LOW-POWER DESIGN AND TEMPERATURE MANAGEMENT

Kevin Skadron
University of Virginia

Pradip Bose
IBM T.J. Watson Research Center

Kanad Ghose
State University of New York, Binghamton

Resit Sendag
University of Rhode Island

Joshua J. Yi
Freescale Semiconductor

Derek Chiou
University of Texas at Austin

ONE OF THE PRIMARY CONCERNS FOR MICROPROCESSOR DESIGNERS HAS ALWAYS BEEN BALANCING POWER AND THERMAL MANAGEMENT WHILE MINIMIZING PERFORMANCE LOSS. RATHER THAN GENERATE SOLUTIONS TO THIS DILEMMA, THE ADVENT OF MULTICORE CHIPS HAS RAISED A HOST OF NEW CHALLENGES. THIS DISCUSSION WITH PRADIP BOSE AND KANAD GHOSE, EXCERPTED FROM A 2007 CARD WORKSHOP PANEL, EXPLORES THE FUTURE OF LOW-POWER DESIGN AND TEMPERATURE MANAGEMENT.

Moderator's introduction: Kevin Skadron

In recent years, power dissipation has become an area of intense concern to the designers of microprocessors for various reasons. Today's processors require sophisticated and expensive thermal packages to control heat dissipation, and battery life is a perennial concern. Reducing power dissipation helps mitigate all these problems, although controlling temperature requires different low-power strategies than those used for energy efficiency. While circuit-level techniques have been a mainstay for managing power dissipation, architecture-level techniques promise additional and synergistic techniques for managing power because they can take advantage of additional knowledge about the current workload's runtime behavior. Unfortunately, most architecture-level power- and thermal-management techniques degrade performance because they turn off or slow down parts or all of the processor. The challenge, therefore, lies in finding power- and thermal-management techniques that minimize the performance loss. Additionally, variations in the semiconductor manufacturing process, which affect circuit speed and power (especially leakage), will increasingly complicate power and thermal management as well as reliability.

A further challenge is the advent of multiple cores on a chip. Power constraints are likely to limit the exponential growth in clock speeds that we have become accustomed to. Instead, Moore's law will increasingly be realized by the growing the number of cores or processing elements on a single chip. This raises a host of new questions, such as the number and types of cores—all of which designers must select to optimize energy and thermal efficiency.

The goal of this discussion (which originally occurred during a panel at the 2007 Workshop on Computer Architecture Research Directions) is to debate the future of low-power design and temperature management. Prior to the panel discussion, panelists Kanad Ghose from SUNY Binghamton and Pradip Bose from IBM agreed on several points, which were not to be debated in the panel:

- Power does matter.
- Multicore isn't the ultimate solution to the thermal problems; we still need to worry about cooling our chips.

Published by the IEEE Computer Society

- We need to invest in building new design tools because current tools aren't enough.
- We need to invest more effort in finding new benchmarks because we can't learn everything we need to from the Standard Performance Evaluation Corporation (SPEC) benchmarks.

This allowed the panel to focus instead on more pressing topics:

- How important is explicit hot-spot management in the multicore era, versus generally minimizing overall power dissipation?
- What is the importance of techniques explicitly focused on leakage management?
- How should we evaluate power and thermal management in multicore chips—detailed simulation or analytical models?
- Do we need dedicated controller cores for power/thermal management?

Highlights of the panel discussion follow the two panelists' position statements.

Position statement: Kanad Ghose

As Yale Patt said, industry did not try to solve the hard problems for improving single-core performance. [*See "Single-Threaded vs. Multithreaded: Where Should We Focus?" in this issue.—ed.*] Going multicore was an easy way to get the product out the door quickly and to keep the customers happy. And now, multicore processors are here to stay.

Buses aren't so bad

My first claim is that applications are unlikely to make use of more than four to eight homogeneous general-purpose cores in a chip for desktops and laptops. Therefore, there is little value in designing complex interconnection networks. Maybe it's better to keep buses at this level.

Buses have some advantages that other interconnects, such as a packet-switched mesh or wormhole-routed mesh, can't provide. For example, buses provide atomicity; cache coherence protocols for buses aren't only easier to design and implement, they are also easier to verify.

Also, because wire lengths on a chip are getting smaller, we can borrow some of the techniques that are used in memory design today, such as differential sense amplifiers, twisted bitlines, and multistage sense amplifiers, to make buses that can span the chip's dimensions. We can use spanning buses, which are like a set of buses in each of the two dimensions, or we can use hierarchical buses and so on. These are probably going to be a lot more energy efficient, because they are much simpler: They don't have routers, they don't have tables that you have to look up to figure out which way you are going, and they don't have congestion control. There is value in simplicity. Buses are like the RISC of interconnection networks, so we shouldn't give up on them.

Power in the on-chip memory hierarchy

Architects need to explicitly focus on addressing power in the on-chip memory hierarchy of multicore designs. Although computer architects have been looking at caches for 35 years or so, there's still a memory wall. That tells us that something else must be done. For example, we may use on-chip buffers (not megabytes of buffers but smaller buffers) that can be used to do some kind of smart prefetching. For instance, dual-core Intel products have prefetchers in between the cache levels. If computer architects can extend that one more level to the memory, then we can have buffers for bulk commit, transactional memory, and so forth, which might be more energy efficient.

Average power is as important as peak power

Although throttling down activity in response to peak temperatures and power is important, limiting the average temperature and power dissipation is just as important. Peak power is the outcome of

About this article

Derek Chiou, Resit Sendag, and Josh Yi conceived and organized the 2007 CARD Workshop and transcribed, organized, and edited this article based on the panel discussion. Video and audio of this and the other panels can be found at http://www.ele.uri.edu/CARD/.

a poor implementation or a bad microarchitecture, which just heats things up and slows things down. So, instead, why don't we design things that are inherently hot spot free? To distribute the heat, we can spread the transistors out. Because average power and not peak power determines whether a circuit is good, we must address the problem where it really matters.

Need for tools

Without accurate, integrated tools for studying power and performance trade-offs for multicore systems, not much progress can be made. Efficient design tools are a requirement. In the same vein, what is a realistic workload for multicore systems? This question needs additional research.

Multicores: What's changed this time around?

Although people are currently talking about hundreds to thousands of cores on a single chip, what is the point of the second coming of multicore systems? Writing a thousand-core application is just as difficult today as it was 15 years ago. Can we really have an effective parallel-programming library? How poor will the energy efficiency be? How about the program debugging challenges?

On the other hand, I think it's quite possible to design a multicore system with x number of small processors and y Pentiums as Yale Patt has described. We can use heterogeneous cores to do all kinds of things. If we look at the way the consumer market is heading, there is some momentum to use some specialized cores to do things like graphics, speech processing, communications processing, compression, and decompression. We can now build supercomputers on a chip, and for this perhaps we need a thousand general-purpose cores or a fancy interconnection network. However, this type of processor will not be for the mass market, but more likely for a single DARPA project where you really need to push the envelope. Furthermore, Amdahl's law ultimately limits performance. It doesn't make much sense to have a highly parallel system if you don't have a lowly but fast single-threaded processor to work on the sequential bottlenecks.

Energy-efficient systems versus processors

In my opinion, designing a general-purpose, energy-efficient processor alone isn't enough. The focus should shift from designing energy-efficient processors to energy-efficient *systems*—a holistic approach that includes operating systems, libraries, protocol stacks, and compilers. Microarchitects should design simple energy/performance management hooks in the processor and all other components (RAMs, controllers, chipsets, and interconnections) that can be exposed to the software, beyond voltage and clock scaling. Subsystems must have power-performance ''gears'' to support power management at a higher level, and not simply by gating and controlling cores. Although inherently low-power components don't need to have adaptive management, global energy management is critical.

Position statement: Pradip Bose

Regardless of the number of cores in a chip, cores are the hot spots, and going from a single core to a multicore doesn't fundamentally change that. I think cores are still the most interesting and important parts of a multicore chip from the point of view of power dissipation. Although there is increasing concern about noncore elements (such as caches and on-chip interconnects), I still think that cores will remain the center of attention in optimizing chip microarchitectures from the perspective of power and thermal dissipation.

Mitigating thermal hot spots

I also believe microarchitecture research should emphasize mitigation of thermal hot spots (high-power density regions). There has been a tendency for computer architects working in this area to develop complex microarchitectural schemes for areas that are fundamentally low power (density), such as caches. However, we need an increased focus on distributing power more evenly across the chip, to reduce peak power density and temperature, by using both static means (such as better floorplanning) and dynamic means (such as thermal-aware

task migration and scheduling). Proposing inherently power-efficient (micro) architectures and functional units should come first, before considering additional leverage from dynamic reconfiguration of structures, especially for low-power-density structures.

At runtime, because dynamic task distribution is effective, we should pursue integrated hardware and software solutions to exploit this. Increasing the complexity of the hardware alone to improve dynamic power-efficient designs—brute force dynamic thermal management (DTM) research, for example—hasn't paid off very well.

Temperature control

Throttling on temperature threshold is just a damage-control emergency measure, and it shouldn't be used routinely because it clearly results in a large performance loss. Computer architects haven't done this trade-off analysis very well in terms of quantifying the performance cost of such proposed power or temperature control devices. Any microarchitecture-centric mechanism that promises x percent reduction in power, with y percent degradation of performance (where $y \geq x$), is suspect because the effect can be achieved by simply reducing the frequency. Furthermore, if I'm allowed to use the voltage and frequency knobs together in dynamic adaptation, the efficiency will be much better than most of the proposed microarchitectural techniques to manage power and temperature. The first job of (micro) architects should be to figure out how to make the microarchitecture efficient so as to reduce waste. For example, a smart instruction fetch policy that gates the fetch process and then flushes (or introduces selected stall points) during a detected period of high misspeculation can result in an excellent power-performance efficiency ratio.

Temperature versus power awareness

There's a difference between temperature awareness and power awareness. For example, consider two chips: One has a 25-W total power consumption, with just one localized hot spot resulting in a rather high peak temperature; the other has a 100-W total consumption but a uniformly distributed power profile that results in a much lower peak temperature. The top half of Figure 1 illustrates this example. Both are modeled to have exactly the same package and cooling solution, which means that by allowing four times the amount of total power, we can get much more performance out of the 100-W chip than the 25-W chip, but at the same package and cooling cost. So if we just worry about power, we might be tempted to go for the 25-W solution, which is the wrong solution because we would not get as much performance. Computer architects don't do well in situations like this because our integrated-analysis methods spanning microarchitecture, floorplanning, packaging, and so on still might not be up to the mark.

Consider a second example: A chip with a 50-W budget that currently consumes 60 W. Imagine that we could choose to save 10 W out of the lowest power density area versus 10 W out of the highest power density region. The bottom half of Figure 1 illustrates this example. Because the peak temperature will be significantly lowered in the latter case, it is a better choice.

Finally, if we consider power consumption in a two-core chip, dynamic activity migration from one core to another can reduce the average temperature significantly. If this is accomplished via temperature-aware scheduling at the system software level, the performance and hardware cost can be minimal. We can see the impact of core hopping by comparing Figures 2a and 2b with Figure 2c. Similarly, temperature-aware floorplanning can help a lot. Thus, by using both static and dynamic means, we can reduce the average temperatures, which in effect reduces the leakage, and therefore the power, and translates into better performance opportunities for a given power budget.

Process variability and noise

Process variability and noise will make statically efficient, reliable designs much harder. On-chip sensors, sensor networks, monitoring, and control of resources for "optimal" management of power, temper-

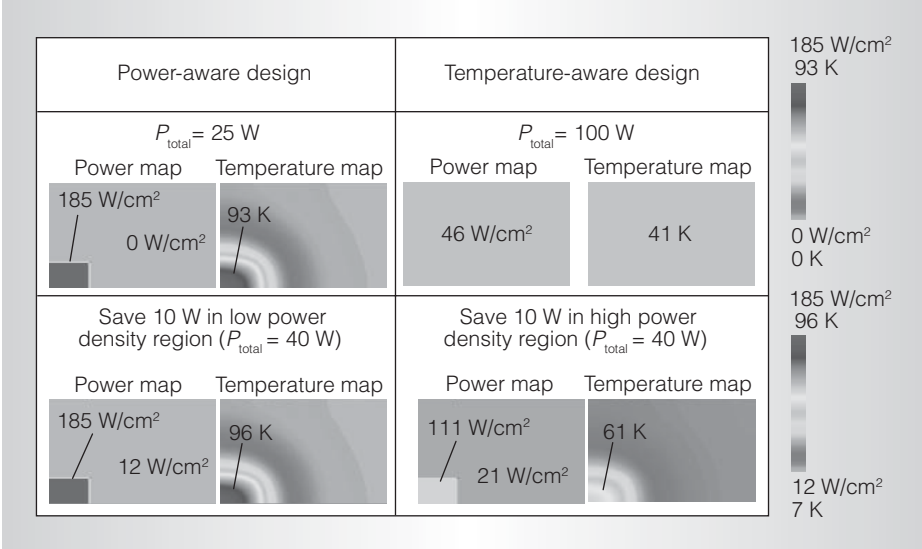

Figure 1. Temperature versus power awareness.[1] The temperatures shown are in terms of degrees Kelvin above room temperature.

ature, reliability, and performance create a multidimensional control and optimization problem that will inevitably point to the need for dedicated on-chip, programmable controllers. This presents many open research problems and issues in ensuring control stability, handling increasing levels of on-chip asynchrony, and so forth. Verification challenges will continue to pose the greatest obstacle to such microarchitectural trends—modular design principles will have to come to the rescue.

Need for models

We need a renewed emphasis on "fundamental understanding" models. Analytical or hybrid models can better guide microarchitects. I believe that this is more urgent and useful in early-stage definition and analysis work than developing either detailed full-system simulators or FPGA-based emulators.

Managing peak versus average temperature

Skadron: We've heard the importance of managing peak versus average temperature. Part of that depends on what kind of target systems you envision. Can you two elaborate?

Ghose: What is the best way of looking at thermal hot spots? Identifying them on a real microprocessor. The way we do it today is to have power models for data-path components, run a simulation at the microarchitecture level, collect the stats, figure out the energy dissipation, look at the thermal-flow equations, and figure out the temperature. Is that the best and most accurate way? My thesis is that if you know that there are hot spots in the design, you should try to make the design inherently hot spot free. We have design tools that help us do that. So, I think temperature-aware computing is a nonissue at this point if we have enough transistors.

Bose: I think that is incorrect. We have to have temperature awareness and power awareness at every stage of the game. The chip's hot-spot regions are the problem, and mitigation via more intelligent physical design tools alone won't solve it. We have to invest properly at the right places. If we invest effort into power efficiency in the hottest (power-dense) areas, that will pay off much more than if we invest into areas that don't have high power density. And since we know which architectural functions or resources are inherently "hotter," we can surely do a better job of designing thermally efficient microarchitectures. This will in turn lead to better power-performance efficiency.

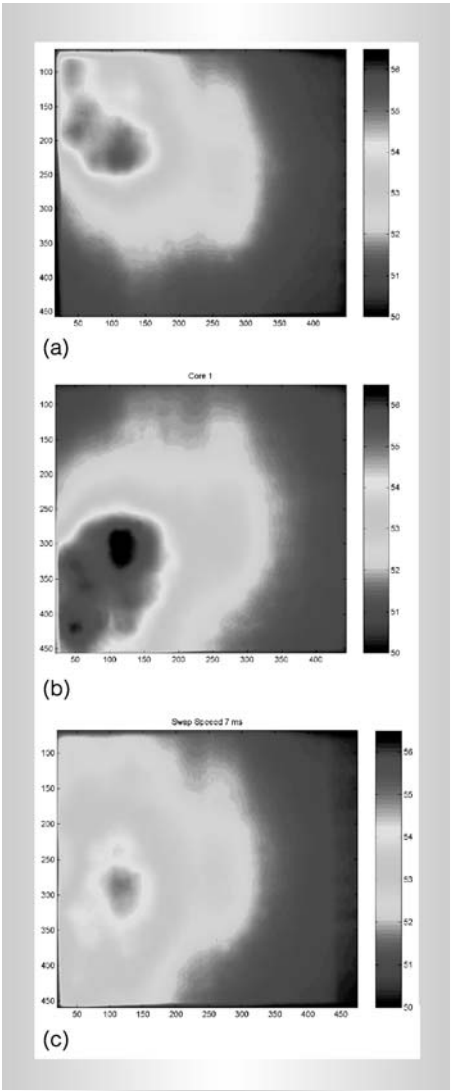

Figure 2. The impact of core-hopping on a processor's temperature profile.[2]

Skadron: So, just to clarify that a little bit, part of it comes down to whether you care about cooling budget versus just battery life. Assuming that we care about the actual costs of cooling the chip, that raises questions of how uniform the temperature must be to be good enough, and—in today's designs—the extent to which you can have high-activity structures like multiported register files. What kind of research questions does this raise?

Bose: Let's focus on one example, which is typical in today's processors: multithreading and specifically simultaneous multithreading (SMT), which is area efficient. For the same area, we can have many more threads running and therefore get better throughput performance per unit area. Yet, that fundamentally increases power density and therefore the core-localized hot-spot behavior. So, the paradigm of slow growth in the number of complex cores, with increasing degree of SMT per core, probably isn't a scalable approach to maintain a good balance between single-thread performance and chip throughput. Or is it? What's the right core complexity for such a paradigm to be scalable? These are some interesting microarchitectural research questions, I would think.

Ghose: Double-pumped arithmetic logic units and register files are examples of components that run very hot. Instead of double-pumped ALUs, we can use two floating-point (FP) ALUs. That's one easy solution. Register files are another story. Do we really need to follow the same path that we have been following for extracting instruction-level parallelism (ILP)? Are there better techniques? Why do we need to design a core that always does four-way issue? Why don't we design a core that morphs on two two-way cores instead of using one four-way core? There is a paper about core morphing in this year's International Symposium on Computer Architecture.[3] That might be a much more tractable approach. If we have that, we can use existing cluster-type designs, use smaller register files to reduce the register pressure, and so on.

Skadron: Let me follow up on that. The claim is that by using many smaller cores, you reduce these hot spots and your cooling problems. Is that something that Pradip would agree with?

Bose: If we believe in that paradigm, we also must believe that each core gets simpler. If cores get simpler, they will be spread out evenly on the chip. Then the temperature will spread out more evenly, and you will have better thermal performance efficiency. So, yes, I would generally agree with that, provided we can figure out how to maintain single-thread performance

through core fusing, assist threads, or whatever.

Ghose: As I said, morph the core: four-way to two two-way cores, depending on the ILP needs.

Bose: I don't think, though, that dynamic morphing of cores is guaranteed to pan out, because there are some really hard verification angles. If you have two morph modes, that doubles your verification effort; if you have four, that makes it 16 times greater, and so forth. And as we all know, verification takes up 70 percent of the total design budget already. So, even though it sounds nice, there are some practical impediments today to producing really exotic morphable architectures.

Audience member: How much of the problem is the question that you two are debating? You are talking about the hot spots between multiple cores. How about load-balancing problems, because one core or few cores are running and have all the tasks, and the others are not? So, you have hot spots as cores. The other problem is within a particular core, the ALU is a hot spot, the register file is a hot spot, and we aren't able to spread things out to reduce the power. But, there are two different problems: one is intracore, and the other is intercore. I'm not sure that I am hearing the same thing.

Bose: You said that they are two different problems, but viewed conceptually, they are the same problem, just applicable at different levels of abstraction.

Audience member: But their solutions are different. Circuit layout techniques can help you with the intracore problem. And perhaps, task management at the operating system level can help you, at the core level, with the intercore problem.

Ghose: I agree with that.

Bose: I think this is a good question that Kevin also raised: whether focusing inside the cores is going to be more important than focusing across the cores. I also agree. When you have more and more cores, the interest will shift more to the core-level granularity, so why worry about looking at things inside the core? But if you stick to Kanad's paradigm, saying that you will only have four cores, then each core still has to be made extremely efficient with exotic schemes like getting rid of hot spots inside the ALU and so forth. There is still some room for innovation there, in that case.

Impact of power and temperature on the number of cores and system-level design

Audience member: In the earlier panel ["Single-Threaded vs. Multithreaded: Where Should We Focus?"], Yale Patt and Mark Hill agreed that there would be many cores. However, Kanad said there would only be four to eight cores.

Ghose: For laptops and desktops. Servers can have hundreds of cores.

Audience member: Why do you believe that there wouldn't be many cores in embedded systems? A lot of embedded systems today have multiple cores.

Ghose: I agree that for those kinds of applications, you can have hundreds of cores. I have a phase-three slide that I couldn't get to into my presentation; that slide talks about a holistic approach to power management. What we are now trying to do is to design an efficient, smart hardware, and the rest of the software system is dumb. It doesn't make any sense. We have many things beyond the processor, the chipsets, the device controllers, and so forth. If we regard people designing these chips as microarchitects, a good avenue of research is to investigate things that might be done at the processor level, the level of device controllers, the level of memory chips, and so on, which the operating system and the rest of the software can look at to shift gears.

I can give you an example from an implementation that I've just completed. If you look at the Linux file system, ext3fs, buffer management can make a big difference on the power consumption introduced by the file system. So, if we have smart buffer allocation and management, and

schedule the I/O request cleverly, you can save 10 to 20 percent on the file access power for some of the large file-system benchmarks. So, there are things that can be done. The point that I'm trying to make is that it doesn't make sense anymore just to have a smart, power-efficient processor if the rest of the system isn't controlled in a holistic manner.

Bose: Let me add something. I'm not saying that the interconnection network isn't important. I'm just saying that for power-aware, temperature-aware research, that's not the primary focus. What are the really interesting areas from a power, temperature point of view? Obviously, you have to take care about total power; peak as well as average power; and so on. We all agree on this. But what I was saying is that if you focus on areas that are low power density fundamentally, and make the argument that you saved 15 to 20 percent power, when you then compute how much cooling cost you saved, it isn't going to be much. So, from an industrial, practical point of view, I would rather put my resources in taking away power from the hot-spot regions. Of course, it is also true that every watt I can save is important no matter from which part; everything adds up.

Alternatives to multicore processors for scaling the power wall

Audience member: I think we have a tough challenge in terms of power. Cooling solutions for mainstream laptops and desktops aren't going to improve: Devices are going to get smaller, which will be even more challenging, and our power budget isn't going to increase; it might even decrease. I think we need some radical approach. The only aggressive approach that I've seen so far is to have many cores, which makes sense, if our power budget isn't going to increase; we want to execute the same activity exercising a smaller part of your chip (smaller capacitance), which calls for simplicity. If we solve the problem of providing many simple cores, that is a good solution. Are there any other approaches—without using many cores or techniques that reduce power in register files or issue queues—that solve the big problem? What other solutions do we have without using many cores and confronting ourselves with this programming challenge?

Skadron: So, if we have concerns about the programmability of many cores, then what are the other alternatives for dealing with the power wall? I think most people probably agree that incremental tweaks to the microarchitecture of the existing core aren't the final answer.

Audience member: Just to add on to that, most people think that by using many simpler cores, we solve the problem. That's because they ignore the whole interconnection network problem. If we just continue to assume that many simpler cores are going to solve the problem without addressing interconnection network, even the many core chip solution won't work.

Bose: Let me ask a question. Which is the harder problem? Is the cooling problem likely to go away? Or is the programmability problem likely to go away if you put enough research or investment into it? I think it is the latter, because you can't fight physics. You can't cool beyond the certain limit. If you have to stay within that form factor and cool things, is there a solution?

Audience member: I think the solution is to find more energy-efficient schemes. For instance, another potential direction is special hardware, which is much more energy efficient. So, there is probably an energy-efficient solution to follow.

Bose: Yes, indeed that is an interesting idea. When we say multiple cores, nobody is saying these are all going to be the same homogeneous cores. There is probably going to be a basic skeleton core, and there are going to be programmable accelerators, which do simple functions in an extremely power-efficient manner, and you could in fact power-gate things off as you run the applications—that is, these accelerators turn on and off. I think that is a nice research paradigm to study.

Audience member: My point is that the challenge isn't thermal or power efficiency;

it is energy efficiency. I think this should be our focus—or, new ideas to help us accomplish whatever task we have without wasting energy as much as possible.

Systematic power-aware design

Audience member: For 60 years, we've mostly focused on performance, and we've never really talked about performance-aware design as sort of a subtopic of our field; it's just that's everything we do. So we are free to come up with any idea that we want and evaluate its performance. As architects have started embracing power- or thermal-aware design, it almost feels like this community has ghettoized into a smaller collection of modifications onto existing architectures. Maybe not as industry folks here who have a different time horizon, but as academics, we could say from a fundamental energy standpoint, where do we want to be? Maybe we want to have much coarser-grained control, where instead of doing management on the order of hundreds of instructions; we do graphs of computations rather than single instructions. So I'm wondering, in the short timeframe, how do we build a four- or eight-core chip that's sensible for today? But there's also this broader picture, where's the most promising fruit out there, in the bigger picture?

Ghose: So are you trying to advocate control over the macro level for power management?

Audience member: I'm advocating a broad view where we bring in almost anything. My sense is that macro-level control is a way of being more efficient because, instead of thinking about dependencies on the order of individual instructions, maybe we think of dependencies on the order of chunks of execution, and, similarly, think of management skills on coarser grains as well.

Ghose: The truth is out there somewhere. I think what you really need is some kind of entropy theory for computations and energy dissipation that says this is the minimum energy that you need to do this computation.

Bose: I think again that the fundamentals here might be missing. In computer architecture, we tend to invent structures, as [the audience member] was saying, and then we morph and do incremental updates on that depending on what's important in the current regime, like power or energy efficiency. We don't fundamentally rethink the algorithms, and this is probably one of the problems.

For example: Assume that there's a branch predictor with two tables and a selector. We never ask the question, if these two predictors are pretty much similar in terms of accuracy, do we need such complex hardware? Just double the size of one table and get rid of the other table and the selector! But we seldom do those trade-offs. I'm agreeing with [the audience member] in that sense. We sort of let things evolve and then say, ''Ah, power is a problem. Let's see how we can dynamically adapt this structure and make it more power efficient.'' We don't go back and solve the original problem of energy efficiency: how to make the structure inherently more energy efficient.

Ghose: You know, the trouble is that starting from a clean slate is always a hard sell. No one will believe in it. That's always part of the problem.

Skadron: I don't think that Pradip is arguing that we start from a clean slate in design. I think he is arguing for a conceptual solution.

Ghose: But I think we need to start from a clean slate, because who's to say that out-of-order cores are the best way to go? Very long instruction word (VLIW), if you want to talk about energy efficiency—now there's a technique. Transmeta came up with a product that had some known problems, and it has been bad-mouthed to death, so no one is going back to look at the viability of the VLIW approach. But with VLIW, you can support dynamic compilation efficiently; you can still support nonatomic transactions to get a significant speedup and reduce energy dissipation dramatically at the same time.

Skadron: But doesn't VLIW require extensive speculation to realize your single-thread performance?

Ghose: Right, but the jury is still out on which one is the more energy-efficient approach: out-of-order superscalar machines or VLIW.

Power-aware adaptive structures and control

Audience member: There has been a lot of work on adaptive structures and control and so forth, which I think could be a solution, but it has been a big disappointment, at least in my opinion, that industry hasn't adopted some of this stuff. Why not, and is there any room in that range? Are we looking at too simplistic a system? Do we need to complicate things more? Are we looking at the wrong things to adapt?

Ghose: Industry has used complicated dynamic voltage scaling. If you look at V_{DD} levels today, the range of change in the dynamic voltage scaling has been shrinking. We're rapidly reaching a point where the voltage control knob is limited when it comes to dynamic voltage and frequency scaling (DVFS). There are techniques that help us out. The Razor latch is a good example. Don't over-design to meet the process variations, but squeeze out as much use as you can, and bring down the voltage as much as you can. There are other similar techniques. I'm a member of academia. All I know is that it takes three verification engineers to work with one architect. Given that, I would suspect that the verification expense of putting out a new idea on the field, and the not-invented-here syndrome, has prevented these techniques from becoming mainstream.

Bose: Take anything, adaptive caches for example; change the cache size adaptively as a function of workload. What do we normally get? It saves 10 percent power at a certain performance cost. We seldom ask the question, why didn't you have a smaller cache to begin with? I mean, you get the same energy-delay performance as if you reduced the amount of cache statically. So it won't fundamentally pay off, especially in areas that have low power density. So if you have to focus on adaptive architectures, you should focus on hot spots. I have no problems if you focus on the issue queue inside one core or if you dynamically redistribute tasks across the cores to reduce peak temperature. Also, the dynamic adaptation idea doesn't pay off unless you demonstrate a good power-performance benefit ratio. If the idea yields 1 percent performance degradation for 1 percent power reduction, that's not interesting. I might as well just adjust the frequency knob.

Audience member: Is this an exhausted field, or is there room?

Bose: There is certainly room. Kanad already brought up one point. We don't do enough analysis to understand the limits of available energy efficiency. Take clock-gating. A couple of years ago I think we tried to write a paper on the limits of clock-gating efficiency. There's a dilemma here. If you look at actual microprocessors and where power goes, there's very little activity. If there are a billion transistors, how many of them are switching in a given cycle? A really, really small fraction. So you would be tempted to say, if I could only deeply mine this and clock-gate the heck out of it, I'm going to have super efficiency. But to mine this, you need to add so much hardware and complexity that after a while you won't win. So at what granularity should you mine this kind of clock-gating opportunity, and therefore, what's the practical, achievable limit?

Asynchronous design

Audience member: I'm not an asynchronous person. I don't have a horse in this race. But if the right person were in the room to defend asynchronous, they would say the answer is that if we think broadly, we don't need to clock-gate; we'll just do an entirely asynchronous design and exploit this feature.

Bose: True, but there you would have to worry about, from an engineering point of view, the right granularity of this asynchronous boundary. Do you make each transistor asynchronous, or do you make each of the ALUs the boundary?—and so on. I don't think we understand that. I don't think that problem has been analyzed. Where does it pay off? Unfortunately, also,

the tools issue comes up: simulation and verification methodology, for example. Asynchronous design has always been a problem from the tools viewpoint.

Power reduction via process technology versus microarchitecture

Audience member: I haven't heard anyone talk about the processes. From an industry point of view, where do you get the most bang for the buck? Isn't the research dollar much better spent to reduce the power losses in the semiconductor technology than on anything you can do with the architecture?

Skadron: Are you arguing that all the investment should go into process?

Audience member: Another audience member asked about an industry perspective. I've been in industry now for many years. And that's the focus, I'm telling you. That's where everyone is, and if there's demonstrative gain that the marketers can use to take to the marketplace, then the design houses—like Intel, AMD, Motorola, and IBM—are going to listen to them and say, can we implement this in a process that we can benefit from? They can be convinced, but it takes a strong argument.

Ghose: There is something that is happening in the real world today: Hafnium-based transistors. It's going to take care of some of the gate leakage. If you believe in the projections and some of the Intel Transistor Lab papers, in about eight to 10 years, gate leakage will come down by a factor of four to five. So process technology has always been at the forefront of this power-reduction story. Having said that, I say that we must look for technology-independent solutions. You can always use a better technology to make such solutions even better. If the design is bad to begin with, there is no point in continuing the bad energy-inefficient design.

Skadron: I can't resist interjecting my own opinion here: I think the architectural contributions are complementary to the process contributions. And you're right, you have to justify both. But some of the ways that our field can help are by coming up with designs that reduce intrinsic hot spots, by coming up with designs that provide abstractions that let us control power and exploit runtime behavior. These are just a few of the ways that we can build on top of any gains that the process provides.

FPGA-based simulation

Audience member: I was wondering if you could comment more on simulation methodology and early-stage design. What is lacking in the current approach, and why doesn't a FPGA make sense?

Bose: You said the right generic phrase, ''early stage.'' What is the right set of tools at that stage of the game? I don't think that you first develop a huge FPGA-based emulator or full-system simulation methodology to address that problem. You can design your basic concepts using fundamental reasoning techniques: Simple analysis tools and formalisms are based on the microarchitect's strength—intuition. Things have become very nonintuitive with such a huge design space, so I'm arguing to go back to simple analytical ways of reasoning rather than these complicated full-system simulation or emulation platforms. Those are important in their own right for later-stage application development and debugging, compiler tuning, and so on. As microarchitects, we want to lead the design team into an inherently energy-efficient, high-performance paradigm, and I don't think we need very sophisticated full-system simulators to try and prove our point. What's the optimal pipeline depth? What's the right core complexity? What's the right number of cores? We don't need very complicated models to make these earliest-stage arguments, in my opinion.

Ghose: We need to start identifying killer thermal applications, power applications, and say that this is the kind of application that will make the issue queue or the register file heat up a lot and use a few of these macro-sized benchmarks to exercise our system. We don't need a detailed register-transfer-level model, maybe something at

the subsystem level. And for goodness' sake, we should make use of multicore hardware to speed up our simulations. We haven't done that yet. MICRO

References

1. J. Choi et al., ''Thermal-Aware Task Scheduling at the System Software Level,'' *Proc. Int'l Symp. Low Power Electronics and Design*, ACM Press, 2007.
2. H.F. Hamann et al., ''Hotspot-Limited Microprocessors: Direct Temperature and Power Distribution Measurements,'' *IEEE J. Solid-State Circuits*, vol. 42, no. 1, Jan. 2007, pp. 56-65.
3. E. İpek et al., ''Core Fusion: Accommodating Software Diversity in Chip Multiprocessors,'' *Proc. Int'l Symp. Computer Architecture* (ISCA 07), ACM Press, 2007, pp. 186-197.

Kevin Skadron is an associate professor in the Department of Computer Science at the University of Virginia. His research interests include power- and temperature-aware design and architecture of many-core and graphics processors. Skadron has a PhD in computer science from Princeton University. He is a senior member of the IEEE, the IEEE Computer Society, and the IEEE Circuits and Systems Society, and a senior member of the ACM. He is a cofounder and associate editor in chief of *IEEE Computer Architecture Letters*.

Pradip Bose is a research staff member and manager at the IBM T.J. Watson Research Center. His research interests include high-performance computer architectures, power- and reliability-aware design, and computer-aided design. Bose has a PhD in electrical and computer engineering from the University of Illinois, Urbana-Champaign. He is a Fellow of the IEEE and the IEEE Computer Society, and a member of the ACM. Bose is also a former editor in chief of IEEE Micro.

Kanad Ghose is a professor in and chair of the Department of Computer Science at the State University of New York, Binghamton. His research interests include microarchitectures and compilation techniques for low power, active memory systems, and high-performance networking. He has a PhD in computer science from Iowa State University. He is a member of the IEEE, the IEEE Computer Society, and ACM, and an associate editor for the computer architecture area of *IEEE Transactions on Computers*.

Biographies of **Resit Sendag, Joshua J. Yi,** and **Derek Chiou** appear on p. 24.

Direct questions and comments about this article to Kevin Skadron, Department of Computer Science, School of Engineering and Applied Science, University of Virginia, 151 Engineer's Way, PO Box 400740, Charlottesville, VA 22904-4740; skadron@cs.virginia.edu.