The Future of Architectural Simulation

Doug Burger
Microsoft Research

Joel Emer
Intel

James C. Hoe
Carnegie Mellon University

Derek Chiou
University of Texas at Austin

Resit Sendag
University of Rhode Island

Joshua J. Yi
University of Texas School of Law

Simulation is an indispensable tool for evaluation and analysis throughout the development cycle of a computer system, and even after the computer system is built. How simulation should evolve as the complexity of computer systems continues to grow is an open question and the subject of this panel from the 2009 Workshop on Computer Architecture Research Directions.

Simulation is used extensively in computer architecture research and development to evaluate ideas in a cost-effective way before expending the considerably larger effort to implement those ideas. The standard design process that incorporates simulation is one of successive refinement, where ideas are first generated and filtered through intuition and discussion. The details of those ideas are then simulated to corroborate the expectations developed earlier. In each round, developers take the simulation results into account when innovating and selecting the next round of ideas to simulate at a greater level of detail and refinement. In a successful refinement sequence, each successive iteration results in a more specific design and a lower risk of abstraction errors, ultimately yielding a design to be committed to hardware.

As computer systems have become more complex, their simulators have also grown in complexity and, as a consequence, simulation performance has suffered. As Joel Emer (a panelist) has noted, Intel cycle-accurate performance simulators currently run at roughly 10 thousand instructions per second (KIPS) when simulating a unicore processor. This is quite slow, especially considering the amount of state (such as caches) in modern machines that needs to be warmed up before simulation results can be reliably taken. Compared to even a relatively slow 1-billion-instructions-per-second (GIPS) processor, this corresponds to a factor of 100 thousand slowdown. The situation is still worse given the current focus on multicore processors. For example, if you try to simulate a multicore processor with eight 1-GIPS cores, even under an optimistic assumption of a linear slowdown, a 10-KIPS simulator would correspond to a factor of 800 thousand slowdown—that is, at 10 KIPS, it will take more than nine days to simulate 1 second of the eight-way multicore processor execution. Things will only get worse as the number of simulated cores grows.

Improving simulation speed while maintaining accuracy and reasonable ease of simulator implementation can significantly impact computer architectural research. One must, however, also step back and consider when simulation is really warranted. Simulators themselves can influence what research is performed. It is a lot easier to study an area if there is an existing simulator for that area. Thus, creating a better, faster set of simulators that are optimized for a particular research area, such as traditional multicores, might funnel more research energy

Published by the IEEE Computer Society

into that area, possibly for a longer time than should be done by academics.

This panel on the future of simulation in computer architecture research explored three questions:

- When is simulation necessary?
- What do we need from simulators to enable more radical research?
- Can the easy availability of simulation actually impede progress?

The first panelist was Joel Emer, an Intel Fellow and a Fellow of both IEEE and ACM. He is the recipient of the 2009 Eckert-Mauchly Award. He comes to the panel from the vantage point of having developed Asim and HAsim, a simulation environment used inside Intel. Emer has also participated in the RAMP (Research Accelerator for Multiple Processors) project. The second panelist was Doug Burger, who manages computer architecture research at Microsoft Research, Redmond. He is the recipient of the 2006 Maurice Wilkes award, an IEEE Fellow, and an ACM Distinguished Scientist. He comes to the panel from the vantage point of coreleasing and cosupporting, with Todd Austin, the SimpleScalar tools for the research community. More recently, he and Steve Keckler codeveloped the TRIPS (Tera-op, Reliable, Intelligently Adaptive Processing System) prototype. James C. Hoe of Carnegie Mellon University moderated.

To start the panel, each panelist had five minutes to present a position statement. Following the opening statements, the panelists had approximately 25 minutes to direct questions to each other. This discussion was followed by open questions from the audience. Here we present an edited transcription of the conversations that took place.

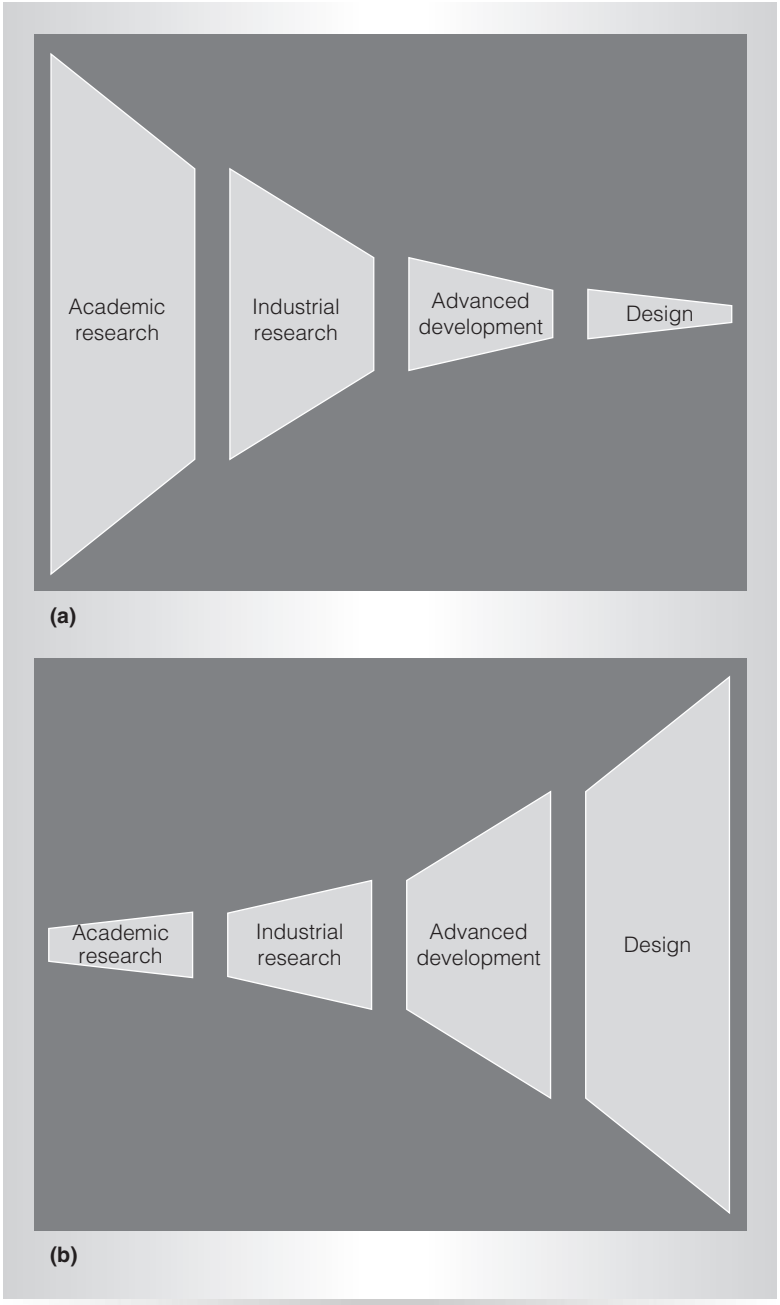

Figure 1. The flow of ideas from academic research to industry design and implementation (a). Ideas are held to higher standards at each successive stage. The relative cost of evaluating ideas from academic research to design (b). The cost to qualify an idea increases at each successive stage.

The need for speed and how to achieve it: Joel Emer

Industry uses the idea-generation, filtering, simulation loop quite extensively throughout the research and implementation phases of computer design. Each successive iteration explores the surviving ideas even further, resulting in some ideas being dropped and the remaining being proved even more (Figure 1a). Each successive iteration also becomes more expensive as more people are brought on board to further refine, model, and design (Figure 1b).

This process implies that before you move an idea from one stage to the next, the

evidence that the idea is a good one must be compelling enough to justify the cost and effort. An interesting question, then, is what is an adequate amount of evidence to proceed to each next stage?

As an example, consider caches, a microarchitectural component that can have a significant positive impact on performance if they're correctly designed. A 64-Mbyte cache with 64-byte cache lines has one million lines. Within Intel, we want to replace each cache line an average of approximately 10 times before we start believing the results. At a miss rate of about four misses per thousand instructions, which is about right for a 64-Mbyte cache, we need to simulate approximately 2.5 billion instructions before the results are believable. At our current simulation speeds of 10 KIPS, it takes roughly 2.5 days to simulate 2.5 billion instructions. Such speeds are intolerable.

We need similar simulation lengths to understand the behavior of multicore systems. To get believable results for eight-core systems, we sometimes run each simulation for a month. The same is true when doing thermal modeling. How many of you are willing to wait a month to get the results so you can publish a paper? [*laughter*]

The consequence of current simulation speeds is incremental ideas. More radical ideas require more extensive simulations to reach the bar of believability, making it far easier to extend old ideas than to try something new and more interesting.

A number of people, including myself and the RAMP community, have looked at the problem and concluded that using field-programmable gate arrays (FPGAs) will improve simulation performance. There are many possible approaches to using FPGAs to accelerate simulator performance. You could use FPGAs to implement a functional emulator that's functionally equivalent to the design but doesn't provide any timing information—or any particular design metrics—except for correct functionality. That doesn't turn out to be very useful for studying a microarchitectural idea because you don't get any timing information from it.

Another approach is to build a prototype. This is logically isomorphic and functionally equivalent to the proposed design except that it's often implemented in a technology, such as FPGAs, that's different from the intended technology, such as a full custom design. The danger of such an approach is that a single FPGA often isn't large enough to contain a full design, creating the dilemma of either changing the design to fit and thus likely creating a functional emulator that doesn't accurately model time, or partitioning the design across multiple FPGAs, significantly increasing the amount of work. In any case, the detail required by a prototype implies that many decisions are already fixed, making a prototype a poor vehicle for exploring alternatives.

Thus, rather than proposing emulators or prototypes, I'm arguing for models that are abstract representations of a design and are sufficiently logically and functionally equivalent to the evaluated design that we can accurately estimate interesting metrics such as performance, power, and reliability.

Everyone has seen the standard simulation trade-offs between simulation speed, accuracy, and modeling effort. If you're using FPGAs, you should be able to be both accurate and fast. However, the effort required to create the model can be much larger than that required for standard software simulators.

We should be focusing on how to do make FPGA-based models that are easy to both create and modify. We need some notion of a standardized ''unmodel'' infrastructure that provides control over the model, the ability to set parameters, the ability to export statistics, and so on. I've long advocated splitting simulators into a functional model and a timing model to avoid needing to re-implement the functional model for each new timing model.

Also, don't put everything into the FPGA. Make sure you have an environment where you can implement some components in the FPGA and some outside the FPGA. You must also provide an environment where the edit-compile-debug loop is fast. We need to write in a language that's more like a standard programming language and less like a hardware design language. We need fast compile times and we need debugging.

The shortcomings of simulation: Doug Burger

I would like to start out by saying that Joel is more of a simulation expert than I am. I see several people in this audience who are considered to be more expert in simulation than I am. So I'm going to take a slightly broader view of the problem in the field. I think Joel is right on many if not all of those points. But, I'm a little worried about RAMP and the shared infrastructure using FPGAs and I think this is a good point to debate.

Our community has reached a multicore consensus that is, at this point, a failure to improve individual cores the way we did in the past. Because of technology constraints, we're running down this multicore path not knowing where it goes, not seeing how far we can run, and hoping that things turn out okay in the end. It's going to change the way systems are designed.

For decades, the academic architecture community was in this nice stable ecosystem where the instruction sets, to first order, were stable, where the technology was predictable, and where PCs and eventually laptops were where most of these chips would be used. There were servers at the high end but, for the most part, we could do research on the core design and the memory system design and come up with interesting ideas. If they were really good ideas, they might find their way into industry, in which case the researcher developing the ideas would have a big impact.

The shared simulation environments from that time benefited from that stable ecosystem. Take, for example, the SimpleScalar simulator tools developed in Guri's (Sohi) group that contained a lot of code and simulation models that Todd (Austin) wrote. SimpleScalar was influential because it reduced the modeling effort for several common problems in the community. It thus let people do research without the startup effort of developing their own simulator from scratch.

A downside to SimpleScalar was that it became the de facto standard for many researchers and thus often shifted the research focus to what could be easily done within SimpleScalar. In some cases, that shifting was a good thing; in other cases, not such a good thing. For example, many were still doing SimpleScalar instruction-level parallelism research when Intel and AMD canceled their high-end unicore parts and shifted to multicores. Rather than leading industry and working on multicore long before industry shifted, many academics followed industry in this latest shift. Similarly, after data centers reached a critical mass in industry, some researchers started research in that direction. Instead of following industry, the academic community should be leading industry by thinking about what systems will look like in 10 years. The question, then, is what shared infrastructure do we build to allow us to think about far future systems without each of us having to implement such an infrastructure on our own. The real challenge is choosing the right infrastructure, especially when funding resources are limited. I'm not sure if the problem is the speed of the simulation and the support issues within simulation. Rather, it's trying to build shared models and shared infrastructure that we can leverage to do interesting systems research and ask questions that are further out than what the advanced development groups in industry are doing. For industry, it would be useful to have a big academic community coming up with ideas that will help them tomorrow. I think that's a positive thing. But I also think that a lot of the academic community should be looking much further out.

Instead of following industry, the academic community should think about what systems will look like in 10 years.

Many, perhaps most, of the papers being written today do apples-to-oranges comparisons. Most compare against their own simulator modified in an attempt to model a reasonable baseline. How often are

comparisons made with an Intel Core i7, Core2, or IBM Power7? What about the proposed mechanism's cycle time impact? Are process variations, thermals, system-on-chip (SoC) effects, frequency binning, and power versus performance also considered? If they are, the common approach is to slap together different models obtained from different sources and hope they make sense running together. It's hard to accurately compare such results with real hardware models.

In the market, cloud computing is causing massive innovation in servers and giant data centers. At the same time, devices are getting much smaller, such as cell phones. Integration is happening across the board, within laptops, netbooks, and even PCs. The processor is only one component among many. Systems are rapidly becoming more SoCs. It isn't sufficient to do core research or memory system research or even whole package research because we have to think about issues such as integrated Ethernet and what it connects to.

In terms of simulation, we should consider whole system design. You might be working on the cores and running SPEC and thinking just about the core and that's fine. But that's just one component. We should be thinking about how to model the entire system using core models, memory system models, thermal models, reliability models, and so on, and how to compose these elements together in a coherent fashion. This is where Joel went at the end of his presentation, and I agree. But I'm not sure that one shared infrastructure will be sufficient. I worry that if we spend a lot of resources developing one shared multicore infrastructure, there will be no alternative to using that infrastructure. Having one shared infrastructure may make the methodology bar so high that the community won't accept results generated any other way, thus forcing us down one research direction. For a paper to be accepted, program committees will want to see RAMP results run on FPGAs. That's a dangerously high bar. So, I think we need to take a broader view and think about a shared methodology that lets us do further out research on whole systems rather than focusing on how to simulate 32 cores.

Shared infrastructure

Emer: I agreed with much of what you said. You seem to imply that having a shared infrastructure will channel research. Maybe the behavioral model you're thinking about is like the SimpleScalar model, where the pipeline model came with and was tightly bound to the infrastructure, as best as I understand it. You notice in my list I didn't include any microarchitectural models as part of the shared infrastructure. The main shared infrastructure I envision includes components besides those that predict timing characteristics. As a participant of RAMP I would love to have the methodology bar be the RAMP bar, but I don't think that implies that you have to use a particular core model or that you have to focus on the core and not the entire system.

Burger: I'll respond but I think Guri is a better person to address this point. [*Sohi* (*Wisconsin*): Don't bother me; I'm having a good time being quiet.] The SimpleScalar release included sim-out-order, whose parameters allowed it to be configured as an in-order model, as well as a couple of other simple timing models that could be used to predict cache behavior from a trace. Todd designed SimpleScalar so that the ISA emulation and parameter setting were decoupled from the timing model. It was a cleanly designed timing model. And some researchers designed their own timing models after some point. But most researchers wanted it for the existing timing model.

If the RAMP infrastructure is released without a timing model, I think a lot of people won't use it. But as soon as someone releases a timing model, a lot more people will use it because it would let them do things that they otherwise couldn't do with the available resources. And that timing model will grow in importance and could become a standard. Research will be done based on whatever timing model becomes popular.

Emer: You're absolutely right. If you don't release a timing model, no one will touch the infrastructure. I don't think you can fight the fact that some people just use the pre-existing timing model. The fact that some people did build their own customized

timing model for presumably more radical and innovative ideas out of the same infrastructure should be viewed as a success.

Burger: Yes, I'm not arguing that the tools weren't successful. In the SimpleScalar case, clearly they were. RAMP has the same potential for multicore.

Is simulator speed an issue?

Emer: The other point you made was that we don't necessarily need faster simulators. I think the main thing that we need are faster simulators. Yes, the faster simulator should be able to handle the full system and multicore. The list of things that you said we wanted to do, such as thermal analysis, power analysis, 3D memory, and fancier memory hierarchies, requires longer simulations to accurately model. The combination of current simulation rates and the fidelity they can achieve doesn't add up to a compelling reason for people in industry to sit up and pay attention.

Burger: I would respond to that, Joel, in two ways. If you want to model all of those features at once with high fidelity you need enormous simulation capabilities and enormous processing power that amounts to simulation speed. Recently Steve [Keckler] and I, like lots of others, have been trying to do power simulations. What we wound up with are models sitting on models sitting on models sitting on models. We really have no idea how accurate those models and their combinations are. At some point, you have to ask yourself whether you can really make a contribution if you can't be accurate without detailed knowledge that you don't have access to, such as the details of Intel's or IBM's technology.

Industry impact and simulation

Emer: You have to figure out some way to get the big picture. It has often been the case that somebody in industry looks at an academic research idea and recognizes that the base technology assumptions aren't right, but the question is always, ''Was the work compelling enough to warrant going to the next level of investigation?'' If we saw a paper from academia that seemed interesting, it would have to cross the credibility bar to warrant us reevaluating that idea with more realistic parameters. If we had a good common infrastructure, maybe some of those modifications would simply be parameter changes, allowing many more things to cross the gap.

Burger: The point I was making was that the stable ecosystem of the past two decades was an environment in which an academic could do some work and, even if the timing model was off, industry could take a quick look and determine that the threshold was crossed. I worry that we're moving into a regime where it's intractable for academics to realistically have our work be at that threshold, especially given technology's ever-increasing impact, the systems' complexity, and the sheer magnitude of what needs to be right to get a correct result.

Emer: I think there has always been a huge gap between academia and industry. [*Burger: I think that gap has grown.*] So we need to address that. There have always been a lot of papers with fundamental defects such as ignoring multiprocessor effects or ignoring context switches. Really fundamental issues, far beyond being in the wrong process technology. Industry can't interpret anything from such results. [*Burger: One would hope that the program committee would have caught such problems.*] Unfortunately, they didn't.

Burger: We can argue about whether the modeling is getting too complex at this level to really influence where industry is going, especially with the shift to multicore and the process variations and the thermals and all that. There is certainly good work going on in academia that I think will affect industry in the long term. Since Moore's law may end (for logic) in six or seven years, or certainly I would say within a decade (we can debate that later), academics should be thinking 10 years out and not trying to influence the current direction of industry. We've recently seen current industry take major shifts. Say you start a project that takes you seven years to complete. By the time you finish, industry might have gone

in a completely different direction. If you base your project's premise on what industry is doing, you're left hanging. I think we're going to see several more major shifts. So, we should be considering modeling that would help academics research things that are radically different, potentially creating new fields or new industries, and not trying to optimize for industrial influence now.

Emer: I absolutely agree. The flow of ideas should be from further-out academic research to nearer-term industry. Whenever an academic comes to me wanting details about current technology, I always say you shouldn't be worried about that. We'll handle today's problems because we have a lot more insight into the details. Look further out.

In terms of Moore's law ending, there's a good chance that it will create more opportunity for us because we won't be able to rely on the fact that there are more transistors. Until now, we haven't had to be as creative as we could be because we're always trying to keep up with more transistors. When that slows, the need to create better designs will increase. [*Burger: And so whatever shared infrastructure we envision should be designed to help that creativity.*] Absolutely. I don't know why we're disagreeing. We should have infrastructure that lets people increase the odds of reaching the bar. If researchers don't have to spend time figuring out how to collect statistics or how they're going to put things onto an FPGA to get those faster simulations, they'll have a better chance of reaching the bar. We should build that infrastructure so that others do not have to. We have to live with the fact that some people just use the provided models. But we can revel in those who take that infrastructure and build on it to do something truly creative.

Whether you do simulation or not doesn't obviate the need for thinking.

Hoe: Do we all agree that simulation is an important technology for finding the answers we need 10 years from now and that we currently don't necessarily have the right simulators?

Burger: I can only speak for myself, but I would have to say yes. It will be some combination of simulation, prototyping, and modeling, and simulation will be a key component.

Don't simulate, pontificate

New question from the audience (Guri Sohi, University of Wisconsin): The future of simulation is to kill it. Let me tell you why. How many of you remember a tool from MIPS called Pixie? Pixie was a tool that gave you fantastic results for in-order pipelined processors. It was a little more complicated for out-of-order processors. Franklin wrote the first simulator of an out-of-order processor, which eventually became SimpleScalar, to make a case for nonblocking caches and nothing else. When we live in an out-of-order world, we need this simulation stuff. Once we move to the multicore world, what happens in an out-of-order processor isn't that relevant. Today, people talk about wanting to have stuff like deterministic execution in multicores. The only thing that's going to matter is the communication that happened and that communication's rate and timing. The communication that's going to happen can be learned nicely using some Pixie-like model and the timing in-between with a good analytical model. I don't have to figure out when the next memory operation will come out at 10 KIPS. I can do it with a simple analytical model.

Why think when we can just simulate, simulate, simulate? Let's try to think sometime. Especially in a multicore environment.

Joel made an excellent point. Incremental versus radical. Simulation leads to incremental rather than radical results. Get rid of simulation and you will definitely get more radical results. Kill simulation!

Emer: I have to respond to that. I don't think you're making an A or B decision.

And, I don't think simulation is counter to radical ideas. I think you have to think about how you approach it. Whether you do simulation or not doesn't obviate the need for thinking. You need to do the thinking and you need to figure out how to do stepwise refinement of your evaluation. But ultimately, you need to do a detailed simulation because simulation helps find things you didn't expect. We always find that we need the complete simulation to guarantee that we haven't overlooked anything. The abstractions aren't good enough in a full system to let us ignore the details. You want abstractions initially to prove that you're on the right track. But ultimately to cross the bar to say this is a good idea, you need the details.

Burger: I would like to make a broader point than Guri did. James, maybe some of the reason that I react against the FPGA stuff is because some of it is motivated by the need to simulate 64 cores. It's not clear that is an interesting point. I think there is good research to be done at 64 cores on a chip with coherence protocols, cache sharing, on-chip networks, and power management, but I think that's going to be a subset of what the community will be doing, and I wouldn't be surprised after Moore's law ends that simulating multicores isn't what we should be focusing on. How can we use transistors, 99 percent of which must be quiescent at any one time? It might be neural networks. It might be SoCs. [*Emer: Whatever it is, it still could require long simulation times to run better, more extensive workloads.*]

Because of the number of devices, there's going to be an enormous amount of stuff to model and that's going to take time however you're modeling it. There's an accuracy-versus-time tradeoff whether you model at the cycle level or at a higher level or use a mathematical model. But, if we presuppose that we're interested in simulating some number of cores—32, 64—on a chip, with some core architecture, we're going to have to design a simulation infrastructure. The research community already focuses too much on multicore in my view. I'm pushing back against that focus a little bit by saying that it's dangerous to presuppose a target and then build shared infrastructure that everyone starts to use because it's going to drive more of us in that direction. It might stop us from thinking. I can just grab some tool, start optimizing, and generate some papers. That's great for me but bad for the community if too many people do the same thing.

Emer: You're saying that if we don't run multicore we don't need longer simulations?

Burger: I'm arguing against the statement that the time to simulate 64 cores is the crisis we ought to solve with shared infrastructure. Obviously, fast simulation is better than slow simulation.

James: I was trying to point out that systems will get more complicated and concurrent in the future. But saying we only want to simulate *N* cores is oversimplifying the problem.

New question from the audience: Are you arguing that there's no need for quantitative measurements as we go to multicores? Or do you have some alternative way of doing that, besides simulation?

Burger: It depends on the kind of research you're interested in doing. If you're interested in how efficient can you make an on-chip 64-core coherence protocol, you need a quantitative evaluation metric. It might be an analytic model driven by arrival times in traces. It might be that you do a full-blown simulation to guarantee your timing results. But, pushing our shared infrastructure so that it is best at generating quantitative results for that type of system is overly constraining to the community if it becomes the dominant shared infrastructure.

So, if the unsolved problem is how to support coherence in 64 cores on a chip and that's what everyone works on, then, yes, quantitative evaluation of coherence in 64-core chips would be important to compare results.

Follow-up from the audience: I personally think that people sometimes use quantitative results because they don't want to think about the problem and it's sort of

an easy way out. But, at the same time, the real question remains that if you're going to build real systems, will quantitative evaluation of ideas at some stage play a critical role. [*Burger: Clearly, yes.*] And, if it will play a critical role, we need infrastructure for doing that kind of work. [*Burger: Yes we do.*] So that's the point where I'm not getting a clear reading. Do you think it will be a fringe activity where only 3 percent of the architects will be doing quantitative evaluation and the rest of us will be generating ideas and pontificating?

Burger: Well, I think we need more generating of ideas and pontificating. I also think that if we require too much quantitative evaluation we'll constrain what people can do with a reasonable effort. I've started to see papers where less quantitative evaluation, or even none at all, is completely justified. I think in terms of the author's strategy for getting his or her work published, where the author needs a thermal model, a power model, and a reliability model—which you need if you're working at the intersection of architecture and technology. Having so many models is constraining. You can't use those models if you want to do something more speculative. For example, I might want to think about whether I can drop my supply voltage to 0.3 volts by creating some FinFETs in some new material and what that would do to the architecture. However, I can't do quantitative modeling at that level, it's too speculative.

If we require too much quantitative evaluation, we'll constrain what people can do with a reasonable effort.

The higher the methodology bar, the more constraining it is for people to cross over it. Detailed methodology is a good thing. It's good to have these results. If you're going to build a system, you really want to understand what's going into it. But I'm seeing the bar as getting more and more constraining.

Emer: There's more than one bar. The bar between advanced development in industry is much higher than the academic bar. You're saying that we're in danger of putting the first bar too high and that we need to have a methodology that enables early-stage speculative exploration. That's true. But a lower first bar will require additional, more incremental steps before it can be considered for a real industrial design.

Burger: That would be a feature, not a bug. We may have been spoiled in academia by having our ideas move into industry too quickly in some cases.

Emer: I agree with you that early research needs to focus further out. So, another characteristic of that flow is that the idea's expected appearance in a design should be further out.

Burger: If we go through the ISCA (International Symposium on Computer Architecture) proceedings and ask how many of these papers are high risk, what ratio would we come up with? How much of the infrastructure required for these papers limits the amount of risk an author can take?

Emer: That's a process problem, not an infrastructure problem.

Burger: You don't think there's a loop from the infrastructure back to the process?

Emer: I mean process in terms of review committees and the nature of the tenure process.

A common infrastructure across levels?

New question from the audience: What I'd like to understand is how the different phases that Joel was putting up require different amounts of fidelity and how well they represent reality. I don't see a good way of

migrating and reusing models between those different levels, and that might be one of the problems. This also relates to the effort and issues in passing research results from one group to another. What we're simulating and why is an interesting question.

Emer: You alluded to using the same infrastructure from top to bottom. I'm not sure if it's feasible to do that. Pieces of this infrastructure might be more suited to specific levels than others and you need them to provide different levels of fidelity. An abstract level doesn't need or want details of every metric at the cycle level, but we do need tools that operate in the abstract domain. However, once we find a sufficiently compelling result, we also need more detailed models to help find the surprises that we don't find when we simply say this is a very elegant idea.

Burger: Another problem is the subtle tradeoff between power and performance that makes it hard to look at any design point and say that this design point is a good result.

Emer: At the very, very fine microarchitectural level that's true, but maybe that's not where the research community should focus.

Burger: Even at a higher level—at a chip level or a memory system level—if you say that you can run 1700 SPEC marks at 16 watts, is that good? What frequency are you running, what process technology are you running in?

Emer: A fundamentally good idea should have legs to be able to move from one process generation to another.

Burger: That's true. The metrics to know whether something is good or bad have gotten hard. Joel, I think your view has been that academics should describe an idea and publish the results but that the quantitative evaluations really don't matter because people in industry will ignore them and do their own evaluation if they're convinced the idea is good.

Emer: No, I'm not saying that the academics should have no quantitative evaluation because then there's no basis to decide whether the idea is worth investigation.

Burger: It seems that the precision of the results you need to make that decision is much lower than what we provide today.

Emer: Frequently, that's not true.

Hoe: One of the motivations for building fast simulators was to enable our software colleagues to start working on operating systems, compilers, and applications before the hardware was available. Will there be a day when architects, operating system developers, and compiler developers come together and do something really interesting?

Burger: I don't see that happening anytime soon. I think the answer is yes but you need to provide the right incentives to the programming languages and operating systems people. And that's hard.

Conclusion

It is an academic architect's job to look far into the future, perhaps 10 years or more, rather than focus on the issues of today. Simulation has its place in architecture research since it provides a higher level of confidence in the validity of architectural ideas by exposing effects that the architects might not have originally considered. Far too often, simulation is used as an excuse to avoid first thinking about the architecture and analyzing it using lighter-weight methods. In addition, heavy emphasis on simulation can sometimes lead to incremental work because the easiest simulator to build is a small delta from an existing simulation model. Thus, any significant shared simulation infrastructure should be designed to avoid channeling research toward near-term, incremental work. It should also be fast enough to generate meaningful results in a reasonable time. MICRO

Doug Burger is director of client and cloud applications research at Microsoft Research, where he also manages the computer architecture research group. His research interests include computer architecture, new computing technologies, mobile user interfaces, cloud applications, and, of course,

computer system simulation. Burger has a PhD in computer science from the University of Wisconsin. He is an IEEE Fellow, an ACM Distinguished Scientist, and chair of ACM SIGARCH.

Joel Emer is an Intel Fellow and director of microarchitecture research at Intel, where he leads the VSSAD group. He also teaches part-time at the Massachusetts Institute of Technology. His research interests include performance modeling frameworks, reconfigurable logic computing, parallel and multithreaded processors, cache organization, processor pipeline organization and processor reliability. Emer has a PhD in electrical engineering from the University of Illinois. He is an ACM Fellow and an IEEE Fellow.

James C. Hoe is a professor of electrical and computer engineering at Carnegie Mellon University. His research interests include computer architecture and high-level hardware description and synthesis. He has a PhD in electrical engineering and computer science from the Massachusetts Institute of Technology. He is a senior member of IEEE and a member of the ACM. For more information, please visit http://www.ece.cmu.edu/~jhoe.

Derek Chiou is an assistant professor in the Electrical and Computer Engineering Department at the University of Texas at Austin. His research interests include computer system simulation, computer architecture, parallel computer architecture, and Internet router architecture. Chiou has a PhD in electrical engineering and computer science from the Massachusetts Institute of Technology. He is a senior member of IEEE and a member of the ACM.

Resit Sendag is an associate professor in the Department of Electrical, Computer, and Biomedical Engineering at the University of Rhode Island, Kingston. His research interests include high-performance computer architecture, memory systems performance issues, and parallel computing. Sendag has a PhD in electrical and computer engineering from the University of Minnesota, Minneapolis. He is a member of IEEE and the IEEE Computer Society.

Joshua J. Yi is a first year JD student at the University of Texas School of Law. His research interests include high-performance computer architecture, performance methodology, and deep submicron effects. Yi has a PhD in electrical engineering from the University of Minnesota, Minneapolis. He is a member of IEEE and the IEEE Computer Society.

Direct questions and comments about this article to James C. Hoe, Electrical and Computer Engineering Dept., Carnegie Mellon University, 5000 Forbes Avenue, ECE-HH-1109, Pittsburgh, PA, 15213; jhoe@ece.cmu.edu.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.