Position Statement

Heterogeneous systems composed of CPUs and GPUs have shown great promise in improving energy efficiency on parallel workloads. Today’s state-of-the-art GPUs are 5-10x more energy efficient than state-of-the-art CPUs on parallel workloads. However, there is still plenty of opportunity for additional improvement, something which is now especially critical since we are in a post-Dennard-scaling era where reduced operating voltage can no longer be counted on for reducing power dissipation with Moore’s law scaling. The good news is that there is ample opportunity for power reduction compared to current state-of-the-art systems if we focus in the right areas. At the micro-architecture and system architecture layers, we can reduce power dissipation by focusing on simpler processor architectures (less per-instruction overheads), reducing waste, improving physical locality of data, further exploiting heterogeneity (and introducing some fixed-function hardware specialization), and by enabling our designs to inch down voltage with better clocking and variation-tolerance mechanisms.

Metrics for understanding the tradeoffs between power efficiency (Perf/Watt) and cost efficiency (Perf/$) are important. Some approaches, such as aggressive voltage scaling down to near-threshold, may benefit Perf/Watt significantly, however, these come at significant detriment to Perf/$, and such tradeoffs should be taken into account at the micro-architectural and system level as well.

Answers to Specific Questions

* What micro-architecture is needed for effective implementation and exploitation of CPOM?

The future of parallel computing is heterogeneous, with an integrated SoC containing a few latency-optimized cores, hundreds of throughput-optimized cores, and some specialized fixed-function hardware. Whereas single-threaded performance on latency-optimized cores will continue to be important, power dissipation will primarily be dominated by plentiful throughput-optimized cores so it is imperative that we should focus our power optimization efforts there. These cores will focus on data parallelism, not task parallelism, since that is highly abundant for most workloads. Some data parallelism will be regular and vectorizable, other data parallelism less so, either due to data-dependent conditionals or memory access patterns or because short critical sections will reduce the available parallelism for periods of time.
* How does the global interconnect look like in systems with a large number of diverse and interacting IP blocks so as to power-efficiently support the required diverse traffic patterns?

Global interconnect built out of regular structured on-chip networks (rather than ad-hoc point-to-point interconnect between interacting IP blocks) has a number of advantages. In particular, power-hungry long global wires can be built using custom specialized circuits. Micro-architecture research for global interconnect should focus on energy-efficient communication (pJ/bit) with simple router micro-architectures that maximize on-chip resources.

* What on-chip synchronization/communication primitives are needed to support both coherent shared memory and message passing style traffic in multi- and many-core systems?

Current GPU architectures support tens of thousands of concurrent threads. Future heterogeneous systems may need to support over 100,000 concurrent threads running on throughput-optimized cores. Clearly, to make this concurrency work, hierarchical synchronization and communication primitives are needed. At the lowest level, dozens of threads need to communicate and synchronize very quickly using fast barriers (and other synchronization mechanisms) with fast communication through local caches and scratchpads. At the highest level, software synchronization mechanisms may suffice, although research into better support for fast atomics could prove fruitful.

* What system architecture (memory hierarchy, computational resources, network-on-chip, communication protocol, etc.) is optimal for CPOM while meeting the application-level requirements?

At the system architecture level, what is critical for power optimization is focusing on mechanisms for exposing and exploiting data locality. Future heterogeneous systems will be fed by a deep memory hierarchy and providing programming systems with the tools for most easily exploiting these memory hierarchies for performance and energy efficiency is critical.

* 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?

We can look to micro-architecture and system architecture techniques in mobile devices for examples where energy-proportional computing is already a first-class product feature. It’s not clear that a lot of new research per se is needed here, but rather, that engineering best practices in terms of clock gating, power gating, etc... need to be adopted in the server/cloud space. A good example of this is NVIDIA’s variable SMP technology in its Tegra products. Low-power and high-performance latency-optimized cores co-exist in the same SoC.

* What other questions should we be asking and answering?

What are the right metrics for evaluating power efficiency? e.g. Some have proposed GOPS/W, energy-delay product, etc... We need to choose metrics that strike the right balance between power efficiency (Perf/Watt) and cost efficiency (Perf/$), otherwise it leads one to the wrong conclusions.
Waking up to Bottleneck Realities!
Rajeev Balasubramonian, University of Utah

Spot Quiz: What is an SMB?
Hint: It consumes 14 W of power and there could be 32 of these in an 8-socket system.

Facts:

Bottlenecks:

- The memory system accounts for 20-40% of total system power. Significant power is dissipated in DRAM chips, on-board buffer chips, and the memory controller.
- DRAM chip power (Micron power calculator): 0.5 W. On-board buffer chip power (Intel SMB Datasheet): 14 W. Memory controller power (Intel SCC prototype): 19-69% of chip power.
- Future memory systems: 3D stacks, more buffering, higher channel frequencies, higher refresh overheads.
- And ... We have an off-chip memory bandwidth problem! Pin counts have stagnated.

(Exaggerated) State of Current Research:

- Number of papers on volatile memory systems: ~1 per conference.
- Number of papers on the processor-memory interconnect: ~1 per year.
- Number of memory papers that have to define the terms “rank”, “bank”: all of them.
  Year of first memory paper that leverages DVFS: 2011.
- Percentage of readers that have to look up the term “SMB”: > 89% (ok, I made up that fact 😊, but I bet I’m right).

For every 1,000 papers written on the processor, 20 papers are written on the memory system, and 1 paper is written on the processor-memory interconnect. This is absurd given that the processor and memory are the two fundamental elements of any computer system and memory energy can exceed processor energy. While the routers in an NoC have been heavily optimized, the community understands very little about the off-chip memory channel. The memory system is a very obvious fertile area for future research.
QUIZ 1

While most ISCA attendees know what a virtual channel is, most would be hard-pressed to answer 2 of the following 5 basic memory channel questions:

1. What is FB-DIMM?
2. What is an SMI?
3. Why are buffer chips placed between the memory controller and DRAM chips?
4. What is SERDES and why is it important?
5. Why do the downstream and upstream SMI channels have asymmetric widths?

In early 2009 (before my foray into memory systems), I would have scored zero on both quizzes. Such a level of ignorance is perhaps ok for esoteric topics... but unforgivable for a component that accounts for 25% of system power.

QUIZ 2

While most ISCA attendees know the difference between PAp and GAg branch predictor configurations, most will struggle to answer the following basic memory system questions:

1. How many DRAM sub-arrays are activated to service one cache line request?
2. What circuit implements the DRAM row buffer?
3. Where is a row buffer placed?
4. Why do DRAM chips not implement row buffer caches?
5. What is overfetch?
6. What is tFAW?
7. Describe a basic algorithm to implement chipkill (what is chipkill?).
8. What is scrubbing?

ACTION ITEMS FOR THE COMMUNITY

- Identify a priority list of bottlenecks. Step outside our comfort zone to learn about new system components.
- Find ways to address obvious sources of energy inefficiencies in the memory system: reduce overfetch, improve row buffer hit rates, reduce refresh power.
- Find ways to leverage 3D stacking of memory and logic. Exploit 3D to take our first steps in the area of DRAM chip modifications (an area that has traditionally been off-limits).
- Understand the considerations in designing memory channels and on-board buffer chips. Propose new channel architectures and new microarchitectures for buffer chips.
- Understand memory controller microarchitectures and design complexity-effective memory controllers.
- Design new architectures that integrate photonics and NVMs in the memory system.
*What micro-architecture is needed for effective implementation and exploitation of CPOM?*

(1) Hardware performance counters: There should be more number of hardware counters to provide power and energy numbers to programmers/applications.
(2) Give more power/energy control ability to software: Hardware should expose individual power control features to software such as DVFS and power/clock gating features.

*What system architecture (memory hierarchy, computational resources, network-on-chip, communication protocol, etc.) is optimal for CPOM while meeting the application-level requirements?*

Heterogeneous computing or/and heterogeneous memory systems will be power efficient systems. Special hardware components for different applications will be power efficient.

*What advances are needed to support performance isolation (physical or virtual) in systems with shared resources?*

Software managing systems should know the characteristics of co-running applications very well. In order to do that, there should be some standard ways of communicating application characteristics to hardware or other software systems. When hardware or software management systems know application characteristics, they can do much better jobs. Dynamic sampling or many other research papers such as partitioning algorithms deal with these issues but so far we have not agreed on any form of common method to transfer application characteristics.

*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?*

In cloud computing domains, software systems become much bigger because of multiple layers of virtual machines or heavy usages of libraries to provide easy development environment etc. Architecture researchers should explore the characteristics of these applications more specifically. Some specific features to accelerate virtual machines can be an example that improves power efficiencies such as indirect branch predictors, hardware based parser, hardware hash functions etc.

*What other questions should we be asking and answering?*

How can we provide power models to academia? Unlike performance models, academia is still struggling with getting good power models. Academia also needs more standard tools to evaluate power and energy efficiency.

*What other the research issues in this area?*

What kind of new research directions do we need to do in mobile platforms?
1. **What micro-architecture is needed for effective implementation and exploitation of CPOM?**
   The capability to operate at several voltage levels.
   Fast response to voltage changes (boosting). Multivoltage domain support.
   Fast switching between supply voltage levels.
   Hardware and software control of these features.


Use QoS metrics to control power management for deadline driven situations, e.g. in big-core / little –core situations.

2. **How does the global interconnect look like in systems with a large number of diverse and interacting IP blocks so as to power-efficiently support the required diverse traffic patterns?**
   Point-to-point non-blocking interconnect. Keeps control simple and buffering to a minimum, which saves power. Point-to-point makes a variety of QoS schemes easier to implement. Managing power domains across interconnects needs thought.

Energy proportional interconnects. The application behavior is likely to be bursty with computation periods and idle periods. The on-chip network that can adapt to these phases and expend proportional energy. Mechanisms which enable fine grained power gating, near-threshold operation of interconnects with light-weight end to end error/flow control, duty cycling, DVFS.

3. **What on-chip synchronization/communication primitives are needed to support both coherent shared memory and message passing style traffic in multi- and many-core systems?**
   Support for lock elision and transactional memory. Managing coherence domains.
   Interconnect fabrics that support barriers.

4. **What system architecture (memory hierarchy, computational resources, network-on-chip, communication protocol, etc.) is optimal for CPOM while meeting the application-level requirements?**
   Covered in answers to the other questions.
   I don’t think there is one answer to power management. There seems to be two themes: 1) energy proportional systems and 2) voltage scaling (DVFS). The first has been promoted by data center operators such as Google, and the latter by SOC types, e.g. designers of mobile phone platforms. It would be interesting to see if the two approaches can be reconciled.

5. **What advances are needed to support performance isolation (physical or virtual) in systems with shared resources?**
   Support for VMs. See also comment about disaggregated memory below.
6. 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?
Apart from things already mentioned that are somewhat CPU-centric.

Memory is going to become an increasing user of energy in cloud and enterprise computer systems. It is difficult to voltage scale (at least the technologies in use today). Reliability will become an issue with large memories.

Energy-proportional approaches that allocate memory during run-time. Disaggregated memory creates a sharable pool and supports this model.

Non-volatile memory is another memory opportunity that appears to have promise as we go forward to smaller technology nodes. It promises density, scaling and power advantages. I expect that many of the promising candidates will fall by the wayside, e.g. PCRAM looks like it will disappoint.

Today's technologies: DRAM and FLASH are will continue to improve for the mid-term.

7. What are the research issues in this area?
Implied in other answers.

8. What other questions should we be asking and answering?
Reliability will become more important for several reasons: process variability and system capacity are among the reasons.

3D technologies?
Can't think of any more at the moment, but I'm sure we'll uncover many in the workshop.
Heterogeneous Parallelism: Power/Performance Opportunities and Techniques

Margaret Martonosi
Princeton University
mrm@princeton.edu

Introduction and Motivation: In modern processors, power efficiency has become a crucial design constraint. This requirement of power efficiency spans the spectrum of computing, from the mobile and embedded domain through servers and even to data centers. For many recent years, dynamic voltage and frequency scaling (DVFS) was a primary technique for managing the power-performance tradeoffs that arise in these systems. As supply voltages drop, however, the leverage of DVFS is being reduced, and new techniques are needed. One promising emerging solution is heterogeneous parallelism.

In some cases, heterogeneous parallelism refers to a collection of processors sharing a common instruction set architecture (ISA) but distinct in their processor/performance characteristics. In other cases, the heterogeneity is more extreme—mixing processors or devices of different ISAs, specialized accelerators or field-programmable gate arrays (FPGAs). Across many different heterogeneous platforms, the common characteristic is that they allow a system to balance its workload among the available processing units according to its needs, for example by temporarily giving up some performance to gain power efficiency, or vice versa. Most embedded systems display both on-chip and cross-device heterogeneity.

Key challenges in heterogeneous embedded systems fall into several topic areas, each of which are discussed below.

Systems Mapping and Scheduling: Heterogeneity often involves chips that include processing units with different instruction set architectures (ISAs). In many cases, the programming abstractions can be quite distinct in terms of their granularity, degree of parallelism, and execution model. Consider, for example, chips that include general-purpose CPUs, one or more GPUs, and additional specialized functional units for cryptography, signal processing, or other purposes. Considerable research questions exist regarding what is the best way to program such systems. Portable abstraction layers such as OpenCL are one possibility, and others argue that we should, as much as possible, exploit automatic parallelization from legacy codes. While the jury is still out regarding how to program heterogeneous systems, I argue that we need to devise and exploit portable intermediate layers that can be viewed as the compilation target from different programming approaches. Just as ISAs have formed a stable abstraction layer that allows software to outlast individual microarchitectural innovations, the new crop of heterogeneous multi-cores will require higher-level abstraction layers that form a long-lasting target for compilers and software, while allowing efficient task scheduling and mapping to cores from a system and hardware perspective.

Data sharing and inter-core communication issues: Data transfers between cores can incur considerable power and performance overhead. First, complex memory hierarchies often require multiple data copies to stage operands for computing. Second, communicating large streams of data in small granularities (e.g. cache lines) can also contribute to power and performance overhead. Third, communication delays due to staging overhead or contention can cause load imbalances between computational cores, which also wastes time and energy due to barrier waits and synchronization. I see a need to rethink systems layers in order to make data communication (alongside data calculations) a
first-class citizen in heterogeneous multicores, from high-level software, through compilers and operating systems, and down to hardware hypervisors and execution units.

**Performance Prediction and Regression Modeling:** The architecture research community has traditionally relied heavily on detailed simulations to guide design decisions. As designs become more complex and require ever-longer simulation times however, design space exploration based on repeated simulations has become more and more time-consuming. Simulation times are even more burdensome when one must consider highly-parallel multi-core executions, and when one must handle different instruction set architectures or other aspects of heterogeneity. For our field to progress, we will need to consider efficient early-stage methodologies for estimating and modeling performance of heterogeneous systems, and for composing together estimates from heterogeneous building blocks.
Position Statement
Pradip Bose
IBM T. J. Watson Research Center
Yorktown Heights, NY

Answers to some of the questions requested:

- **What micro-architecture is needed for effective implementation and exploitation of CPOM?**

  In the multi/many-core era, it is essential that we architect the chip-level (micro)-architecture in such a manner that at least one of the active threads is allocated the task of power management. In industry, we have seen the emergence of an era where on-chip (or on-board) power management controllers are becoming commonplace (Intel, AMD, IBM…). Whether these should be architected as “out-of-band”, dedicated sub-cores (or microcontrollers) with a firmware-OS-hypervisor system management stack, or whether “in-band” software power management threads (or daemons) scheduled on available hardware cores is the right answer for future scalability: remains an active research area in my opinion. There could, of course be an optimal solution in between these two extremes as well. Innovative hardware support to facilitate software-aided power management is the key idea I would like to stress in this context.

  The trend towards many simpler cores (which in some cases could be viewed or used as accelerators – *programmable or not*) mixed in with a few strong cores is inevitable – at least in some markets where demand for strong single-thread performance is unabated. The simpler cores would be used, of course, as accelerators for parallel throughput-oriented code; but they could also be used to provide support as assist threads for the strong cores. Could we then just rely on homogeneous many-core designs, where some of these cores could be used to assist others in order to accelerate single-threads (when needed)? In some sense that would be the best solution, from the point of view of simplicity, system software support and development cost. But, we are not there yet! Micro-architectural research in the areas of “core fusion” and related concepts, along with software assist threads continue to be very important topics for investment in my opinion. Such research aims at the delivery of CPOM via synergistic interplay between compilers, OS/hypervisor and hardware micro-architecture.

- **What system architecture (memory hierarchy, computational resources, network-on-chip, communication protocol, etc.) is optimal for CPOM while meeting the application-level requirements?**

  *Optimal* (or near-optimal) balance across the stated components of the system is obviously application-dependent. Therefore, adaptive system architectures that are able to

---

1 The views and opinions expressed by the author are his own. They do not necessarily represent the views of the author’s department within IBM Research or IBM Corporation as a whole.
dynamically change the “nominally architected” balance are a key aspect of any CPOM paradigm. And, to effect such optimal, application-dependent choices, one would need to architect hierarchical power-performance management systems: from within cores and chips all the way up through to system-level interconnects, memory and disk. Therefore, research on hierarchical sense-and-control loop control system architectures (as an integral part of regular chip micro-architectures) should be a top priority, in my opinion. The problems associated with such control systems – e.g. issues related to stability, overshoot/undershoot, damping characteristics, general pre-silicon verification challenges and hardware security issues point to cross-discipline research that is not sufficiently stressed today. (We addressed some of these issues and solution approaches in our ISCA-2011 tutorial on Energy-Secure System Architectures; to be offered again in 2012). In terms of the actual hardware support issues in this context, I would stress the following: malleable (partitionable) cache memory hierarchy, adaptive (resizable or reconfigurable) core and chip buffers, on-chip VRMs (voltage domains) with core-level DVFS or power-gating support.

• What advances are needed to support performance isolation (physical or virtual) in systems with shared resources?

Again, malleable shared cache architectures are a key example. Threads or VMs that run against other threads or VMs, must be allocated varying amount of cache resources, depending on the particular workload combination – in order to achieve the goal of performance isolation. Here again, we see the implicit role of on-chip/on-system power-performance control system architectures.

• 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?

Hardware support for some of the traditional system runtime software elements is a key requirement in future CPOM systems. For example, micro-architectural support to enable fast time-base (OS) resynchronization on core wake-up is an enabler of dynamic power gating of cores with minimal performance overhead. Also, on-chip hardware implementation of runtime queues (as in Intel’s Carbon project, Kumar et al. ISCA-2007) is another general example. Providing hardware-software support for fast core-level power gating, core restart and task scheduling is a key enabler of energy-proportional computing.

• What other research issues? What other questions to be addressed?

Circuit-microarchitecture co-design is an important research issue within the hardware layer of the CPOM concept. Predictive power gating is an illustrative example in this category. Here, idle duration predictors work in conjunction with circuit-level power gating support to provide leakage power control. Early work by Zhu, Kaxiras and Martonosi (cache line decay, ISCA01) and Flautner et al. (drowsy caches, ISCA02) have been followed by other fine-grain, unit-level and coarse-grain, core-level predictive power gating work (e.g. Zhu et al. ISLPED04, Lungu et al., ISLPED09, Madan et al., HPCA-2011 etc). More research investment needed in this general area.
1 Introduction

In spite of the huge advances in computing, computing till now has been used in a limited number of domains, such as automating repetitive tasks to improve efficiency, personal entertainment and scientific computing. There are vast swaths of society that are still unperturbed by computing; and, this is where, in my view, the future of computing lies. Computing must interact with physical world to solve some of the grand challenges facing the society, such as personalized health care delivery, enhancing infrastructure efficiency. To enable this vision there will be three layers of computing that interface digital and physical worlds. At one end of the spectrum there will be plethora of mobile field-deployed computing and sensing nodes that continuously monitor our environment. Environment is broadly defined as both our physical surroundings as well as our physiological state. The information may be partially processed on these mobile devices and in many cases the data is then transmitted over the network to a datacenter. Datacenters provide the computational resources that are necessary to mine useful information from the volumes of data being collected by the in-field sensing and computing nodes.

1.1 New challenges in mobile computing

In order to diffuse computing into all aspects of our living, the two ends of the computing spectrum must overcome energy inefficiencies in current implementations. While data center energy efficiency concerns have received significant attention in the past few years, mobile systems’ energy efficiency has only received marginal attention, particularly from computer systems researchers. Take for...
instance the promise of people-centric sensing using mobile phones. By harnessing the power of billions of mobile devices it is possible to gain unprecedented visibility into our social and environmental context. But unfortunately, these devices have limited battery capacity. Hence our ability to sense, compute and communicate will be limited by how much battery will be consumed by context monitoring.

There are significant differences between desktop-style computing nodes and those that are used in mobile computing. (1) In desktop systems CPU, GPU, memory and I/O are the primary energy consumers. A mobile device on the other hand is a collection of a vast number of IP-blocks that are integrated from a variety of, potentially competing, vendors. In addition to the traditional CPU, GPU, memory, there are non-volatile storage elements, several radios, GPS, cameras, audio recorders, and a plethora of sensors for sensing light, temperature, air pressure, accelerometers and gyroscopes. There is no single component that dominates energy consumption. (2) Backwards compatibility is not as critical as it has been in desktop computing. Due to faster refresh cycles, coupled with an ecosystem built around automatic software updates, the notion of installed software base is not as applicable to this domain.

1.2 New Opportunities

Given these differences, there is a need for CPOM research at the architecture layer that embraces all IP blocks as equal partners, not just CPU, memory and disks. Even simple power models that incorporate and measure approximate power of the entire system will become more valuable than highly detailed single component power model. New modeling abstractions for display, communication and sensing circuits will become important to evaluate the overall system design. For instance, a parameterized power model for a camera sensor can help designers with the cost-benefit tradeoffs of increasing camera resolution versus increasing display resolution in a next generation mobile device. Given the importance of continuous network connectivity in mobile environment, there will be many wireless radios integrated on the device. These network interface provide heterogeneous bandwidth, power and latency tradeoffs. These tradeoffs can not be statically determined since the wireless channel behavior varies dynamically. Modeling these dynamic tradeoffs will become necessary for building energy efficient runtime systems.

The software market for mobile applications is radically different than desktop software. Given the very low entry barrier for mobile application development there are many application developers who focus primarily on application functionality. At each stage of a mobile application design and operation, the application developer has a wide range of options to trade-off battery life of a mobile phone for other important metrics. For instance, location can be sensed in multiple ways that tradeoff location precision with energy efficiency. Context detection, such as user state, can require throttling down or ramping up sensor sampling rates. Similarly the application developer has to make energy tradeoff decisions on computation and communication costs. Finally, energy efficient
operation point itself drifts due to changing external conditions, such as time of the day and location. Given the daunting list of tradeoffs that dynamically vary, it is imperative to provide a mobile application development framework where the developer focuses primarily on functionality, while new frameworks implement sensing, computation and communication services in an energy efficient manner. The underlying frameworks will be hardware-assisted runtime systems that can continuously monitor power consumption and take dynamic decisions to improve overall battery efficiency.

Mobile computing paradigm also requires a shift in benchmarking techniques. Benchmarks such as SPEC-CPU are irrelevant in evaluating mobile designs. New application domains, such as speech to text, picture-based search will become the dominant consumers of network and compute resources. Hence, new benchmarks must capture a broader range of usage models of these devices and must also be updated much more frequently than traditional desktop computing. Traditionally benchmarking was done based on industry segment: CPU2006, SpecWeb, transaction processing. In mobile computing there is a need to tailor benchmarks based on user models. For instance, we may have to rely on well-known models for innovation diffusion that classifies users as innovators, early adopters, early majority, late majority. It has been shown that each of this user base has very different usage models and hence stress different system components differently. Irrespective of the way we may choose to categorize usage models, what is clear is that there is a need for a fundamental shift in how we benchmark the energy consumption and performance tradeoffs in these systems.
Dark Silicon Aware Architecture

Prof. Michael B. Taylor
University of California, San Diego

Until recently, the dominant paradigm for future technology scaling has been Moore’s Law. Recently, poor CMOS scaling has led to a utilization wall [VSG+10] which limits the benefits that we can achieve due to Moore’s Law.

Utilization wall: With each generation of Moore’s Law, the percentage of a chip that can switch at full frequency drops exponentially due to power constraints.

A direct consequence of this utilization is the rise of dark silicon – silicon area that must be left passive, or under-clocked in order to stay within the chip’s power budget. Today, only a few percentage of a chip can be active at one time within a chip’s power-budget. Dark silicon and the utilization wall has sweeping ramifications for how we will build machines going into the future. We need to conceive systems for which only a tiny fraction of the silicon is active at any one time, and the remaining area is power-gated. Our key tool going into the future is specialization, which can reduce the energy of a computation by 10-100× versus an efficient general-purpose processor.

One such design paradigm is to build the chip as a massive collection of highly-specialized circuits that use silicon area to achieve 10-100× improvements in energy efficiency versus energy-efficient, general-purpose cores [VSG+10, SVGH+11]. Execution hops from specialized core to specialized core, using the design that is most suited for the computation at hand. Unused cores are power gated.

To address these challenges, we need new approaches to generating specialized cores; to reduce them from highly labor intensive processes to largely automated processes. We need to have ways of influence the operation of these cores to track software evolution, and ways of instrumenting and measuring power consumption at fine granularity in order to understand power usage with the same resolution that we currently can analyze performance. We also need ways of virtualizing specialized cores and accelerators; to allow the system to maintain power benefits even if the number of threads requested is higher than the number of available accelerators.

In the dark silicon regime, we should expect to have large swaths of chip area that are power-gated. The network requirements of such a design are very different from the traditional multiprocessor networks and their more recent cousins, the on-chip network. While those networks are routinely evaluated assuming extremely high loads, and implicitly mimic dense metropolitan networks like in NYC, on-chip networks for dark silicon will look more like interstate highways crossing empty expanses of the MidWest. Latency is important, but congestion is low and energy costs are the primary issue. Here, a major scalability issue is in the power-consumption of the networks. Leakage is a problem since the networks will not scale down as transistors shrink – they will need to connect more and more things across long expanses of the chip. Active power is problematic if we use designs for which the number of routers crossed increases as we connect more and more power-gated cores to the system. To solve these issues, we need mostly power-gated (MPG) networks whose links can be activated to form a dynamic topology that adjusts according to what is lit up, and power-gated when the links are unused. To do this without incurring the latency costs is a key challenge.

For power reasons, we also need to consider how to localize structures in the system. Shared memory is a key programming simplification that allows programmers to avoid deadlocks due to limited network buffer space. However, we need to have shared memory implementations that are focused on locality to avoid large power costs due to cross-chip traversals to access directory structures. For example, we should consider hierarchical directories that cluster coherence traffic close to the nodes that are initiating it. Furthermore, we should consider the concept of binding home caches and directories via TLB to particular regions of a chip (for instance, as done in Remote Store Programming.) This enables more efficient control
over the movement of data, and thus is a key way to eliminate hidden energy costs of shared memory systems.

Performance and energy consumption are linked, even in a post-voltage scaling world. To the extent that software can inform the hardware that the deadline for a task is far away, we can achieve more efficiency by using vast simpler processors (ala Stanford ELM) to do the task. The challenge is that currently the hardware has few mechanisms for understanding the criticality of tasks. We need lightweight mechanisms for software to histogram the slack in execution time, and to pass this information to a hardware scheduler that can select less aggressive hardware to execute the code.

Finally, we must leverage through silicon via (TSV) and die-stacking technology in order to address today’s DRAM energy non-proportionality. Current DRAMs expend orders of magnitude more energy than is necessary, and much of this is because of the need to signal through packages and out over PCB traces. After DRAM traffic is moved to TSVs, all that will be need for I/O is a few “low-bandwidth” 100G data links for video, networking, and disk traffic.

References


CPOM Workshop Position Statement

Sudhanya Gurumurthi
University of Virginia

The memory hierarchy is a significant power consumer. I believe that achieving the dual goals of high performance and low power will require the use of emerging Non-Volatile Memory (NVM) technologies, such as Phase Change Memory (PCM) and Spin Transfer Torque RAM (STT-RAM), in the memory hierarchy. Both PCM and STT-RAM are inherently low-power (there is no leakage in the memory elements themselves), provide high-density through a combination of small cell sizes and multiple bits-per-cell capability, and have access latencies that are comparable to SRAM and DRAM (by STT-RAM and PCM respectively). While PCM suffers from limited write endurance, its endurance is significantly higher than Flash memory. The ITRS Roadmap also predicts better scalability for PCM compared to Flash, making PCM a candidate for use in future Solid State Disks (SSDs). Recent research in the architecture community has demonstrated how the reliability and security problems associated with the limited write endurance of PCM can be addressed for main memory using techniques such as wear-leveling. The high endurance and faster access times of STT-RAM makes this NVM suitable for use in processor caches. Recent research has also demonstrated that the high write latency associated with both PCM and STT-RAM can be reduced, for example, using DRAM-based buffering techniques for PCM and reduced non-volatility designs for STT-RAM. While developing the memory technology has been the focus of the devices and circuits communities in both academia and industry, there are several “Microarchitecture-to-Systems” level research questions that need to be addressed in order to fully leverage the benefits that these technologies offer. These include:

- **What should be the right mix of technologies to use at each layer of the memory hierarchy?**
  Should all the caches be replaced with STT-RAM, hence achieving the greatest leakage power reduction, or should the L1 cache be retained in SRAM to achieve a higher performance? What is the mix of DRAM and PCM to use for main memory? Should DRAM be used only as a buffer for PCM or should the physical address space be partitioned between the two? Given the blurring between memory and storage that NVMs create, should we actually have a shallower memory hierarchy?

- **How do we manage an NVM-based memory hierarchy?**
  The high density of NVMs facilitates the design of larger on-chip caches and a deeper memory hierarchy. Should such a hierarchy be hardware-managed, software-managed, or some combination thereof? How do we place, move, and search for data in such a large hierarchy? Recent research on STT-RAM (from UVA, NYU-Poly, Qualcomm, and others) has shown the ability to tradeoff non-volatility for performance and energy efficiency, thereby making non-volatility a “knob”. How do we manage a memory hierarchy where each layer has a different retention time?

- **How Does Memory System Non-Volatility Affect Systems Design?**
  The presence of non-volatility closer to the processor blurs the distinction between memory and storage. It is important to explore how this new property of the memory hierarchy can impact systems design. For example, non-volatility in the caches can facilitate an “instant on” capability, which can allow for more active low-power server states. Recent work has looked into the implications of this property on various aspects of system design, including file systems (BPFS
from Microsoft Research) and memory interfaces (NV-Heaps from UCSD, Mnemosyne from Wisconsin, and CDDS from HP Labs). More research is needed in this area to assess the benefits and challenges that memory hierarchy non-volatility poses to systems design.

- **How Does the Use of Emerging NVMs in the Storage System Impact the Software Stack?**
  Current storage software stacks are designed under the assumption that disks are slow. The use of NVMs, whose access times are orders of magnitude faster than disks and even Flash memory, and with different access semantics than disks or Flash (e.g., byte addressability) mandate rethinking the design of the storage software layers. As a case in point, recent research from the Moneta and Onyx PCM-based SSD prototyping projects at UCSD have demonstrated that migrating the underlying storage medium of SSDs from Flash to PCM can cause serious software bottlenecks that need to be addressed by taking a cross-layer approach of co-designing the hardware and the software. Further research along these lines is necessary.
As we attempt to conceive the highly-integrated many-core of the end of the decade, it is clear that the architecture has to be designed from the ground up for power and energy efficiency. We expect that such 1K-core chip will be designed with low-voltage devices, possibly running at near-threshold voltage (NTV). Moreover, since data movement/communication will be much more energy-intensive than computation, we expect that the architecture will try to minimize data movement at all costs. A research agenda for computer architects that want to design such chips includes the following:

* **Architectural techniques to manage and tolerate process variation at NTV**
  Operation at NTV will result in large variations in static power and frequency across different regions in the chip. To tolerate such variation, the chip needs to be organized in clusters, and support multiple frequency and voltage domains. However, voltage regulators consume power and crossing frequency domains takes time.

* **Architectural techniques to perform power management**
  Most power dissipated will be static power, and will be dissipated by on-chip storage elements. We need multiple power-gating modes, each with different latency and leakage-elimination capabilities. We will have to save the state from power-gated storage elements. We need to study the granularity of power-gating, since it impacts its latency, effectiveness, and power losses. We have to support both hardware and software activation modes, each with its own advantages and shortcomings.

* **Simple cores for energy efficiency**
  Cores will necessarily be simple and include heterogeneity. We need to understand the optimal level of heterogeneity. Also, some cores will be close to the memory system or off-chip ports, and can perform processing-in-memory operations. These are memory-intensive, relatively simple operations such as copies or simple transformations on large data chunks.

* **Memory hierarchy organization for minimal data transfer**
  Since data transfer is highly costly in energy, we need to design an on-chip memory hierarchy that minimizes unnecessary data movement. This suggests moving away from hardware cache coherence and using instead software-managed caches and scratch pads, which provide a high degree of control on the data moved.

* **Efficient synchronizations**
  The presence of so many cores implies the use of efficient, fine-grain synchronization. Synchronization should include aggressive primitives such as full-empty bits or multi-location synchronizations.

* **Block operations**
  Multiple data transfers can be bundled together into block transfers for high energy-efficiency. In addition, these block operations can perform some computation as they transfer data. The goal is to minimize data transfer overhead and overlap computation with communication.

* **Efficient networks**
  In addition to having a high-bandwidth, low-latency network, we need special networks for barrier synchronization and to perform reductions and broadcasts. They provide significantly efficiency.

* **Resiliency though efficient check-pointing**
  Power- and energy-efficient check-pointing is necessary to recover from the many faults that such a chip may suffer, resulting from process-variation, aggressive power management, and other reasons. We need something like ReVive, which performs in-memory, incremental check-pointing. Such check-pointing may be driven by the compiler if it understands the application well.
Some Ideas and Principles for Achieving Higher System Energy Efficiency: Slack-Based, Coordinated, Hardware/Software Cooperative Management of Heterogeneous Resources
Onur Mutlu, Carnegie Mellon University (onur@cmu.edu, 412-268-1186, http://www.ece.cmu.edu/~omutlu)

This whitepaper briefly describes some broad ideas, principles, and research directions that seem promising, in the author’s opinion, to achieve a system that is overall more energy efficient (and also higher performance) than today’s systems.

Energy is performance - be frugal: In today’s systems, which are overwhelmingly power-limited, energy is performance. If one can do the same “task” with less energy, one can do more of the same task or other tasks by spending the same amount of energy as before, increasing system throughput[1]. Therefore, doing the same task at the same performance but more efficiently is key to improving system performance as well as energy. Hence, we should design building blocks (e.g., functional units, cores, caches, libraries, software components) that are as simple and efficient as possible without significantly sacrificing performance.

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[5]. Not all network packets or memory requests have the same criticality[2][3]. Not all cache blocks are equally important[1]. Not all threads limit performance[1][4][6]. Not all queries/requests in a server need to be serviced fast. Not all threads or programs are of the same importance. Efficient systems should identify criticality and bottlenecks at different levels of processing (instruction, request, task, program, system) 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.

Consolidate many tasks on shared hardware resources while providing the QoS each task needs: Efficient utilization of hardware resources can be achieved by consolidating more tasks on the same hardware to exploit idleness of resources. However, consolidation leads to contention and potential starvation, leading to inefficiency. Therefore, efficient systems should incorporate mechanisms that provide QoS to different tasks. A unifying principle to designing such mechanisms is the identification the slack/criticality/latency-sensitivity each task has and allocating more resources to those tasks that have little slack. We have recently shown that 1) identifying the slack of network-on-chip packets and prioritizing packets with lower slack[2], 2) identifying limiter threads in parallel applications using cooperation between the hardware and the runtime system and prioritizing such threads in the memory controller[4], 3) identifying latency-sensitive applications and prioritizing them in the memory scheduler[7] can yield significant performance improvements. We believe these techniques can also improve energy. Many other such opportunities exist to manage resources using the concepts of slack, criticality, and latency-sensitivity.

Coordinate the management of shared resources: If one resource prioritizes a critical task while another deprioritizes it, system efficiency degrades. Similarly, if one resource implements a technique that is aimed to improve performance/efficiency (e.g., the core generates multiple concurrent memory requests assuming they will be serviced in parallel by the memory system) while another employs policies that destroys the benefits of the technique (e.g., the memory controller serializes the memory requests[10]), system efficiency degrades. For best efficiency, therefore, different resources should work in tandem with each other: policies employed in cores, caches, interconnects, and memory controllers should be co-designed or aware of each other. As we have shown in recent work, designing the interconnect packet scheduling policy[2] and memory controller scheduling policy[10][8] such that they are aware of the cores’ policies, and designing the last-level cache policies such that they are aware of memory controller policies[9] can significantly improve performance. We believe significant more potential exists in coordinated optimization of policies across resources.

Exploit architectural and technology heterogeneity: Heterogeneous or partially reconfigurable components (not only processing elements, but also shared resources with different power/performance tradeoffs and fitting different computation/access/communication patterns/needs) are essential to enable power-efficiency using automatic resource management. If components are the same, energy and performance of a task will be suboptimal regardless of on what component a task is executed. Heterogeneous components enable customization opportunities such that a task can be executed on the best-fit resource, i.e., one that executes the task most efficiently while satisfying its performance requirements. We posit that not only cores, but also interconnects, caches, memory controllers, and entire memory hierarchies can be designed in a heterogeneous manner and the runtime system can be designed to automatically choose the best-fit components for each task dynamically. Management of heterogeneity 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, as we have shown in recent work[12][9].

Coordinate hardware and software: To accomplish all of the above efficiently, we strongly believe hardware/software cooperation is essential. Work should be divided across the layers such that the overall system is the most efficient. Examples abound in both our and others’ past works. Many future research opportunities exist in all of the above directions.


1A “task” can be thought of an arbitrary unit of work, for the purposes of this whitepaper. It can be an instruction, a memory request, a function, a thread, a collection of threads, an entire application, a collection of applications, etc. We believe the principles outlined to improve energy efficiency is fundamental across these different entities, which we will refer to as tasks.
Position Statement for the NSF Cross-layer Power Optimization and Management Workshop

Jeff Draper

1. What micro-architecture is needed for effective implementation?

A scheme that is able to adaptively scream with high throughput, high performance in an energy-efficient manner when needed while also being able to dissipate close to 0 energy when idle. This probably requires a mix of lightweight and heavyweight cores with ample instrumentation for both hardware/software to work together to make decisions about when to adapt modes for the most energy efficient operation. It also implies true dynamic voltage frequency scaling.

2. How does the global interconnect look like in systems with a large number of diverse and interacting IP blocks so as to power-efficiently support the required diverse traffic patterns?

Before addressing what the interconnect should look like, I think it must first be noted that serious effort needs to be applied across all layers of the system, especially at the algorithm and application levels, to minimize data movement. Data movement has been the real killer in terms of energy dissipation. Without some effort to better exploit locality I’m not hopeful that we can adequately address the energy problems, regardless of what interconnect is used.

Having said that, I think the most crucial characteristic needed for the global interconnect is similar to the micro-architecture. It needs to be able to perform in an energy proportional way. It is well known that for many applications, communication is bursty. When the network is not busy, there needs to be some mechanism to shut down many links so that energy is saved. This implies fast turn-on and turn-off mechanisms as well as intelligence to recognize when an idle period is starting.

3. What on-chip synchronization/communication primitives are needed to support both coherent shared memory and message passing style traffic in multi- and many-core systems?

This is a key question. The big challenge for systems today is not only how to provide energy-efficient mechanisms for synchronization but also high-performance. It’s a very difficult problem, especially in the since of barrier synchronization mechanisms, because by their very nature, they involve many participants if not a global set. How do you make something energy efficient that involves the whole machine? The best answer I can provide is to minimize overhead. The more energy-efficient autonomous hardware mechanisms that need little software oversight, i.e., overhead, we can throw at such problems, the better. It may be time to seriously consider full/empty bits, at least on some memory locations. True atomic memory operations will probably be a key component of any solution. Transactional memory concepts may come into play but we have to find more efficient ways to implement them.
Similarly there is a challenge for message passing operations. We need to find mechanisms for reducing overhead for dispatching/receiving messages while still providing protection needed in such resource sharing operations. This is a case where improving performance will also decrease the energy dissipated. While most messaging operations can be done at the hardware level in 10s or at worst 100s of ns, by the time software overheads are factored in, it is difficult to achieve sub-microsecond messaging.

4. **What system architecture (memory hierarchy, computational resources, network-on-chip, communication protocol, etc.) is optimal for CPOM while meeting the application-level requirements?**

As noted by others, I think the key aspect of any system that really addresses the CPOM challenge is dynamic adaptability. While some subcomponents of today’s systems provide some level of this, e.g., dynamic voltage-frequency scaling, the system as a whole has not been addressed holistically with this perspective in mind. The more we can adapt system resources to dissipate energy proportionally to the work being accomplished, the closer we will be to seriously addressing the CPOM challenge.

I think new technologies will come into play. The 3D memory stacks that Micron and others are working on may allow for removal of some levels of the memory hierarchy since the memory cubes promise a higher-bandwidth, lower-latency, lower-energy access to main memory. On the contrary, it looks like a resulting system using these memory cubes will have a lot less memory per node to fully exploit the bandwidth offered.

5. **What advances are needed to support performance isolation (physical or virtual) in systems with shared resources?**

I think the only-bullet proof way to ensure performance isolation may require overhead that is too burdensome to justify it. For each region of isolation to have some sort of quality-of-service target, and then for the system to have a background manager that tracks all these targets and instrumented system behavior against the targets sounds like a lot of overhead in a system which is attempting to minimize energy. Nonetheless, the analysis should be done to see whether such a scheme with its energy overhead could be used to not only guarantee performance isolation but also do it in a way that somehow reduces overall system energy---another area that seems rife with tension.

6. **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?**

I’ll go way out on a limb here and say that as we approach the end of Moore’s Law, we will have to once again consider domain-specific architectures. Similar to D. E. Shaw’s special-purpose Anton supercomputer that was tailored for molecular dynamics applications, I can imagine that we need to address cloud computing environments in their own domain. I believe there are a number of virtualization and protection issues that arise in cloud computing that are entirely different from other platforms, e.g., mobile computing. For any such domain, a close look should be taken at the most frequently used operations and see if
there is some way to move most aspects of such operations mostly into hardware to reduce the overhead. I think it is highly likely that solutions tailored for different domains may be the only way to achieve the 100GFLOP/w numbers needed to truly advance energy-efficient computing.

7. **What are the research issues in this area? What other questions should we be asking and answering?**

Many of the research areas are noted in previous answers. One of the key areas of concern is how to balance efforts at reducing energy with efforts at enhancing other system characteristics. With the increasing variability in transistor performance at aggressive technology nodes, and similarly the increasing susceptibility to soft errors, I am very mindful of resilience concerns. Most approaches at providing some level of resilience involve redundancy, which implies increased energy. Balancing energy efficiency and resilience will be a key challenge.