocessors collectively cover a large fraction of the workload, then the energy efficiency of the system will rise significantly. Alternatively, if the coprocessors are able to cover large

fractions of individual applications, we can enable new functionalities for the platform. As the amount of dark silicon area increases, chip designers can afford to include more coprocessors, each one specializing to an even greater degree to cover a smaller fraction of the workload even more efficiently.

Heterogeneous processors are already prevalent in mobile application processors in cell phones and tablets already contain many specialized accelerators for base-band processing, graphics, computer vision, and media coding. However, the performance-first orientation of accelerators limits their applicability greatly. In order to attain their central goal of improving performance, they typically target computations that are highly parallel, execute tight loops, have relatively regular memory accesses, and are relatively easy to parallelize into hardware. This focus on performance and thus parallel execution makes these accelerators difficult to construct because the problem inherits a similar set of problems as parallelizing compilers -- such as pointer analysis, and code restructuring. This limits the class of programs for which accelerators can be built, and also makes the process very labor-intensive in the typical case.

An alternative, power-first approach looks at the problem differently. Typically processors impart a 10-100x overhead over the underlying ALU and memory accesses. If we abandon our focus on increasing performance, but rather seek to maintain performance but reduce energy we might be able to map a much larger class of computation into specialized hardware, and we could extend our targeted computations to include serial computations in addition to parallel computations. Because this mapping would not require the use of parallelizing compiler technology, it could cover a much wider class of code and be more susceptible to automation. The potential for energy savings from eliminating the overheads of instruction interpretation -- 10-100x -- are highly compelling, and potentially, large portions of common applications could be targeted -- from the Web Browser to even the Operating System.

Of course, there are a large number of challenges in this class of system, which drives a need for cross-layer optimization. We need ways of hiding the complexity of these cores from the programmer; we need ways for the OS to manage these cores; we need tools that can scalably facilitate identifying and generating cores that would be useful to have; we need new memory systems that can be scalable and/or coherence; we need ways of ensuring compatibility; and also retiring cores that target software this obsolete. The system would need power-gating technology that has ultra-leakage -- perhaps through NEMS or tunneling transistors; and we would need mechanisms that allow specialized hardware to stay relevant even as the software changes.

Over all, the drive for energy efficiency, and the rise of dark silicon calls for a new perspective on computations and results in new novel designs. These in turn require us to reconceptualize the entire computation stack in an energy-first way to arrive at new design points that are fundamentally more effective than today's approaches.

Interfaces that facilitate effective cross-layer optimizations

Existing interfaces between layers (e.g. the Hardware/Software interface or the System Software/Hardware interface or the Application/Compiler interface) are rigid and

relatively non-expressive. They do not allow enough information to be passed across layers to optimize for power. Since each layer is designed relatively independently of the underlying or overlying layers, significant inefficiency ensues. First, design in each layer is usually over-provisioned to maximize a local optimization metric that may not necessarily be beneficial for overall system efficiency, e.g. bandwidth or latency, rather than a much more meaningful global metric like user-experience, total cost of ownership, or end-to-end performance and Quality-of-Service (QoS.) Second, usually significant effort and complexity is spent in each layer to rediscover information that could have easily been obtained from another layer. Third, since layers do not coordinate well with each other due to a lack of information, the decisions made in different layers sometimes contradict each other, leading to an overall suboptimal and inefficient design. For example, since OS-level priorities of threads are not conveyed to hardware in existing multi-core systems, the memory controller hardware can unfairly prioritize lower-priority applications over others (because it is optimized to maximize a local metric, main memory bandwidth), leading to denial of memory service and significant slowdown of high-importance applications (see [13]) and eventually large user discomfort and energy waste for the entire system. All three of these lead to large waste in power. Finally, significant opportunity for optimization is missed in each layer because information needed to do some optimizations is missing and was not communicated by other layers. This again leads to energy waste and limits the optimizations one can do both cross-layer and within-layer.

It is clear that a rethinking of the interfaces across layers is necessary to enable effective and efficient cross-layer information communication that would lead to global optimization of power across the layers. In fact, such a rethinking is necessary and useful not only for power optimization purposes, but can also lead to significant optimizations and gains in performance, complexity, reliability, and potentially other optimization criteria. Ideally, we would like to design new interfaces that can overcome these aforementioned critical shortcomings without sacrificing the benefits provided by simple interfaces (i.e. design productivity.)

We believe there are several key research questions to address to devise effective and efficient interfaces for cross-layer optimization:

1. What "unifying, cross-cutting principles" should drive the design of the interfaces? These are principles that span across layers to coordinate information flow and policies across layers and to guide/ease the co-design and management of resources within and across layers?
2. At what granularity should the information be communicated between different layers, especially the HW/SW interface?
3. How can we design the interfaces such that designers at each layer can reason about the global implications of the optimizations they do at each local layer? How can we design the interfaces such that the designers in each layer can be productive?
4. What information needs to be passed across layers to enable new global power optimizations?
5. How can we keep the interfaces simple and efficient while at the same time making them much more powerful and expressive?

6. What kind of new interfaces are useful to manage computation and communication in heterogeneous systems that employ heterogeneous computation, communication, and memory components?
7. Do we have a single interface that spans all use cases (e.g. mobile, server, desktop, sensor, embedded) or do we design different interfaces for different use cases? Can we devise general interfaces that can be adapted to all use cases?

Fortunately, several promising research directions and ideas exist:

1. "Criticality" as a unifying principle to design new interfaces.

One key principle that spans across all layers to optimize power and performance is the notion of criticality. A system can be made significantly more efficient (both high performance and low power) if it can 1) dynamically identify critical "tasks" and spend most of the power/energy to perform these tasks and 2) spend minimal power/energy on all non-critical tasks. Here, what we mean by a task is any unit of work that needs to be performed; this could be an instruction, a memory request, a function, a thread, a collection of threads, an entire application, a collection of applications, etc. The principles outlined to improve energy efficiency by taking advantage of criticality could be fundamental across these different entities, which we will refer to as tasks.

Examples of exploiting criticality to improve performance and efficiency abound, yet these examples are usually specific to a layer or no principled way of communicating criticality with an efficient interface to enable large cross-layer optimizations exists. Not all tasks are equal - spend as much energy on a task as it really needs: Not all tasks that need to be done are equally important. For example, not all instructions are equally critical to system performance [14]. Not all network packets or memory requests have the same criticality [15][16]. Not all cache blocks are equally important [17]. Not all threads limit performance [18][19][20]. Not all queries/requests in a server need to be serviced fast. Not all threads or programs are of the same importance. Not all critical paths in logic are exercised frequently [21]. Recent research has shown that 1) identifying critical instructions and executing them in fast pipelines improves both performance and power in single-threaded applications [22], 2) identifying the slack of network-on-chip packets and prioritizing packets with lower slack [15] can large improve performance in multiprogrammed systems, 3) identifying limiter threads in parallel applications using cooperation between the hardware and the runtime system and prioritizing such threads in the memory controller can improve both performance and energy [19], 4) identifying latency-sensitive applications and prioritizing them in the memory scheduler can yield significant performance improvements [23], 5) identifying critical threads in parallel applications and slowing down all others can lead to large power reductions [18]. We believe these techniques can also improve other metrics. Many other such opportunities exist to manage resources using the concepts of slack, criticality, and latency-sensitivity.

Efficient systems should identify criticality and bottlenecks at different levels of processing (circuit, logic, instruction, request, task, program, system, etc.) and accelerate/prioritize critical/bottleneck tasks while spending little energy on non-critical tasks. This could be done by 1) identifying the "slack" present in each task and 2) designing resources and 3) managing execution such that the slack of every task is always as close to zero as possible.

All three are promising and unsolved research directions that can enable us to develop novel "criticality interfaces" to largely improve the way power and performance management is performed today.

2. Interfaces that can enable designers to reason about the effects of their decisions on system power. Feedback/control loops for effective power management.

A critical research question going forward is to design such interfaces that can enable designers at the higher layers to reason about the effect of their decisions on global metrics such as system power, QoS, performance. For example, feedback mechanisms that the software designers can probe to measure the effects of their algorithms, functions, computation structures, etc. on power/energy would be very valuable in designing power-efficient software, especially within the context of mobile and server systems where power is at a premium. Similarly, such feedback mechanisms would be very useful for the system software and compiler in managing dynamic execution with the goal of maximizing power efficiency.

3. Secure interfaces for exposing and controlling power/thermal sensors and fine-grained power/thermal management mechanisms.

Exposing fine-grained power/thermal management and measurement mechanisms to the higher layers can significantly aid cross-layer power optimization. Key research challenges lie in doing this in a secure fashion.

4. New interfaces for managing heterogeneous execution

Management of heterogeneous resources can be accomplished using the concepts of slack, criticality, and latency-sensitivity such that tasks with low slack and high criticality/latency-sensitivity are allocated higher performance and higher power resources whereas non-critical tasks are allocated low-power resources (e.g. [18][19][20][24].) We expect a criticality based interface would significantly aid heterogeneous execution. Other interfaces are also very much worthy of research.

5. Coarsening the granularity of the hardware/software interface

Existing HW/SW interfaces communicate information at a very fine grain, using simple instructions. A much more expressive interface can be formed by coarsening the granularity of communication and using "blocks" as a means to communicate between the hardware and software. These blocks could specify various information, including resource requirements associated with computation, criticality of computation, reliability requirements of computation, etc. This information can largely aid optimization at the hardware layer. A coarsened, block-level feedback interface from the hardware to the software can enable dynamic optimizations and adaptation of policies at the software layer.

Area 4 – Systems, Applications, and Beyond

Area Leader: Luca Benini

Co-Editor: Paul Bogdan

Other Area members: David Andersen, Pai Chou, Rajesh Gupta, Mark Hill, Benjamin C. Lee, Per Ljung, Jose Moreira, Kunle Olukotun, Viktor Prasanna, Parthasarathy Ranganathan, Vijay Janapa Reddi, Steve Swanson, Tom Wenisch

Information technology (IT) has dramatically transformed the world we use to live in from sporadic epistemological based type of communication to an extremely turbulent, to some extent, chaotic communication environments where computing is part of our life (see the "Success Examples in Computing Systems" textbox for a short summary of achievements in computing systems). Nevertheless, the exponential growth in computing performance sustained by transistor miniaturization has reached a critical point where even extremely parallel architectures cannot reduce power consumption within a sustainable envelope (see National Research Council's findings [25] below).

A system can often be described as a hierarchy of independent modules, interacting across well-defined interfaces. Cross-layer optimization implies additional state information or algorithms can be used if additional information is provided beyond the existing interfaces. For example a lower module can provide hints to a higher level module, and similarly a higher module can provide requests to a lower level module. This working group diverges slightly from this categorization and regards cross-layer power management as any cross-disciplinary mechanism that can influence power or energy.

Along the same lines, President's Council of Advisors on Science and Technology (PCAST) [26] advocate for an "end of performance increases in individual processors, making the use of numerous processors in parallel systems indispensable, as well as the increasing need for system designers to minimize power consumption and heat generation. Improving battery life and energy efficiency can be at least as important as making devices faster". The need for a cross-layer (holistic and comprehensive) review of design methodologies of future computing systems is not only required to minimize power consumption and thermal issues, but also to offer a higher degree of robustness, reliability, and security.

Clear Examples of Success and Failure

We need to avoid hazardous situations [27][28][29] and build systems that we can trust and rely on in the context of continuously changing and interacting environments. There are many "failure examples" that would advocate for a cross-layer (cross-disciplinary) power management and most likely computing system design. For instance, overspecialization in micro-controller-based systems results in code that is hard to maintain, understand, or upgrade. It is tempting to specialize the solution to each application domain, but it is costly and impractical, because most

often the requirements may be just slightly different but the implementation can be very different. Even more so, under specialization using OS is more limited and requires a platform with a lot more resources. Another failure example is the power management in a bus-powered wired network such as Power over Ethernet or Power over CAN (Controller Area Network). Power managing the nodes by themselves is ineffective because the link needs to be kept on but is expensive. Other failure examples are represented by micro-solar-powered cell phones, Java on deeply embedded systems, automatic cloud-based sync for mobiles (e.g. iCloud). Nice idea but in practice drains the phone battery very quickly.

Along the same lines of defective power management solutions (mostly because of a narrow-sighted single-layer optimization approach), Android power management uses wakelocks, which prevent some resources to shutdown of various components (e.g. screen) for some applications. This is on top of the Linux OS, which already has proactive power management (cpu-idle & cpu-freq). The end result is that Android applications can take wakelocks and prevent the Linux OS power management to kick in and minimize the device power usage. Unfortunately, Android applications exist within the Android virtual machine and there is no way for Linux to know if a wakelock taken by an application that is currently not in focus can be overridden or not. Hence you can have apps that dry up your battery even if they are not being executed.

National Research Council's perspective on future computing systems

Findings

- The information technology sector itself and most other sectors of society—for example, manufacturing, financial and other services, science, engineering, education, defense and other government services, and entertainment—have grown dependent on continued growth in computing performance.
- After many decades of dramatic exponential growth, single processor performance is increasing at a much lower rate, and this situation is not expected to improve in the foreseeable future.
- The growth in the performance of computing systems—even if they are multiple-processor parallel systems—will become limited by power consumption within a decade.
- There is no known alternative to parallel systems for sustaining growth in computing performance; however, no compelling programming paradigms for general parallel systems have yet emerged.

Recommendations

- Invest in research in and development of algorithms that can exploit parallel processing.
- Invest in research in and development of programming methods that will enable efficient use of parallel systems not only by parallel-systems experts but also by typical programmers.
- Focus long-term efforts on rethinking of the canonical computing "stack"—applications, programming language, compiler, runtime, virtual machine, operating system, hypervisor, and architecture — in light of parallelism and resource-management challenges.
- Invest in research on and development of parallel architectures driven by applications, including enhancements of chip multiprocessor systems and conventional data-parallel architectures, cost-effective designs for application-specific architectures, and support for radically different approaches.
- Invest in research and development to make computer systems more power-efficient at all levels of the system, including software, application-specific approaches, and alternative devices. Such efforts should address ways in which software and system architectures can improve power efficiency, such as by exploiting locality and the use of domain-specific execution units. R&D should also be aimed at making logic gates more power-efficient. Such efforts should address alternative physical devices beyond incremental improvements in today's CMOS circuits.
- To promote cooperation and innovation by sharing, encourage development of open interface standards for parallel programming rather than proliferating proprietary programming environments.
- Invest in the development of tools and methods to transform legacy applications to parallel systems.
- Incorporate in computer science education an increased emphasis on parallelism, and use a variety of methods and approaches to better prepare students for the types of computing resources that they will encounter in their careers.

Another classical failure example is the overdesign to maintain unnecessary abstractions. The guard bands used by the hardware (chip and system) designers have increased from 10% to over 60% in less than two decades with no end in sight due to increasingly unreliable and variation-prone nano-scale Moore and beyond Moore microelectronic devices. Latest measurements show 1.8X power consumption over nominal active power and over 9X sleep power consumption caused by increases in process variability. The situation gets worse with variations in the operating conditions (e.g. temperature, which exponentially affects leakage). The chief reasons for this overdesign are the abstractions needed to maintain how hardware and software designers see the system.

Going forward to more than Moore devices including CNFET, this variation -- measured as 3xsigma/mean variation increases to 70% for 32nm CNFET devices. The basic reasons for this variation and resulting guard bands are two-fold: as device dimensions shrink in the range of molecular dimensions, manufacturing variations bound to become a greater portion of variability seen across devices. Secondly, our system designs -- from devices, circuits to architectures --

follow a methodology that seeks to optimize a design for "average" design parameters: that is, the designs continue to be centered around mean of parametric distributions that are increasingly spread out.

Getting into specifics, there are numerous failure examples in current computing systems such as: Android wakelock for sleep prevention (see above), carriers preventing 3GPP quick disconnect to minimize signaling traffic, cellular network controlled power level of mobile radio which overrides user, Windows laptop not sleeping when lid is closed, cheap voice using Skype, but Skype needs UDP keepa-live signals every 20s making it energy expensive, WIFI beacon congestion, app/OS/hw/firmware often designed by separate entities, internal benchmarking shows 2x difference in mobile handset energy efficiency between both external vendors and internal product teams, centralized resources (e.g. shared memory coherence, multiport memory controllers) have well-known scaling issues.

Also, the cellular network controls the power state of handsets. The high power DCH state used for data communication has a T1 timer that changes the state to a lower power FACH state, which in turn has a T2 timer that changes the state into an idle state. The AT&T femto basestation is configured with a 4m T1 timer preventing handset from sleeping. Ideally the better SNR of a nearby basestation would enable less communication energy to be used, but the carrier configuration of a very long T1 timeout prevents the 1W radio from sleeping. If a mobile handset is configured to check for email every 20m then the battery of a typical handset is depleted after only 4h. Similarly, TCP is commonly used for both wired and wireless connections. Unfortunately TCP does not distinguish between dropped packets and congestion encountered in wireless systems. A dropped packet therefore results in a TCP slow restart significantly reducing throughput, indicating that TCP is not optimum for wireless connections. A wireless workaround is to use striping with multiple TCP connections to minimize the impact of a single restart.

Intel's TurboBoost typically disabled by HPC operators because it interacts badly with scheduling/load balancing mechanisms falls under the category of failure examples calling for a cross-layer power optimization and management.

Modern, commercial datacenter servers are homogeneous and are not balanced to reflect the diversity of emerging applications. Some applications have compute and communication requirements (e.g. online transaction processing, databases) while others have much lower communication requirements (e.g. distributed memory caching, big data analytics). Homogeneous machine balance for heterogeneous application intensity is a failure to provide energy-efficient computing.

With democratized access to cloud computing (e.g. Amazon EC2, Microsoft Azure), formalizing and generalizing auto-tuning frameworks will be increasingly important. At present, the burden of extracting performance from datacenter hardware is difficult and will become more so. Users cannot easily determine application intensity and the required machine balance. Without frameworks to help users extract performance from datacenter hardware, the proliferation of cloud computing is in jeopardy.

Success Examples in Computing Systems

- Shifting computation to different datacenters based upon the availability and cost of power and cooling.
- iPad getting 10 hours of battery life playing movies.
- Kindle getting 2-3 weeks of battery life using E-ink.
- Proxy based systems (Ethernetproxy, sleep proxy, sleep server) have demonstrated up to an order of magnitude reduction in active power consumption by systems that are able to exploit inherent system heterogeneity, or introduce heterogeneity to enable new power control "knobs". Examples of these are paging radio and secondary domain processors that effectively create new "hybrid power states".]
- Recent system implementations (e.g. Convey) point architectural mechanisms that enable applications-specific coprocessors to be integrated and programmed.
- Coordinated application-architecture tuning has had success in big applications. High-performance scientific computing has made advances in auto-tuning software frameworks for portability and efficiency (e.g. linear algebra, signal processing). Commercial datacenters with a few, large applications are also tuned and optimized (e.g. Google search engine is frequently re-written and optimized -- see Barroso and Hoelzle).
- High-performance scientific computing robustly defines machine balance and application intensity. Machine balance is the ratio of peak computational throughput (op/s) to communication bandwidth (byte/s) while application intensity is the ratio of computational demand (op/s) to communication demand (byte/s). Partly, as a result of this robust matching of machines to applications, HPC has successfully navigated the design space and chosen small, energy-efficient cores when appropriate for the application (e.g. Blue Gene, Green Flash).
- The Opera mini browser is a proxy browser that minimizes communication traffic, reducing latency and communication energy with slightly degraded fidelity of content. Similarly incremental precision (e.g. fractal images, font feathering, image pixelation, text before images) allows content to be initially displayed without being refined if the user skips to another item of interest.
- Event-based systems (QNX, tickles Linux) instead of timer/polling OS
- Arm Big-little
- Speculative web fetch
- The backlight of typical mobile displays is controlled by ambient light. This is significant on a laptop where the display is the dominant energy sink.
- Reactive sleeping systems that wake on events are both responsive and minimize static power. Common examples are wake-on-ethernet and wake-on-radio.
- High-level abstractions such as CUDA / OpenCL / WebCL can compiled to various hardware platforms, allowing performance increase, a productivity increase, and some portability. Auto-tuning (e.g. Berkeley Seijits) has demonstrated that even expert-designed libraries such as FFTW can be improved upon by 2x; aggregation of notifications, events
- Berkeley expert-designed "stovepipes" templates for common parallel design patterns
- Energy proportional computing where Vdd (sub, near, super threshold) set by computational load, OS controlled DVFS
- OS synchronization of event queues
- Positioning using last known/cell tower/wifi/ gps, select cheapest connection (BT, wifi, cellular), piggyback data transmission during voice call , burst transmission of streaming media enabling radio to sleep, using burst of N*Bytes instead of stream of N transmissions of Bytes, MIMO radio at edge of cell, device2device communication bypassing base station, multiple cheap hops in mesh network rather than 1 expensive hop, W8 always-connected, wakes on recognized network activity,
- MS somniliquoy low power process monitors network
- Cloud-based aggregation / enqueing of notifications, message push from cloud-based notification center, offloading e.g. Online gaming using remote processing
- Modern data centers (e.g. Google, IBM, HP) typically achieve PUE of 1.1 or better, through the use of technologies such as free cooling. On the negative side, this means there is not much more to improve in that front.
- Cores that morph between ILP and multi-threading have been moderately successful in delivering a trade-off between single-thread performance and throughput efficiency.
- Warehouse computing designs (e.g. Google, Facebook)
- HP's smart floor tiles and intelligent fan control - successfully connecting the "cyber" and "physical" aspects of data center optimization.

Current heterogeneous architectures cannot be programmed easily by most applications programmers. Existing applications can no longer take advantage of the additional compute

power available in these new and emerging systems without a significant parallel programming effort. Writing parallel programs, however, is not straightforward because in contrast to the familiar and standard von Neumann model for sequential programming, a variety of incompatible parallel programming models exists. Programming models are available; each with their own set of tradeoffs. Emerging heterogeneous systems further complicate this challenge as each accelerator vendor usually provides a distinct driver API and programming model to interface with the device.

To overcome these challenges for future computing systems, both the National Research Council's and President's Council of Advisors on Science and Technology (PCAST) reports advocate that "DoE and NSF should be major sponsors of research for achieving dynamic power management in applications ranging from single devices to buildings to the power grid.".

Challenges: Exemplified by the Case Studies and More

Advances in device physics and computing have led to the development of complex multiprocessor-on-chip (MPSoC) platforms supporting a large variety of applications. Nevertheless, future applications (e.g. big data analytics, gaming and entertainment, high-quality interactive environments, virtual reality, business analytics, recognition mining and synthesis, smarter planet applications, social media) will pose serious challenges for building models of both computation and communication that later can be used for dynamic power optimization (mapping and scheduling, on-chip traffic regulation, DVFS techniques).

Some applications are still computationally bound, most are at least equally bounded from the memory and communication viewpoint. Hence, power management strategies focusing exclusively on maximizing CPU energy efficiency are going to have limited impact. We need however better ways to gauge non-CPU resources at the application level (and in application development support tools). Memory and communication are much less carefully analyzed than CPU usage: thus we need more advanced profiling, online monitoring and optimization tools to maximize the energy efficiency of storage and communication synergistically with computation.

On one hand, we will have to deal with sophisticated computation over huge amounts of data. Understanding of shortcomings of current programming models corroborated with exploitation of exhibited patterns behind computational algorithms will be the key for establishing accurate prediction strategies. Such prediction strategies can infer which thread is most critical, which one will generate the highest number of memory accesses, or suggest how to deal with diversity in data types, etc. All this information can be exploited for dynamic optimization strategies targeting computation aspects.

However, optimizing the computation side alone cannot offer the best solution. Accurate on-chip traffic modeling (that takes into account time-dependent and fractal characteristics of network traffic) will be essential not only for predicting the timing requirements (network latency, system response times) and so guaranteeing a specific QoS level but also for defining reliable/robust and optimal dynamic optimization system strategies. Knowing or determining the right models that describe the network traffic over certain periods of time can enable predictions about system

performance and help to identify strategies to minimize latency during high load or localized congested regions.

For instance, Virtualized environments (managed runtime environments) are getting more and more common (VMM in servers and Android VM in mobile are most notable examples). These environments are challenging because isolate the SW application from the actual physical machine, and an application can run on a wide variety of different hardware, with different energy efficiency requirements and bottlenecks. We need to define a way for applications to exchange information with their virtual environment to help power management, but not do directly control it (as they are not empowered to do that). Extensions to concepts such as SLA are needed (not only in the data-center, but also in the mobile and deeply embedded, even though it makes more sense on larger systems).

Another challenge is represented by the necessity of designing cross-layer optimization strategies that exploit diverse methods like algorithmic transformations, dynamic control, compiler transformations, etc. Nevertheless, how do we make sure that the complexity we add in itself does not lead to further inefficiency? We need to somehow anticipate the rapidly changing application market space, especially in mobile systems, and prepare future system designs accordingly. Above all, there is great need for the right interfaces for sharing information across different layers and various scales. We need to identify the best global objective functions that encapsulate both performance and power consumption and their correlations between different layers. We need to provide access to actuators, to identify the right time constants for control purposes, to communicate information about system state that can enable accurate and robust actuation. More broadly, we need to provide visibility to system controllers across all layers while minimizing exchange of global information.

Related to all the above, we need to determine and define what the “perfection” would mean in future computing systems. Having a way to reason about some bounds on least energy for a particular task and particular architecture would help identify points of diminishing returns. We need to learn how to exploit at runtime the consumer usage patterns and the availability of resources.

Consolidation of workloads in a single server or installation of servers leads to higher machine utilization and therefore better energy efficiency (more work done per $ of energy). In the technical side, isolation, security and privacy are the biggest barriers to further consolidation. Commercial customers are reluctant to share infrastructure with other customers. For example, even when two customers are collocated in the same data center they often require separate and isolated networks, storage and servers. If we could solve that problem, we would greatly increase the demand for cycles in a shared environment. Consider the case of Dreamworks, as presented in SC11 (http://sc11.supercomputing.org/schedule/event_detail.php?evid=mswk113). Most of their rendering is done a few months before the release of a new movie. If they had access to more computing power on demand they would have more freedom of schedule to release their movies (significant market opportunity) and would not need as big of a proprietary server farm. In addition, rendering is a perfect example of a background workload that can be used to fill empty computing cycles. However, it is easy to see that those computations are highly confidential as Dreamworks does not want their frames leaking.

Cross-layer power optimization and management should guarantee optimal power values. For instance, it is hard to manage the power consumption of a sensor network system that consisting of a large number of nodes interconnected via wired or wireless interfaces. More precisely, it is hard to coordinate multiple nodes because it is expensive to keep the links alive for inter-node communication. A similar optimization challenge is represented by power distribution across a locally networked system (e.g. power over Ethernet, power over Controller Area Network). Main stream research doesn't usually discuss the energy efficiency of such systems because they are invented for convenience of connection and not for energy efficiency. Nevertheless, a large fraction of power can be lost on the power transmission line. This is cross-layer because it bundles supply power level with network protocol level and system-level power (consumption) management.

Along the same lines of a holistic approach is represented by the significantly high power values of mobile devices A typical smartphone handset has a 5Wh battery but barely lasts one day before it is recharged, resulting in an average power of 200mW. The static power of a phone with all notifications turned off is about 20mW comprises the radio modem (about 5mW), memory refresh, and any OS idle tasks. The static power with typical notifications (email, Skype) turned on varies from 50-150mW depending on the model. Worldwide quantitative measurements by Nokia show that static power typically consumes 50% of the battery energy.

Additionally, from a security point of view, we need to deal with bad or greedy “non-trusted” controllers and identify strategies to avoid malfunctioning or malicious attacks that can render system integrity in part or as a whole. It is imperious necessarily to exist a cross-layer analysis such that such dynamic control is done in an optimal way via both hardware and software means.

Getting now into more details, cross-layer specialization introduces a fundamental tension between power and expressiveness, on the one hand, and complexity and coupling on the other. The compelling power savings advantages of cross-layer optimization must be balanced by the substantial big-picture system challenges they introduce: Engineering has long relied upon strong separation of modules and concerns in order to facilitate independent development, testing, and failure and performance isolation. A major question that research should address is how to walk the line between these two ends.

Moreover, there is a need for the right balance between the level of abstraction needed versus the operational efficiency. This balance may shift dynamically (e.g. multi-radio interfaces). Developing the right application level abstractions that enable programmers to develop functionality and reason about their application performance without burdening them with the details of the underlying architecture features is also very important.

Layers were established to manage complexity by having interfaces to implementations that are “black boxes.” How do we peek into boxes without unduly making interfaces complex and exposing implementation aspects that may then constrain future implementations. One option is using so-called “grey boxes” where one enters implementation attributes by probing rather than widening the interface.

Managing the tension between abstraction and cross-layer management will not be an easy task. Some examples may include the network stack abstraction, the division of the architecture into

multiple components each individually designed, etc. In addition, performing a collaborative cross-layer optimization that would enable system scaling is of great demand for future computing systems. For instance, a cloud datacenter can often have millions of moving parts and cross-layer collaboration across such a large ensemble that could have to deal even with dynamic availability is going to be a big challenge [30][33].

A natural introspection would be if we can alleviate some of this tension by statically "compiling" or "pre-optimizing" applications across the boundaries. Systems already do "pre linking" to reduce dynamic library load time (they assign each library a fixed virtual address for loading and then pre-link all the other libraries against it). If we restricted (or provide a way to clearly specify) how components fit together, we could aggressively optimize across layers, and preserver (or less the impact on) the expressive interfaces that are so valuable. Sacrificing programmability or even requiring large-scale changes to existing interfaces is going to be a tough sell. Ultimately, programmers care more about productivity that saving power.

In addition to all the above, we need to consider the problem of finding the right architecture that allows for an optimal cross-layer coordination and optimization. Hierarchical seems to be the general approach, but should we be looking at federated peer2peer architectures? While distributed architectures are likely the way to practical realization, what do we leave on the table compared to a perfect "global-knowledge" centralized controller? On a related note, how do we determine the perfect division of labor -- across hardware and software, across platform and cluster, and so on? All such questions are becoming even more important when considering the power optimization while dealing with heterogeneity in both hardware and software domains.

One way to address the heterogeneity issue is to rely on compilers and static analysis. Static analysis adds less energy/power overhead to managing power consumption and can be more aggressive / computationally intensive because it can be done offline. For instance, one can examine an application to identify regions that are a good / efficient match for the system heterogeneous resources. There are a variety of techniques that would enable this analysis. The simplest is identifying calls to hardware supported/accelerated libraries. Using pattern-based to identify common sequences of calls to libraries (there's been some software engineering work along these lines that specifies valid sequences of library calls. For us, the question is not "valid" sequences, but sequences that are amenable to a particular heterogeneous resource). Another tool for dealing with heterogeneity is to use graph-based similarity / isomorphism / pattern matching techniques to identify code regions that are amenable to a particular type of resource. These kinds of analysis could be combined with a load time/install time binary conversion to allow for transparent utilization of heterogeneous resources. A changing pool of heterogeneous resource could be continuously matched against annotations on running program to identify opportunities for utilizing newly available resources. Nevertheless, these are not the only options.

Alternatively, to deal with various types of hierarchies and heterogeneity we need to design online learning strategies that can profile / monitor both application and architecture figures of merits and decide on the fly whether to turn off or allocate more resources for energy efficiency or any other form of re-configurability. Nevertheless, there is a need for identifying the best profiling aids or the right system metrics (energy per operation, GOPS). One other challenge would the scale at which profiling should take place and what tools should be used to reduce its

overhead. There should be a trade-off between fine-grained and coarse grained which might be dictated by both application and architecture features.

Cross-layer collaboration requires co-designing and coordination across different disciplines. Providing the right educational frameworks to enable the next generation of researchers to embrace cross-disciplinary research is very challenging. Future research and education curriculum need to address challenging questions such as: How do we get the average programmer to understand power constraints and program accordingly? Is it through tighter integration of research in the classroom? Should we still start and extensively teach, analyze and scrutinize sequential computing or jump much earlier into parallel world of computing? Wide distribution of easy-to-use tools that measure or simulate the energy costs of computation and communication are needed before the developer can explore alternative implementations. One example limited to Nokia mobile phones is the Nokia Energy Profiler which graphically shows power correlated with cpu activity, cpu frequency, packet transmissions, radio state, and display state.

The interdisciplinary nature of cross-layer optimization creates practical challenges in funding and publishing the research. NSF should seek mechanisms (e.g. perhaps joint programs with ENG) that ease the challenge of funding projects that span mechanical, materials, electrical, and computer science aspects of CPOM. Moreover, research publication venues that encourage cross-disciplinary research will also be important.

Opportunities Across Layers

On one hand, the hardware cannot perform power optimization alone. For power optimization purposes, software need to continuously adapt based on inferences about the input or application to be executed. *On the other hand, the application cannot perform power optimization alone either.* Several reasons are the lack of global visibility; programmer effort and competence required, lots of hardware-specific behavior that it would have to adapt to and understand. Nevertheless, adopting maybe the “gray box” paradigm would mitigate all the above and open the avenues for truly cross-layer power optimization and management.

There are good reasons in following this idea. Intuitively, eliminating the middlemen is always more efficient! The prior work in this area that have looked at piece-wise cross-layer collaborations (many listed in first section) have, time and again, shown the potential from this approach.

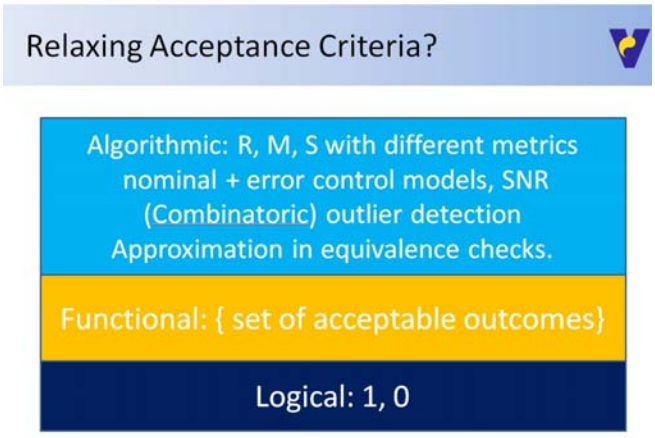
It is possible to (automatically) configure and customize manufactured hardware to individual systems and even to dynamic operating conditions through a software stack that can adapt to an individual system and its components through adaptive instruction sets, compiler optimizations and runtime monitors. This capability is possible by making the micro-architectural design and architectural interfaces between software and hardware more transparent. No longer will the data-sheet be sufficient (or even necessary) part of system design optimization. Just as instruction-level cycle-counting became irrelevant to software delay estimation in the face of increasingly non-deterministic execution hardware, an increasingly variable hardware forces us to rethink the contract between hardware and software in an era where software is increasingly becoming “reflective” of the environmental needs in embedded computing environments.

Along these lines of thinking, we require a managed runtime layer that is dedicated for power management. This layer should co-exist harmoniously with other system software. Its job should be to continuously monitor execution, and dynamically tailor the execution of different program threads' code to meet application and system-level power, performance and reliability constraints.

Recent advances in circuit-level designs -- notably, Razor, EDA and TRC -- have shown that it is possible to respond to path delay variations in a manner that systematically captures dynamic errors in computation, tune the supply voltage and/or borrow time from the clock to ensure the circuit remains operational and is an energy-efficient by directly reducing the voltage margin and the timing guardbands. Going further up the hardware/software abstraction stack, recently the programming language community has devised "decompose, calibrate and acceptance" tests (through principled approximation, [31], and Probabilistic accuracy bound and early phase termination [32]) that shows feasibility of reliable/recoverable computation that minimize energy use. At an even higher level of abstraction, it is possible to build functions and applications (e.g. using the notion of algorithmic noise tolerance or ANT) that provide for use of a "relaxed" acceptance criteria that enables use of a higher level semantic information to mitigate the effects of uncertainty in computation.

In contrast to recovery mechanisms for occasional faults (e.g. Razor, TMR), stochastic computing functions with continuous significant faults. By relaxing the fidelity of the computed results, such a mechanism can be adaptive to process, voltage, temperature and aging effects for DSP style algorithms.

The ANT work leading up to SSNOC, SoftNRM, VASCO and ERSA have shown it is possible to exploit the structure in errors to improve reliability. The results have shown 3X gain in energy efficiency in DSP applications. The next step is to introduce variability monitors in software that enable dynamic optimization of binaries to adapt the computation to process variations, aging and temperature conditions using appropriate sensors that are embedded in the physical computation substrate and expose compact "hardware signatures" to the software. This in turn presents challenges for the system software stack to opportunistically exploit the variability, such as where in the software stack the control should be done and how should the contract between the application and the OS and compiler be redefined to enable variability to be exploited for power optimization.



Possible benefits

As emphasized in Section 2, before dwelling into the possible benefits of a cross-layer power optimization and management (CPOM) approach, there are many issues that should be addressed. For instance, it is unclear whether the metrics we have today are sufficient to evaluate the benefits of CPOM. Also we need to identify to what degree we need to rely on hardware and software solution from the system optimality point of view. Nevertheless, all such questions and many stated in more details in Section 2 provide strong basis for believing in significant benefits for CPOM.

One possible benefit could be the significant gain that can be obtained by crossing into power supply and energy storage layer. Depending on the duty cycle, distance, scale, it can be 10x-100x in energy savings.

Furthermore, exploiting higher communication bandwidth than consumption throughput allows burst transmissions, enabling the radios to sleep reducing overall power. For example a single large burst can use considerably less power than the same amount of data sent in smaller fragments. Burst transmission for sleeping wifi [35]) can also help at mitigating the power consumption values.

Other motivating examples showing the benefits of a CPOM approach are represented by Arm Big-Little (400 pJ/op vs. 1000 pJ/op), proxy browser (1J vs. 4J), 2x piggybacking (1W v 2W), 50x offloading onlive gaming (2W vs. 100W), positioning (1s vs. 10s), Berkeley seijits auto-tuning shows 2x improvement over expert-library FFTW.

By adopting an end-to-end approach about actually usage and relying on a CPOM approach, one can get 10x benefits in power or energy values. For instance, a mobile device can briefly use 32W even if it is in 2W package, as long as its computation duty-cycle is low, as it is in many emerging uses [36]. We fully expect more opportunities like this.

Embracing the CPOM methodology is also needed for coordination in distributed systems or sensor networks. For instance, there have been some efforts that looked at coordinating the time at which sensor measurements are taken on sensor nodes -- so that if you have two processes that each want to read the GPS, it only warms it up once.

All in all, CPOM could not only change the abstraction models or the current design algorithms, but by making the systems more energy efficient could indeed make them more pervasive in many societal fields.

Possible approaches (key ideas, promising areas that should be investigated)

For building successful CPOM methodologies we need to rely on intelligent and accurate mathematical relationships between power, performance and reliability. This calls for a rethinking of hardware/software interfaces, re-defining in a holistic manner the storage / memory hierarchy from the disk to the cache, and new software execution models better tailored to emerging hardware trends (with reduced software bloat). Researchers have been harping on power efficiency by identifying lulls in program behavior. However, we need much more aggressive techniques where we intelligently make trade-offs with the level of reliability that is

required within an application. In addition, although ignored mostly by previous and current approaches, there is a great need for new metrics that quantify "end-user experience" (mobile space) and workload service-level agreements (in the enterprise space). Ultimately the better we understand the intent behind how computers are used and communicate that effectively across the layers, the more energy-efficient we can get.

One possible approach is to consider auto-tuning, self-calibration and self-regulation, together with higher abstraction levels (OpenCL, WebCL) integrated or merged in with operating system. For instance, while now the OS deals with the GPU resource as an IO device, we would rather prefer that the future OSes will treat it as an implicit part of its resource pool.

In-situ energy/power measurements and the information availability across different layers can enable the self-calibration, feedback control of the entire system toward minimum power consumption values. Depending on the nature and characteristics of the application, it will be possible to propose adaptive systems that exploit the QoS tradeoff between fidelity and energy. In addition, relying on accurate mathematical analysis of either the historical location patterns or prediction mechanisms for speculative pre-fetching can also contribute to lower energy consumption and possibly better performance.

Platform-specific cross-layer approaches

Deeply embedded

Direct 3D stacking of sensors, solar cells, batteries, and microcontrollers creates an opportunity for tight integration of power management. An example of this is the millimeter cubed computer system from Michigan [37].

Continued development of coordinated communication and computation in meshes of sensor networks for the Internet of Things is warranted.

Another example of cross-layer approach is the need for lightweight mechanisms that enable remote in-field reprogramming, supported by a powerful cloud back-end. CPOM approach will also prove crucial for the design of efficient convertors (dc-dc).

Along the lines of energy efficiency, there will be a need in both embedded and mobile platforms for optimal strategies that partition applications between client, local offloading, and remote cloud offloading while balancing local battery power usage, throughput, and responsiveness. We need programming tools to make this easier and rigorous ways to reason about how an application can be partitioned, as well as tools to investigate QoS of selected partitioning.

Mobile

Regarding the mobile specific cross-layer optimization, there is an urgent need for refactoring from mobile to cloud such that we minimize both static and dynamic power. The power of an idling mobile device is significant since it periodically activates its transmitter to poll for new notifications. Exploiting local resources using OpenCL / WebCL is in its infancy. Mechanisms to share computation and communication costs across client, local and remote resources need to be developed. Ideally a framework would make this available to non-specialized application developers, simultaneously reducing energy and reducing development efforts.

It will be desirable to push the power management goals beyond 1 day battery life to enable high-value always-on applications, achieve a better form factors, reduce size, weight, cost, and thermal. In addition reducing the radiation profile may be beneficial if it is found to be detrimental due to health related issues.

Enabling always-on applications will require ultra low power sensors, communication, computation fabrics. Efficiently exploiting heterogeneous resources will be necessary to span always-on applications such as low-priority background notifications, immersive user interfaces, and augmented reality. Hibernation of distributed operating systems running on different resources may be necessary for an efficient implementation, where a cheaper module triggers the fast resumption of a heavier module. The tradeoffs of moving computational state between different resources (e.g. Arm Big-Little, Nvidia 1+4 Tegra) compared to a distributed architecture are unclear with respect to ease of development, deployed size, and energy efficiency.

Exploiting dark silicon has been proposed to offload particular functions to efficient conservation cores to reduce power (rather than accelerators which try to improve throughput). It is presumed that the activity level of these conservation cores is low, which results in a larger die but low thermal budget. However given the price sensitivity of mobile handsets, it is unclear if conservation cores are economically viable. Further work identifying the tradeoffs between acquisition cost of expensive resources versus cheaper operational costs are warranted.

Since the mobile is energy limited, different performances may be desired depending on the state of charge. For example if only 20% of battery capacity remains then high power tasks may be run less often or with less fidelity, enabling basic communication to be maintained to an empty battery.

All the above cannot be done without a good understanding of the behavior of emerging application and the mobile user preferences. Applications and OS are not the only ones subject to optimization in this context.

With recent advances in technology and the possibility to develop circuits that can detect motion and convert it into energy, it will be beneficial to enhance future mobile devices with methodologies that harness all types of energy generation resources (kinetic, temperature, chemical, sound).

Relying on good metrics for measuring user experience and satisfaction are even more needed in the context of mobile platforms. Consider the user's demand for responsiveness when deciding how much power to invest in a particular computation. E.g. when the phone is in someone's pocket, compute slowly. When a user is staring at the display, make UI update go faster. This is cross-disciplinary in that it requires thinking jointly about power management, mathematical tools and human-computer interaction.

Server

The most important motivation for power management is to allow more compute capability to be deployed within a given facility (i.e. more computation within a fixed number of watts). A secondary goal is to reduce the operating cost (energy) per compute operation. There are many

alternatives that need careful investigation. For instance, resource sharing and disaggregation can reduce per-server over-provisioning. Also, we need to rethink the power distribution infrastructure to eliminate voltage conversion steps. This approach can be applied both to data-center wide distribution, power supply within the box, and voltage regulation on chip. Moreover, we need to redefine the stacking of memory systems and processors to reduce memory system energy requirements. We can improve energy efficiency by replacing spinning media with alternatives. However, in doing so, we must consider life-cycle costs (e.g. do the added energy costs of exotic materials in new storage devices outweigh operational energy advantages over the life of the device?)

Nevertheless, power management cannot be done without a systematic treatment of the uncertainty and variation existing in the system due to the application set and data input dynamics while still enforcing a certain level of performance.

Making efficient use of heterogeneous compute infrastructure (e.g. “wimpy” and “hefty” nodes in the same mix) can improve energy efficiency, but requires, e.g. identification of sequential bottlenecks (particularly for same-chip heterogeneity), straggers (multi-machine heterogeneity), multi-version compilation and tuning, etc.

Along the same lines, by exploiting the heterogeneity that comes from the application side (i.e. assuming that an application expresses its requirements to the hardware in terms of number of transactions per second) we can reduce the power consumption values.

Gaps and Potential Benefits

As emphasized in Section 2 and Section 3, there are many great challenges ahead of CPOM, but also potential benefits. It is well known that some applications could be beneficial to society, but we cannot implement them because energy consumption is not affordable. Examples of applications being prohibitive from an energetic point of view may include: advanced health informatics, enhanced virtual reality, advanced personalized learning, engineer tools for scientific discovery (e-science). In all these cases, the computing systems require more compute performance per unit of energy consumed.

CPOM needs to educate researchers to consider using non-traditional frameworks (e.g. map-reduce, openCL, etc.) in all platform levels. At the same time hardware is also changing (end-of Moore’s Law, the era of architectural innovation, and non-CMOS technology innovation). We are in an era of forced changes where we can use electrons to compute, ions for storage, photons for communication! Consequently, the relevance of CPOM is not at all questionable.

Maybe the biggest sources of inefficiency are the layers of abstractions we made to make the design complexity more affordable. We need CPOM to go across the layers to recover some of these inefficiencies. We need quantitative examples of software abstractions that have multiple orders of magnitude. ASICs represent the squashing of the abstraction stack whereas the gap between ASIC and Python implementations represents the potential for cross-layer optimization. We need abstraction tunneling provided by CPOM to recover the energy efficiency losses created by multiple “isolated” abstraction layers.

We believe that CPOM will help at reducing the thick levels of abstraction and enable the vertical re-engineering for energy efficiency. This implies a holistic analysis (e.g. engineering the Gmail system, revising the Java virtual machine in Android) of cost metrics across all layers.

It is not reasonable to expect that end application programmers can manage CPOM, but that instead middleware can support this (example - Websphere – all recovery mechanism is in the middleware, incorporated into the framework). For instance, we can introduce energy-awareness features in application-building engines (e.g. Google applications). Researchers developing middleware can introduce something in the application framework to identify latency, vs. throughput computation, vs. specialized computation. One way to do this is to specify responsiveness (or precision or refresh rate) requirements for programmer's code at a fine grain so that the runtime system can make decisions. Possibly it will help to provide a framework to specify utility functions (by the application developer) and then the middleware framework implements policies and makes the final decision.

Another important gap that needs to be overcome is related to the fact that local cross-layer optimization can create inadvertent cross-coupling between components. Also, CPOM virtualization emphasizes the tension between abstraction, speed of deployment, throughput, and energy efficiency. A CPOM research agenda should take into account the risks created by these tensions.

Research Opportunity with Applications and System Focus

Co-designed algorithms and architecture, specifically focusing on data movement and compute offloading. The cross-layer issues primarily stem from iterated design across hardware and software. This is a great time to attempt such a redesign since we have a unique confluence of expected changes in the hardware and software that enable such designs. Also a CPOM approach would help at building dynamic models and incorporate features and effects specific to certain devices (e.g. spintronic). Dealing with fractality, correlation, strong variability, non-stationarity within dynamic optimization framework has not been addressed by previous computer science methods and methodologies and those will play a crucial role for the design of future computing systems. We can exploit fractality to overcome complexity challenges.

Rethink the supply side in a distributed system. We are used to have stable supply of power (AC; DC etc.)., but as we get in harvesting and renewable, we need to start thinking that supply is not reliable and needs local buffering and transmission of power. In this context, the cost of communication for coordination is sometimes not affordable. This has not been done before and is totally hidden by an idealized power supply "abstraction". It will be useful to expose broader degrees of freedom.

Models and methods for special temporal variability, for application specific operation. The need for abstraction layers is foreseen to solve variability issues. This has not been done before for various cultural reasons. For instance, specifications given by HW to SW designers are considered as sacred, while in reality every HW system is "different" because of variations. By ignoring this you leave a lot of "personalized" optimization on the table.

Propose strategies for efficiently using accelerators in a virtualized environment. Regarding the cross-layer perspective, the investigation of low-level HW mechanisms that play well with the SW level that sits on top. The benefit of accelerators is because silicon has slowed down and with dark silicon the cost of accelerators is now more affordable.

Design and deploy heterogeneous server architectures (processing, memory networking storage). Regarding the cross layer perspective, it is interesting to look into how much heterogeneity you need and want up to the SW or how applications contend to the resources (i.e. application intensity vs. machine bounds). This has not been done before since previous work has focused on homogeneous environment for strong determinism performance guarantees.

Optimizing across layers designed and implemented by multiple vendors is extremely difficult. For instance a mobile device from a first vendor may contain SoCs and firmware from a second vendor, an OS from a third vendor, and apps from additional vendors. Typically these deliverables include proprietary information, and are not shared among other vendors. Identifying cross-layer opportunities is therefore extremely difficult since it is not possible to inspect different layers. As a result, higher layers cannot provide hints to lower levels for particular use cases with lower energy. In most cases, the user of the higher layer first has to identify a use case that can be optimized, and then contact the vendor of the lower layer to confirm the problem, and together work towards designing a cross-layer optimization. Defining and implementing such cross-layer optimizations requires coordination among 3 or more parties. In some cases a vendor can draft a proposed cross-layer API which a second vendor implements under contract. As a result, there is limited opportunity for design exploration by the greater community. Developing easy, robust mechanisms to facilitate automated cross-layer interfaces would be important.

Improve computation efficiency by consolidation (utilization and less redundancy). Regarding cross layer perspective, because it needs several levels (e.g. security), it is essential to find commonalities across several levels. Although the consolidation problem has been addressed in previous studies (e.g. workload consolidation), the tension between isolation and consolidation (achieving consolidation while providing the "illusion of isolation") and computational redundancy elimination are still not well or in a comprehensive manner explored.

Domain specific compiler framework that enables to take a specific application and the abstraction that can be used by others. The goal is to generate specialized architecture and make it easier to target software to heterogeneous HW. This is a difficult problem – at the junction of huge heterogeneity and we need a solution. No one has looked at this approach, as in the past there was a search for generality.

Co-designed algorithms and architecture. Regarding the cross layer perspective There is opportunity for approaches for HW and SW iterated design. Have all these changes happening at the same time, makes it hard and so it was not done before as there were shortcuts.

Top down understanding of characteristics of emerging new applications – browser as the number one application (and application container for mobility). Regarding the cross layer perspective, There is interest in a vertical optimization top down from application to the HW.

This has not been done before; standard have been changing a lot recently to meet demands of end users, to browsers are a new beast.

Synthesizing special purpose hardware for code that is not accelerable. Regarding the cross layer perspective, we need to start from the software, identify which are the software sections that are important, squash to HW and then orchestrated the triggering with a runtime system. This has not been done before because dark silicon has not existed before.

Deep application specific case studies medical imaging platform. Regarding the cross layer perspective, there is interest to cut across the whole abstraction layer and investigate case studies to show convincing results with a quantification and clear advantage. This has not been done before because only recently we have algorithms that are computed limited, and only now we are within one order of magnitude for a handheld platform.

References

[1] D. Shin, Y. Kim, N. Chang, and M. Pedram, "Dynamic voltage scaling of OLED displays," *Proc. of the 48th Design Automation Conf.,* Jun. 2011.

[2] M. Dong, K. Choi, and L. Zhong, "Power modeling of graphical user interfaces on OLED displays," *Proc. of the 46th Design Automation Conf.*, July 2009.

[3] H. Yu, L. Zhong, A. Sabharwal, and D. Kao, "Beamforming on mobile devices: a first study," *Proc. ACM Int. Conf. Mobile Computing and Networking*, September 2011.

[4] S. Chakradar, M. Sankaradas, V. Jakkula, S.Cadambi. "A dynamically configurable coprocessor for convolutional neural networks," ISCA 2011, pp. 247-257.

[5] Special Session on Brain-Inspired Architectures: Abstractions to Accelerators, ICCAD, 2011

[6] C-Y. Tsai, Y-J. Lee, C-T. Chen, L-G. Chen. "A 1.0TOPS/W 36-Core Neocortical Computing Processor with 2.3Tb/s Kautz NoC for Universal Visual Recognition," ISSCC 2012.

[7] M. Suri et al. "Phase Change Memory as Synapse for Ultra-Dense Neuromorphic Systems: Application to Complex Visual Pattern Extraction," IEDM 2011.

[8] B. Smith, E. Ruetsche, and B. Michel, "Toward zero-emission data centers through direct reuse of thermal energy," *IBM Journal of Research and Development*, May 2009.

[9] C. Li, W. Zhang, and C.B. Cho, "SolarCore: Solar energy driven multi-core architecture power management," *Proc. of the IEEE 17th International Symposium on High Performance Computer Architecture*, 2011.

[10] A. Nigam, C.W. Smullen et al. "Relaxing Non-Volatility for Fast and Energy-Efficient STT-RAM Caches," *Proc. of HPCA*, 2011.

[11] E. B. Condit, J. Nightingale, C. Frost, E. Ipek, B. Lee, D. Burger, and D. Coetzee. "Better i/o through byte-addressable persistent memory," *Proc. of SOSP*, 2009.

[12] J. Coburn, A. Caulfield, A. Akel, L. Grupp, R. Gupta, R. Jhala, and S. Swanson. "NV-Heaps: Making Persistent Objects Fast and Safe with Next-Generation, Non-Volatile Memories," *Proc. of ASPLOS*, 2011.

[13] T. Moscibroda and O. Mutlu, "Memory Performance Attacks: Denial of Memory Service in Multi-Core Systems," *USENIX Security*, 2007.

[14] B. Fields, et al., "Focusing Processor Policies via Critical-Path Prediction" Proc. of ISCA, 2001.

[15] R. Das et al., "Argia: Exploiting Packet Latency Slack in On-Chip Networks," Proc. of ISCA 2010.

[16] E. Ebrahimi et al. "Prefetch-aware shared resource management for multi-core systems." Proc. of ISCA, 2011.

[17] M. K. Qureshi et al., "A Case for MLP-Aware Cache Replacement," Proc. of ISCA, 2006.

[18] A. Bhattacharjee and M. Martonosi. "Thread criticality predictors for dynamic performance, power, and resource management in chip multiprocessors." Proc. of ISCA, 2009.

[19] E. Ebrahimi, et al. "Parallel Application Memory Scheduling." Proc. of MICRO, 2011.

[20] J. A. Joao, M. A. Suleman, O. Mutlu, and Y. N. Patt. “Bottleneck identification and scheduling in multithreaded applications.” Proc. of ASPLOS, 2012.

[21] D. Ernst, et al., “A Low-Power Pipeline Based on Circuit-Level Timing Speculation,” Proc. of MICRO, 2003.

[22] B. Fields, et al., "Slack: Maximizing Performance Under Technological Constraints", Proc. of ISCA 2002.

[23] Y. Kim et al. “Thread cluster memory scheduling: Exploiting differences in memory access behavior.” Proc. of MICRO, 2010.

[24] M. A. Suleman, O. Mutlu, M. K. Qureshi, and Y. N. Patt. Accelerating critical section execution with asymmetric multi-core architectures. Proc. of ASPLOS, 2009.

[25] National Academies Press (2011.) The Future of Computing Performance: Game Over or Next Level? (available at http://www.nap.edu/catalog.php?record_id=12980)

[26] PCAST Report entitled “Designing a Digital Future”, (available online at http://lazowska.cs.washington.edu/nitrd/)

[27] Australia probe into iPhone 'fire' on plane, BBC News (available online at http://www.bbc.co.uk/news/world-asia-15932846)

[28] Man Claims Droid 2 Smartphone Exploded in His Ear, (available online at http://www.wired.com/gadgetlab/2010/12/droid-explosion/)

[29] Exploding laptops prompt Dell battery recall, The Telegraph, (available online at http://www.telegraph.co.uk/news/1526424/Exploding-laptops-prompt-Dell-battery-recall.html)

[30] L. A. Barroso and U. Holzle. 2007. “The Case for Energy-Proportional Computing.” Computer 40, 12, December 2007.

[31] W. Baek and T. Chilimbi, “Green: A System for Supporting Energy-Conscious Programming using Principled Approximation,” *Microsoft Tech Report*, 2009.

[32] M. C. Rinard, “Survival Strategies for Synthesized Hardware Systems.” *Proc. of MEMOCODE*, 2009.

[33] P. Bogdan and R. Marculescu, “Non-Stationary Traffic Analysis and Its Implications on Multicore Platform Design”, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.30, no.4, pp.508-519, April 2011.

[34] P. Ranganathan, “Saving the world together, one server at a time, ACACES 2011

[35] F. R. Dogar, P. Steenkiste, and K. Papagiannaki, “Catnap: Exploiting High Bandwidth Wireless Interfaces to Save Energy for Mobile Devices”, ACM MobiSys 2010.

[36] A. Raghavan, Y. Luo, A. Chandawalla, M. Papaefthymiou, K. P. Pipe, T. F. Wenisch, and M. M. K. Martin, “Computational Sprinting”, Proc. of the 18th International Symposium on High Performance Computer Architecture (HPCA), Feb. 2012.

[37] Millimeter cubed computer system from Michigan, (available online at http://www.edn.com/article/512851-Researchers_claim_millimeter_scale_computing_system.php)

Appendix

Workshop Organizers

Massoud Pedram (USC) -- CPOM Workshop Chair
David Brooks (Harvard) -- CPOM Workshop Co-chair
Timothy Pinkston (USC) -- CPOM Workshop Co-chair

NSF Sponsors

Sankar Basu (Program Director, NSF CISE/CCF)
Ahmed Louri (Program Director, NSF CISE/CCF)

Attendee List

Area 1 - Technology, Circuits and Beyond

Mohamed Allam (Qualcomm)
Kaustav Banerjee (UC Santa Barbara)
Keren Bergman (Columbia Univ)
David Blaauw (Univ of Michigan)
Eby Friedman (Univ of Rochester)
Lei He (UCLA)
Payam Heydari (UC Irvine)
Peng Li (Texas A&M)
Farid Najm (Univ of Toronto)
Michael Orshansky (Univ of Texas)
Kaushik Roy (Purdue) --- Area Lead
Sachin Sapatnekar (Minnesota)

Area 2 - Circuits, Micro-architecture and Beyond

Christopher Batten (Cornell)
Naehyuck Chang (Seoul National Univ)
Jason Cong (UCLA) --- Area Lead
Sandeep Gupta (USC)
Engin Ipek (University of Rochester)
Bill Joyner (SRC)
Eren Kursun (IBM)
Renu Mehra (Synopsys)
Vijaykrishnan Narayanan (Penn State)
Qinru Qiu (Syracuse Univ)
Karthick Rajamani (IBM Research, Austin)
Karu Sankaralingam (Wisconsin)
Mircea Stan (Univ of Virginia)

Lin Zhong (Rice Univ)

Area 3 - Micro-architecture, Systems and Beyond

Murali Annavaram (USC)
Rajeev Balasubramonian (Univ of Utah)
Pradip Bose (IBM)
Jeffrey Draper (USC)
Sudhanva Gurumurthi (U. Virginia)
Brucek Khailany (Nvidia)
Hyesoon Kim (GaTech)
Margaret Martonosi (Princeton) --- Area Lead
Rami Melhem (University of Pittsburgh)
Trevor Mudge (Univ of Michigan)
Onur Mutlu (CMU)
Mani Srivastava (UCLA)
Michael Taylor (UCSD)
Josep Torrellas (Univ of Illinois UC)

Area 4 - Systems, Applications and Beyond

David Andersen (CMU)
Luca Benini (University of Bologna) --- Area Lead
Paul Bogdan (CMU)
Pai Chou (UC Irvine)
Rajesh Gupta (UCSD)
Mark Hill (U. Wisc-Madison)
Benjamin C. Lee (Duke Univ)
Per Ljung (Nokia)
Jose Moreira (IBM Research)
Kunle Olukotun (Stanford)
Viktor Prasanna (USC)
Parthasarathy Ranganathan (HP Labs)
Vijay Janapa Reddi (UT-Austin)
Steve Swanson (UCSD)
Tom Wenisch (Univ of Michigan)

Government Agencies

Sankar Basu (NSF)
Brian Davidson (ST Associates)
Jon Hiller (ST Associates)
Charles J. Holland (DARPA MTO)
Ahmed Louri (NSF)