# Architectural Simulators Considered Harmful

Much as Edgar Dijkstra observed the dangers of relying on the "go to" statement, the authors here observe the detrimental effects of overreliance on quantitative simulators. They describe three broad pitfalls of simulators and simulator use and discuss how to avoid these problems. They also propose using the footprint—the breadth of architectural layers that a technique affects—for recalibrating evaluation standards.

Tony Nowatzki
Jaikrishnan Menon
Chen-Han Ho
Karthikeyan Sankaralingam
University of Wisconsin–Madison

"For a number of years we have been familiar with the observation that the quality of architecture researchers is a decreasing function of the reliance on quantitative architecture simulators in the architecture papers they produce. More recently we discovered why the use of architecture simulators has such disastrous effects, and we became convinced that the architecture simulator should be abolished from all 'higher level' architecture research."

Although this tongue-in-cheek rewriting of Djkstra's anti-go-to treatise[1] overstates the need to eliminate architecture simulators, we argue that we are in an era of overreliance on architecture simulators, and the effects of this are tangible and detrimental. For this article, we define *simulator* as a detailed software modeling tool that provides cycle-level performance, area, power, and/or energy estimates without relying on a register transfer level (RTL) specification of the machine it models. From the early days of SimpleScalar[2] with its register-update-unit-based out-of-order (OOO) model and fixed-latency DRAM, to the gem5+DramSim+GPGPUSim+McPAT mashup simulator, we have come a long way in what some architects would claim as being "validated." This level of added detail has led to the belief that we have stronger tools and are doing better quantitative evaluation.

However, we argue that our overreliance on simulators has three broad pitfalls in terms of their development, use, and effect on our community. The first is when key phenomenon are modeled at a too-high or incorrect level, simulators act like first-order models, are labeled as detailed simulation, and therefore have the benefit of neither. The second is when simulators are used as black boxes to get supporting quantitative data, while glossing over potential bugs, modeling errors, or lack of validation that could make the results less meaningful. The third is the detrimental effects on evaluation standards that an overreliance on simulation has caused, specifically an underappreciation for direct analysis and additional noise in the review process due to lack of consensus on evaluation standards. To substantiate the above, we consider examples from four modern simulation infrastructures: gem5,[3] McPAT,[4] GPGPUSim v2.x,[5] and GPUWattch.[6]



Published by the IEEE Computer Society

## The Role of Simulation and Alternatives

To aid in discussing the role of simulation in architecture research, we briefly summarize the general purpose of an academic research paper. For a standard "synthesis-type" paper, there are generally three components:

1. motivate a problem,
2. invent and explain a new idea, and
3. demonstrate the new idea's feasibility or potential benefits.

The purpose of an "evaluation tool" is primarily for tasks 1 and 3.

Cycle-level simulation, the most common approach, is well suited for capturing low-level, detailed phenomena. We avoid the use of the term "cycle-accurate," as this implies accuracy to an existing system at cycle granularity, which is rarely the case. We encourage authors to be responsible about the use of this term.

An important benefit of simulation-based prototypes is that they encourage architects to reason about microarchitectural feasibility (though they are not a substitute for hardware implementations). However, it is often taken for granted that quantitative simulators are required or best suited.

One alternative is implementation at the Register Transfer Language or even taping out designs. Although this is appropriate for many clean-slate designs, in practice it has limited applicability because of the development time, optimization effort, and integration with other system components.

Another alternative is first-order models. This categorizes a broad range of techniques, which can include trace-driven simulators, analytical models, mathematical proofs, or even event-driven simulators to some extent. These approaches or techniques are characterized by the ability to concisely and precisely describe all modeling components. Although it prevents the study of low-level phenomenon, the simplicity of first-order models allows them to be fully described in research papers, which can be more insightful and transparent for readers.

Fortunately, these pitfalls can be addressed, and we discuss in detail one approach for advancing more consistent and appropriate evaluation standards. Specifically, we suggest using the notion of *the footprint*, or layers of the stack that a research technique affects or relies on. Figure 1 highlights how different techniques, represented by gray boxes, can affect different stack layers, and it also shows our view of appropriate potential evaluation approaches. Using the footprint to calibrate our evaluation standards could lead us to insightful higher-level modeling techniques in appropriate domains. For more discussion of the relationship between evaluation techniques and architecture research, see the sidebar, "The Role of Simulation and Alternatives."

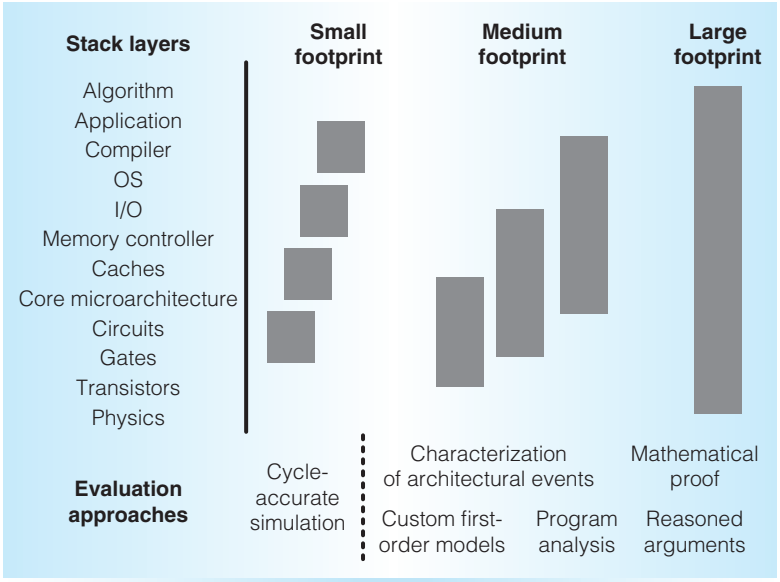


Figure 1. The relationship between a technique's footprint—the scope of layers it interacts with—and our view of appropriate evaluation approaches. For small footprints, an appropriate approach can be simulation, whereas larger footprints may be more amenable to higher-level approaches.

## Pitfalls of architectural simulation

Here, we explain the three broad pitfalls of architectural simulation, with examples drawn from various widely used architecture simulators. This is not to be construed as criticism of these simulators, and not all tools (and their respective uses) suffer from all pitfalls.

### Pitfall 1: Simulators with poorly modeled first-order phenomenon

The need for quantitative simulation results is driving the development of simulators in many new and challenging domains. Although they are promising, many have errors or oversimplified modeling abstractions in key features. These oversimplifications mean that the results can be only as good as first-order models, but they are labeled as detailed simulations. We argue that they have the benefits of neither. We also

emphasize that whether a phenomenon is first-order can depend on the use case. We discuss modeling examples here which, in our opinion, could have first-order implications in most use cases.

*Pitfall 1a: Oversimplified modeling abstractions.* Simulator writers must decide what to model in detail and what to leave abstract, and these decisions can affect what can be studied appropriately. Too often, there is an implicit statement in tutorials and documentation that unmodeled things have little first-order impact. However, this is often not the case and can misguide researchers.

We substantiate this with an example, considering a widely adopted version of GPGPUSim, v2.x, and describe several abstracted architectural features. Although GPGPUSim v3.x has addressed many of these issues, we focus on this version specifically because much GPU research has been published using this tool, and it remains the option of choice by many researchers studying precisely these features. The architectural features we consider include the following:

- *Register file microarchitecture.* The operand collector (single-ported register file banks + arbiter + X-bar + collector units) is modeled assuming fixed-latency accesses to the static RAM (SRAM) with some additional queuing latency. It does not model low-level details, such as contention, that impact performance in high-computational-bandwidth scenarios.
- *Thread/warp/wavefront scheduling and dispatch.* Thread scheduling is functional, and although a number of different warp scheduling schemes are implemented, these are not modeled in the microarchitecture.
- *Branch divergence structures and the branch unit.* Similar to thread dispatch, branch divergence tracking structures are functionally emulated as part of the abstract hardware model, and the branch unit microarchitecture is not modeled at the cycle level.

The effect of omitting the detailed modeling of these microarchitectural features, and accounting for them abstractly or functionally, is that it encourages architects not to reason about the proposed technique's microarchitectural feasibility, and obfuscates or incorrectly quantifies their performance impact. Our opinion is that GPGPUSim v2.x was not the appropriate choice for many GPU microarchitecture studies, and better documentation would have deterred such usage.

This issue will become only more pertinent as emerging and future simulators tackle even more difficult problems and must make further abstractions. One example is ZSim,[7] which models hardware at a very high level of detail and shows validation to commercial products in the final performance estimation. Such simulators present both an opportunity and a risk. For studying some phenomenon, like issues in large multicore cache design, ZSim is an extremely fast simulator that models a processor's appropriate components. However, it would not be useful for core microarchitecture research.

We suggest that simulator writers document the reasons for abstraction decisions and explicitly describe inappropriate use cases. These disclosures are strengths, not weaknesses, because this information guides appropriate tool use.

*Pitfall 1b: Unnecessary detail and overfitting.* Conversely to the previous pitfall, it is equally easy to add unnecessary detail to a modeling approach, with a side effect being overfitting to a particular design point. As an example, we consider the GPUWattch simulator[6] and briefly describe its methodology and potential for inappropriate use.

GPUWattch models cycle-level GPU power by using GPGPUSim to obtain activity factors for various components, which are fed as inputs into McPAT. Because McPAT is meant for CPU-like structures, GPUWattch scales these activity factors, bounded by ratios between 10 and 50×. These factors are tuned by first measuring a specific GPU's power (physically with sensing resistors) on custom microbenchmarks and then manually adjusting the scaling factors to lower the least squares error. We raised several concerns with the GPUWattch methodology elsewhere,[8] and we discuss the primary issues below.

Fundamentally, GPUWattch embeds a detailed model (McPAT) inside a high-level

regression model, and in our opinion does not have the advantages of either. To explain, we consider two usage scenarios. First, if the target is a validated design point, GPUWattch is similar to simple linear regression on the activity factors, because the high scaling factors on the GPUWattch inputs essentially cancel out the McPAT modeling component. It is less accurate than pure regression, however, because the bounds on scaling factors prevent full fitting to the data. It also adds potential for error by using McPAT internally.

A second usage scenario would be to study other GPU design points by modifying GPUWattch's architectural input parameters (which are ultimately fed to McPAT). However, these results would be error-prone, because using McPAT to handle the scaling of GPU components has not been validated. Intuitively, the power scaling of GPU structures will be different than the power scaling of the CPU structures that McPAT models and was validated for.

More generally, mathematically sound approaches are more robust and comprehensible than introducing levels of indirection and detail and building ad hoc techniques. The approach of scaling McPAT's inputs is an example of the latter, and its popularity reflects our community's reluctance to use the former, even when it is more appropriate.

Thus, we suggest that we not put more trust in tools because they embed more details, but rather that we treat with skepticism (as tool developers, tool users, and reviewers) the use of such detailed tools outside their validated domains. Instead, we can embrace simpler models.

### Pitfall 2: Treatment of simulators as black boxes

It is undeniable that simulation is indispensable when the details are the subject of study. However, they often get misused as black boxes to generate quantitative results. We substantiate this by showing the existence of significant bugs that have remained undiscovered for long periods of time, and we describe two incorrect lines of reasoning used to justify simulation's black-box usage.

*Pitfall 2a: Inaccessible simulator errors.* Finding and fixing simulator errors, much less becoming aware of their existence, is difficult because simulators are distributed as C/C++ code with little specification. Many of the features are obscured behind implicit assumptions, lack of documentation, and lack of good reporting of results.

It is true that simulators, like any piece of software, will always have bugs, and the presence of these bugs is not a criticism of the simulators. While the world works with buggy software in general, we substantiate why in this case these bugs indeed matter. We specifically use three example errors from the widely used and respected gem5 and McPAT simulators:

- *Inconsistent writeback mechanism.* The gem5 OOO model schedules instructions for issue only if there are guaranteed to be enough writeback buffers. The default number is the issue width, meaning that a few long-latency instructions will hold up writeback buffer slots, forcing the effective issue width to 0. This is not a design choice representative of real designs.
- *Inefficient and mislabeled micro-operations.* In gem5, X86 micro-ops are optimized for correctness and economy rather than efficiency, leading to unnecessary register dependencies. This often occurs because operations unnecessarily read the destination register, due to similar operations with the same micro-op that requires that ability. Also, the resource usage of many micro-ops is mislabeled. For example, both floating-point moves and loads unnecessarily use the FP unit, whereas some floating-point single-instruction, multiple-data instructions do not. In the worst case, this could produce integer-factor power-estimation errors for floating-point codes.
- *Pipeline and clock power errors.* Figure 2 shows the dynamic power that the pipeline contributes for in-order and OOO processors (65 nm), which is visible only by instrumenting the McPAT source code. This component of power is incorrectly dropped

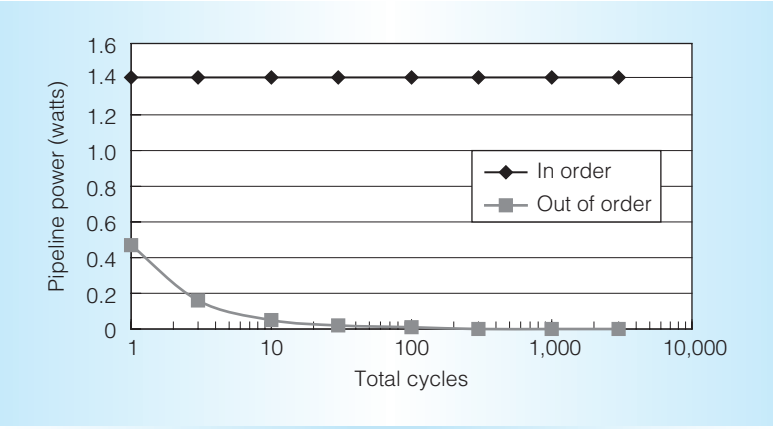


Figure 2. McPAT V1.1 pipeline power for an idle in-order and out-of-order (OOO) processor at 65 nm for different total execution cycles. The OOO core's pipeline power estimate incorrectly drops as the total cycles increases.

for all OOO core experiments lasting longer than a few cycles. This error is in versions of McPAT up to v1.1 (March 2014) and was fixed after our conversations with the developers. Nevertheless, it has led to errors in power breakdowns in many published papers.

The major implications of these errors is that without understanding whether the simulator is correctly capturing the phenomenon a designer is interested in, off-the-shelf usage renders them ineffective for even first-order analysis of effects. Thus, we suggest that researchers validate and sanity check the simulator with their own workloads and microbenchmarks. Inspired by the work of Rajagopalan Desikan, Doug Burger, and Stephen W. Keckler,[2] we developed a micro benchmark suite (www.cs.wisc.edu/vertical/microbench) that can serve as a template or starting point for validation of general-purpose processors. Alternatively, when it makes sense, we suggest considering evaluation with first-order models with known abstractions.

*Pitfall 2b: The trends myth.* An established argument in our field is commonly used to justify the use of a simulator as a black box, ignoring potential underlying or hidden errors. It is casually stated as, "Although specific details of the simulation are wrong, the overall trends will be correct." More specifically, the argument holds that relative performance comparisons may be correct, even if there is absolute error caused by a poor assumption or bug. However, for this to be true, the new technique being evaluated through simulation must be insulated from or statistically uncorrelated with the source of simulation errors. Because simulators can have significant errors, which are completely unknown, only in rare cases can we be sure this argument holds. Therefore, we call this argument the "trends myth." To explain, we give an example of an error from gem5 and explain why the trends myth is a fallacy.

Consider gem5's inconsistent pipeline-replay mechanism. The gem5 OOO model for pipeline replay allows cache misses to stall the pipeline, but it causes a pipeline flush if the cache has no available miss status holding registers (MSHRs). This is inconsistent, because both are mispeculations on the pipeline schedule for variable-latency operations and would use the same recovery mechanism. Also, the pipeline is repeatedly flushed on long-latency cache blocks, leading to significant and unnecessary reflushing overhead.

Consider a hypothetical research technique that reduces the penalty of pipeline flushes, and suppose it is evaluated on memory-intensive benchmarks that fully use the MSHRs. The trends argument would say that the new technique's high relative payoff would be representative of true designs. However, we know this to be false, because the reason for the extra flushes is an error in the pipeline scheduling model. Thus, we suggest that the trends myth not be used as an excuse not to validate and sanity-check simulators.

*Pitfall 2c: False confidence from validation.* Validation data, though typically factually presented by tool developers or other validation papers, is often misinterpreted by readers. A common misconception is that if the parameters are changed and configured for some other design point, the accuracy will be similar. This belief perhaps comes from a misunderstanding about how simulator validation is usually performed; tool validation is often carried out by fitting parameters to the "specific" validation targets, not about ensuring the

underlying modeling is accurate for individual phenomena or their interactions.

Consider McPAT's modeling of multicore power: the tool paper offers impressive validation data—around 20 percent or less error for total processor power for four designs, and even less for power breakdowns.[4] However, according to their own documentation and code comments, the researchers sometimes chose constants to match the validation targets. For example, McPAT assumes a constant dynamic power usage for idle functional units in OOO cores, citing "average numbers from Intel 4G and 773 Mhz (Wattch)."[4] For in-order cores, this same parameter is zeroed out. A related example is that a functional unit's per-access energy is divided exactly in half if the processor is of the type "embedded" ("According to ARM data embedded processor has much lower per acc energy."[4]).

We agree that this is a reasonable decision for features such as highly custom functional units. The danger is when researchers attempt to generalize the results or use McPAT simply by changing its configuration parameters; these constants likely will not be appropriate. Thus, we suggest that simulator authors provide methodological details and consider including an implication statement on what the validation implies for the tool's usage outside the validation points. Simulator users should not rely on validation to irrelevant design points.

### Pitfall 3: Detrimental evaluation standards

The proliferation of the described tools, and the misconceptions that spread with them, are having several harmful effects on quantitative evaluation standards.

*Pitfall 3a: Useful analysis does not see the light of day.* Often, works that use alternative analysis with simpler models to fairly capture first-order effects are not considered publication worthy. We include here some examples of works that were not published but that we feel should be even more highly valued. Guz's "Stay Away From the Valley" work is an example of insightful performance analysis of GPUs and CPUs without considering any simulation.[9] The simulation-compliant version was ultimately published in the IEEE International Conference on Computer Design (ICCD) proceedings.[10] Even when authors take an extremely detailed approach (for example, the complete RTL development, synthesis, place-and-route in the Efficient Low-Power Microprocessor[11]), correctly using nonstandard benchmarks and metrics that best capture the research idea can be frowned upon in the review process. Thus, we suggest that the community encourage authors, when appropriate, to use first-order models for research and publications.

*Pitfall 3b: Misuse and overuse of cycle-level models.* Contrarily, in some cases, we believe a first-order model would suffice, but the authors apply (in our opinion) unnecessary simulation and associated power/energy models. In the recently published "Neural Acceleration for General-Purpose Approximate Programs," the authors use cycle-level simulation and McPAT energy estimation of the core with a high-level energy estimation of the NPU unit.[12] From an external perspective, this mix of detail and abstraction makes the result's implications more difficult to understand, and we believe that analysis of simpler models could have provided further insights. If our community was welcoming of less detailed modeling, such papers could focus on high-level models, analysis, and well-reasoned arguments, and ultimately provide more insight.

*Pitfall 3c: Underappreciation for first-order analysis.* Regardless of the particular approach, the most important role of an evaluation is to provide insight into the effects of underlying proposed mechanisms or techniques. Simulators can create the temptation to generate an abundance of data and summary statistics suggesting large improvements without actually explaining the reasons and intuition for the benefits. In such cases, a lack of first-order analysis of effects can obfuscate technical conceptual flaws, leading to reduced-quality research findings.

We suggest that a first-order analysis of a technique's effects should be required, regardless of the evaluation approach. Unless the authors can demonstrate with a simple first-order analysis why something works, demonstrating improvements on certain metrics of interest is not useful.

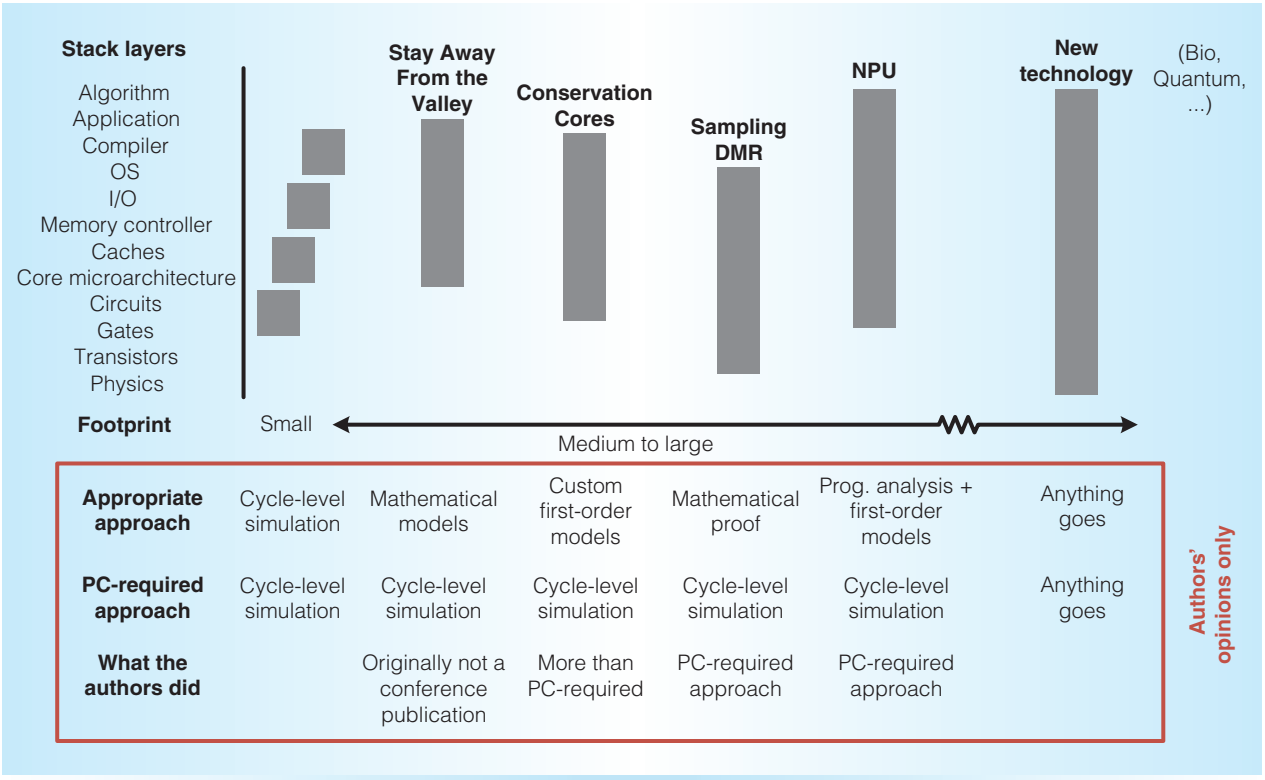


Figure 3. Footprints and approaches for various architectural research. Gray boxes indicate the footprint for several items of architectural research. For each technique, the figure describes our opinion of the appropriate approach, the standard or templatized approach required by the Program Committee (PC), and how the authors coped with the discrepancy.

*Pitfall 3d: Amplified reviewer noise.* Members of industry and senior researchers often say that they do not focus on the evaluation section but instead "go for the big idea." Yet, not all reviewers reflect this magnanimous attitude (as we have observed from many Program Committee meetings). The broader issue is an unequal standard in evaluation methodology and how reviewer assignment unnecessarily plays a role. Why should the variable tastes of potential reviewers play a significant role in a paper's outcome?

We suggest that the community create a common standard on what is expected and reasonable. In a small community such as ours, this should not be difficult. Co-reviewers and program chairs should confront any reviewer who inappropriately harps on quantitative detail early in the review process.

## Calibrating evaluation standards

To help our community calibrate evaluation expectations and avoid overuse of architectural simulation, we present a simple concept for understanding a research idea's scope: the footprint, the span of the stack layers a research idea affects or is influenced by. Figure 3 depicts the footprint for several research papers.

The first row of Figure 3 describes what we believe would have been the best approach for evaluation. On the left, when an idea's footprint is small (not to be confused with incremental), there is value in detailed quantitative simulation by considering a design point close to validated design points from existing tools. One example is sampled temporal memory streaming.[13]

In the middle and right of Figure 3, when the footprint is large, first-order effect modeling of various sorts should be considered sufficient and encouraged. Right now, we believe our field does not have enough appreciation of an approach's footprint. Instead, in all but the most speculative research based on fundamentally new technology, the standard

or templatized approach is cycle-level simulation (second row of the figure).

The final row of Figure 3 describes the taken approach. First, the authors of the "Stay Away from the Valley" paper[9] were not able to publish the non-simulator-enhanced version in a conference venue, which we believe was partly because they used mathematical models rather than simulation. For Conservation Cores,[14] we think a first-order energy and performance model would have sufficed. The authors chose to develop a fully placed-and-routed design with parasitic extraction and used toggle-level simulation from this (and Cacti modeling) to obtain performance and energy. Their methodology was outside the mainstream and, in our opinion, exceeds the credibility of cycle-level simulation. However, the authors were subjected to reviewer nitpicking on evaluation details, which we diagnose as miscalibrated expectations. For the sampling DMR work,[15] where simple math showed the technique had almost negligible overhead, reviewers compelled authors to spend weeks on unimportant simulator implementation and evaluation. Ultimately, the simulation "proved" the simple math.

Overall, authors studying cross-cutting phenomena often are compelled to spend significant time in conforming to templatized expectations. If reviewers can tailor their expectations to match a technique's footprint, authors can be free to use the most appropriate approach for their respective problems.

In addition to addressing the pitfalls of simulation, in terms of their development, use, and community impact, we suggest that there are several further opportunities. Community-based evaluation tools benefit greatly from an active community that continually fixes bugs and makes improvements, while errors in in-house tools are likely to remain hidden without a wider audience to shed light on issues. As simulators are only becoming more complex, community-driven tools seem ever more important. As a step forward, we also believe funding agencies should consider tool maintenance an important endeavor in the same league as physical infrastructure.

Our top conferences should avoid the "lack of novelty" mindset and encourage publishing tool papers that are "mere engineering" achievements. In particular, we should avoid building and investing in tools that are by design overfitted to one particular design point.

An interesting opportunity to consider might be how to build new types of first-order models that have the same type of community involvement as current-generation simulators. These new tools could help provide reliable insight for future research ideas, while continually improving and evolving.

Overall, as the architecture community branches out into new cross-stack research areas, now is an opportune time to rethink and debate how to promote our field's growth in the coming decades. MICRO


## Acknowledgments

We thank the anonymous reviewers, along with Thomas Wenisch, Steve Reinhardt, Gabriel Loh, Mark Hill, Nilanjan Goswami, Michael Taylor, Uri Weiser, Simha Sethumadhavan, Kathryn McKinley, Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, Doug Burger, Sheng Li, Norm Jouppi, Vijay Janapa Reddi, Nam Sung Kim, Jingwen Leng, Martha Kim, Gagan Gupta, Jason Powers, Zach Marzec, and Marc de Kruijf for their thoughts, experiences, criticism, and advice. A special thanks to Vamsi Krishna Kodati, Paul Gratz, Yingying Tian, Zhe Wang, Daniel Jimenez, Ping Xiang, Huiyang Zhou, Nilanjan Goswami, and Tao Li for their help with verification. The views expressed in this article are the authors' alone; none of those mentioned here either implicitly agree or disagree.


## References


1. E. Dijkstra, "Go To Statement Considered Harmful," *Comm. ACM*, vol. 11, no. 3, 1968, pp. 147–148.
2. R. Desikan, D. Burger, and S.W. Keckler, "Measuring Experimental Error in Microprocessor Simulation," *Proc. 28th Ann. Int'l Symp. Computer Architecture*, 2001, pp. 266–277.
3. N. Binkert et al., "The gem5 Simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, 2011, pp. 1–7.

4. S. Li et al., *McPAT 1.0: An Integrated Power, Area, and Timing Modeling Framework for Multicore Architectures*, tech. report HPL-2009-206, Hewlett-Packard Labs, 2009.
5. A Bakhoda et al., "Analyzing CUDA Workloads using a Detailed GPU Simulator," *Proc. IEEE Int'l Symp. Performance Analysis of Systems and Software*, 2009, pp. 163–174.
6. J. Leng et al., "GPUWattch: Enabling Energy Optimizations in GPGPUs," *Proc. 40th Ann. Int'l Symp. Computer Architecture*, 2013, pp. 487–498.
7. D. Sanchez and C. Kozyrakis, "ZSim: Fast and Accurate Microarchitectural Simulation of Thousand-Core Systems," *Proc. 40th Ann. Int'l Symp. Computer Architecture*, 2013, pp. 475–486.
8. T. Nowatzki et al., "gem5, GPGPUSim, McPAT, GPUWattch, 'Your Favorite Simulator Here' Considered Harmful," Workshop on Duplicating, Deconstructing and Debunking, 2014; www.cs.wisc.edu/vertical/talks/2014/wddd-sim-harmful.pptx.
9. Z. Guz et al., "Many-Core vs. Many-Thread Machines: Stay Away from the Valley," *IEEE Computer Architecture Letters*, vol. 8, no. 1, 2009, pp. 25–28.
10. Z. Guz et al., "Threads vs. Caches: Modeling the Behavior of Parallel Workloads," *Proc. IEEE Int'l Conf. Computer Design*, 2010, pp. 274–281.
11. W.J. Dally et al., "Efficient Embedded Computing," *Computer*, vol. 41, no. 7, 2008, pp. 27–32.
12. H. Esmaeilzadeh et al., "Neural Acceleration for General-Purpose Approximate Programs," *Proc. 45th Ann. IEEE/ACM Int'l Symp. Microarchitecture*, 2012, pp. 449–460.
13. T.F. Wenisch et al., "Practical Off-Chip Meta-Data for Temporal Memory Streaming," *Proc. 15th IEEE Symp. High Performance Computer Architecture*, 2009, pp. 79–90.
14. G. Venkatesh et al., "Conservation Cores: Reducing the Energy of Mature Computations," *Proc. 15th Int'l Conf. Architectural Support for Programming Languages and Operating Systems*, 2010, pp. 205–218.
15. S. Nomura et al., "Sampling + DMR: Practical and Low-Overhead Permanent Fault Detection," *Proc. 38th Ann. Int'l Symp. Computer Architecture*, 2011, pp. 201–212.

**Tony Nowatzki** is a PhD student in the Department of Computer Sciences at the University of Wisconsin–Madison and a member of the Vertical Research Group. His research interests include architecture and compiler codesign and mathematical modeling. Nowatzki has an MS in computer science from the University of Wisconsin–Madison. He is a student member of IEEE. Contact him at tjn@cs.wisc.edu.

**Jaikrishnan Menon** is a software engineer at Intel. He has an MS in computer science from the University of Wisconsin–Madison, where he completed the work for this article. Contact him at menon@cs.wisc.edu.

**Chen-Han Ho** is a research engineer at Qualcomm. His research focuses on energy-efficient computer architecture, software-hardware codesign of accelerators, RTL design, and FPGA prototyping. Ho has a PhD in computer architecture from the University of Wisconsin–Madison, where he completed the work for this article. Contact him at ho9@wisc.edu.

**Karthikeyan Sankaralingam** is an associate professor in the Department of Computer Sciences and the Department of Electrical and Computer Engineering at the University of Wisconsin–Madison, where he also leads the Vertical Research Group. His research interests include microarchitecture, architecture, and very large-scale integration. Sankaralingam has a PhD in computer science from the University of Texas at Austin. He is a senior member of IEEE. Contact him at karu@cs.wisc.edu.