gem5, GPGPUSim, McPAT, GPUWattch, "Your favorite simulator here" Considered Harmful

Tony Nowatzki Jaikrishnan Menon Chen-Han Ho Karthikeyan Sankaralingam
University of Wisconsin - Madison
tjn@cs.wisc.edu menon@cs.wisc.edu ho9@wisc.edu karu@cs.wisc.edu

Much as Dijkstra, in 1968, observed the dangers of relying on the go to statement, we observe that the dominant reliance on quantitative simulators is having a detrimental effect on our field. Over time, simulator tools have become more sophisticated. From the simple days of the now debunked SimpleScalar with its RUU-based OOO model with fixed DRAM latency, to the gem5+DramSim+GPGPUSim+McPAT mashup simulator, we have come a long way in what architects are claiming as validated tools. We argue, though, that new generations of simulators are often overfitted to certain benchmarks or configurations for validation and can have significant modeling errors that researchers are not aware of. Though the existence of these errors are unsurprising, they can cause unaware users to derive incorrect conclusions. Simultaneously, and even more problematic, is that reviewers demand researchers inappropriately use these tools. We enumerate eight common, but not acknowledged or recognized pitfalls of simulators or simulator use, considering four modern simulation infrastructures. We propose that the evaluation standards for a work should match it's "footprint," the breadth of layers which the technique affects, and conclude with our opinion on how to escape out of our field's simulate-or-reject mindset.

1. Introduction

For a number of years we have been familiar with the observation that the quality of ~~programmers~~ *architecture researchers* is a decreasing function of the ~~density of go to statements~~ *reliance on quantitative architecture simulators* in the ~~programs~~ *architecture papers* they produce. More recently we discovered why the use of ~~the go to statement~~ *architecture simulators* has such disastrous effects, and we became convinced that the ~~go to statement~~ *architecture simulator* should be abolished from all "higher level" ~~programming languages~~ *architecture research*. At that time we did not attach too much importance to this discovery; we now submit our considerations for publication[1]. Much as Dijkstra observed the era of reliance on the go to statement was having a negative effect, we observe the era of over-reliance on quantitative simulators is having a detrimental effect on the field, and should come to an end.

We observe that simulation, in particular "detailed" tools that provide cycle-accurate performance estimates, area estimates, power and energy estimates, as a vehicle for architecture research is ubiquitous. From the simple days of the now debunked SimpleScalar [8] with its RUU-based OOO model + fixed memory to gem5+DramSim+GPGPUSim+McPAT mashup simulator, we have come a long way in what architects are claiming as validated simulators[2]. This level of added detail has led to the belief that we have better tools and are doing better and better quantitative evaluation. It has also led to the preponderance of papers relying on such tools and has created an implicit standard and template of how quantitative evaluation must be done. This reliance and belief in such detailed tools is hurting the field and creating various pitfalls.

Part of the problem is that these tools are commonly overfitted for validation, meaning that their parameters are tuned such that they are accurate only on a small set of benchmarks or configuration parameters. The implication of overfitting is that simulator models capture the noise rather than the fundamental relationships and tradeoffs. In addition, simulator tools often have significant modeling errors which are not easily

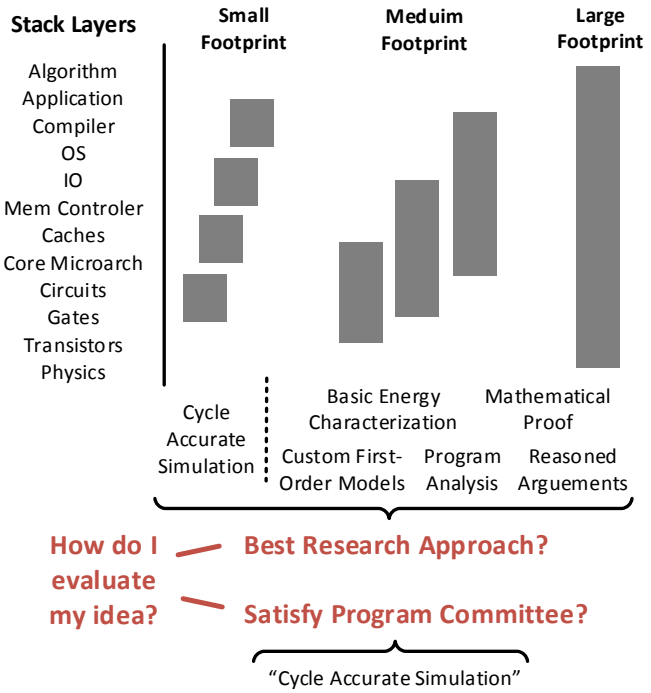

Figure 1: The *footprint* of a technique (the scope of layers it interacts with), and the choice researchers face between appropriate evaluation and PC-compliant evaluation practices.

[1] This paragraph is reproduced and criticisms are modified from Dijkstra's seminal *A Case against the GO TO Statement [10]*. Additions are in *italics*.

[2] We remark that not all simulators' authors themselves claim validation.

accessible by users. Overall, the reliance on simulators are creating many pitfalls both in technical aspects and in hurting the field by distorting reviewer expectations of what entails good quantitative evaluation.

As a way forward for researchers, we believe that the correct approach depends on the *footprint*, or layers of the stack which the technique affects or relies on, and that there is no one-size-fits-all solution to architecture research. Figure 1 highlights how different techniques, represented by gray boxes, can affect different stack layers. Unfortunately, it is too often the case that researchers make the choice of research approach based on what will get their paper accepted, rather than what is the most scientific. We revisit a version of this figure with specific examples in Section 4. Most importantly, we believe that reviewers must recalibrate their evaluation standards, and appropriately gauge them to the footprint of the research. We believe this issue is important and vital now as more research in our field is moving toward larger footprints, evidenced by recent keynotes [7] and funding calls [21]. Restricting ourselves to an ill-suited one-size-fits all approach could curtail scientific advancement of these efforts.

This paper enumerates eight common, but not acknowledged or recognized pitfalls, considering four modern simulation infrastructures: gem5 [5], McPAT [19], GPGPUSim V2.x [3], and GPUWattch [18]. We begin this paper with a section describing errors in popular simulators, which we use to substantiate the pitfalls. In discussing simulator errors and pitfalls, our goal is not to offend or criticize but to inform and provoke thoughtful discussion. We conclude with strategies which can allow us to escape out of our field's templatized simulate-or-reject mindset.

2. Errors in simulators

We begin by first outlining some example instances of errors in mainstream and popular simulators. We believe the existence of these errors should neither be surprising, nor are they intended as an attack on particular simulators or simulator authors; any large body of code will have errors. We only bring attention to add some context to our pitfalls and aid in substantiation. If anything, our criticism is squarely aimed at users of such tools, for example Govindaraju et al. [12]. Error reports are available at `http://www.cs.wisc.edu/vertical/sim-harmful`, which have been verified by at least one other person not affiliated with our research group. Their purpose is to point out the type of problems which can be detrimental if users are not aware. In this section, for each tool, we first present observations about an issue, then give our opinions the issue's implications.

2.1. gem5

To be clear, the errors discussed in this section have only been verified on the X86 version of gem5, and the micro-op issues can only apply to X86. Also, some of the below errors have been communicated to the quite active gem5 community, and we believe these problems can be tackled without difficulty. We revisit the benefits of community driven tools in Section 4.2.

Conservative/Obscure Default for Writeback Mechanism: The gem5 OOO model only schedules instructions for issue if there are guaranteed to be enough "writeback buffers" for them, where the total buffers are calculated by writeback-width $\times$ writeback-depth. The default, across all ISAs, is a writeback-depth of 1. This means that if a few long latency instructions hold up writeback-buffer slots, then the effective issue width goes to 0. For an OOO 2-wide core with benchmarks that have long-latency memory references, adding 5 buffer slots increases performance by more than 5X. We do not believe this tradeoff is representative of real OOO designs, and this important parameter is not sufficiently defined in the documentation or source code.

Inconsistent Pipeline Replay Mechanism: gem5's OOO model for speculative instruction scheduling and pipeline replay appears to be both contradictory and unnecessarily conservative. To explain, a deeply pipelined OOO core must speculatively schedule instructions to enable back-to-back execution. When an unexpected latency occurs, the schedule for the miss-dependent instructions needs to be corrected. In gem5, when a load issues to a blocked cache, gem5 conservatively models the "correction" to the speculative schedule by flushing the entire pipeline. The larger issue is that after a pipeline flush, instructions are immediately rescheduled, even if the cache remains blocked. This leads to a cycle of repeated flushing of the entire pipeline. While the performance does not take a significant hit, the amount of energy can double on some benchmarks versus a design with a handful more MSHRs to prevent the cache from blocking.

To be consistent, an architecture which flushes the pipeline on a cache-block should also flush the pipeline on other variable-latency events. However, gem5 does not flush the pipeline on events like cache misses, which would have variable latency. In short, the pipeline replay mechanism is simultaneously both highly conservative and optimistic.

Inefficient/Mislabeled Micro-ops: gem5 micro-ops are optimized more for correctness and economy rather than efficiency. One example is that the same micro-op that performs conditional moves also performs regular register moves. This means that regular moves will incur the dynamic dependence and energy cost of reading the destination register, even though they are completely overwriting it. Also, though the gem5 flag register implementation has greatly improved in recent versions, a few instructions still require extra dependencies and register reads because of flag register grouping. One example is how logical instructions (like `XOR`) don't write the `AF` flag, but since it is grouped with the other flags, it must be read before written. This is arguably acceptable, but difficult to understand and access as a user.

An important yet fixable problem is that some micro-ops are

outright mislabeled. For example, both of these instructions are labeled as requiring the FP ALU unit: `mov2fp` which moves values from floating point to integer registers (no fp conversion), and `ldfp` which loads a value from memory into a floating point register. SIMD floating point multiplies, along with other SIMD instructions, do not show up in the final statistics. Without fixing these bugs, using the data from gem5 to perform energy analysis on floating point code would produce incorrect results by potentially integer factors.

2.2. McPAT

Unclear/Overfitted Functional Unit (FU) Energy Modeling: In the McPAT model, if the core is OOO, then a small dynamic component of energy is added for each FU regardless of whether the FU is being used. This constant is cited as "average numbers from Intel 4G and 773Mhz (Wattch)". Why this occurs in OOO but not Inorder processors could be due to overfitting in validation. Another related example is for the per-access energy of an FU. If the processor is "embedded," then this power is divided by two, citing: "According to ARM data embedded processor has much lower per acc energy". Whether or not these (in our opinion) seemingly arbitrary decisions are valid or not, since they are not easily accessible or decipherable by the user, they may come to incorrect conclusions about the quantitative results.

Error in Pipeline and Clock Power McPAT calculates an estimate of the pipeline and clock power considering switching factors in pipeline flip-flops. This power is not reported directly in McPAT, rather, it is distributed equally amongst the various processor structures, making it difficult to determine when there are errors. Figure 2 shows the dynamic power which the pipeline contributes for inorder and OOO processors (65nm), which can only be seen by instrumenting the McPAT source code. Our experiments show that this component of power is effectively dropped for all OOO core experiments lasting longer than a few cycles. The error appears to be introduced when converting between power and energy, where a factor of the number of cycles is lost for the OOO core only. This apparent error is in all versions of McPAT that we tested (from v0.7 to v1.1 March 2014). The implication of this error is that it creates uncertainty about the estimation of pipeline and clock power[3].

2.3. GPGPUSim V2.x

In this subsection, we consider a widely adopted version of the GPGPUSim tool, and describe several missing or abstracted components of its architectural model. We discuss GPGPUSim V2.x, even though it is not the latest version of the tool, specifically because many researchers are still using this version [16, 17], and we believe the following claims about

[3]For those aware of the exact details of McPAT, when it is used in a fine-grained mode (called every cycle - as opposed to the XML interface of calling at end of millions of cycles of simulation), this issue will disappear. However, the XML bulk mode is the most prevalent usage of McPAT in literature.

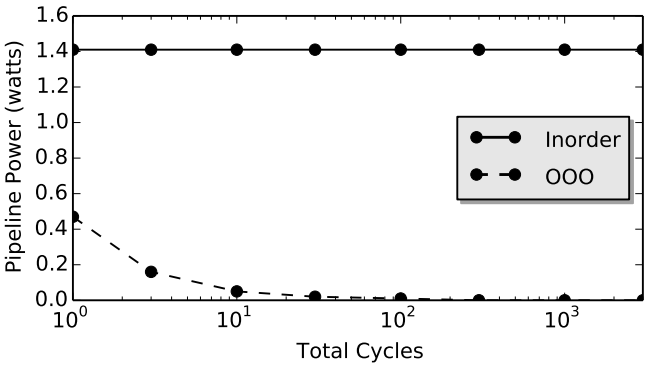

Figure 2: McPAT pipeline power for a 65nm idle processor.

its modeling features can be made without controversy. GPGPUSim 3.x has fixed many of these issues, (see slide 20 in the tutorial [2]).

- **Register File microarchitecture:** The operand collector (single-ported register file banks + arbiter + X-bar + collector units) is modeled assuming fixed latency accesses to the SRAM with some additional queuing latency. It does not model low-level details, like contention, which impact performance in high-compute bandwidth scenarios.
- **Thread/warp/wavefront scheduling and dispatch:** Thread scheduling is functional, and while a number of different warp scheduling schemes are implemented, these are not modeled in the microarchitecture, they are simply generated functionally.
- **Branch divergence structures and 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 microarchitectural feasibility of the proposed technique. For example, consider developing and evaluating a non-trivial warp scheduling technique in GPGPUSim V2.x. Its model would be a functional one, meaning that it would not capture the individual components of the hardware, their communication, or their pipeline stages. This would be tantamount to a CPU load-store queue design evaluation which functionally models the dependence predictor, while ignoring cache port contention etc. For the CPU domain, this mashup of high-level modeling and low-level simulation would not be considered sufficient to understand the effectiveness of a technique quantitatively.

2.4. GPUWattch

Given the straightforward reading of the GPUWattch [18] paper, its methodology has a form of modeling error which we call "mathematically irrelevant" modeling. We define this as model which, when taken as a whole, contains mathematically-irrelevant sub-components. The first part of this subsection

will describe how this form of error applies to the methodology (as presented) in Leng et al. [18]. Essentially, the *detailed modeling* using McPAT, empirical memory models and synthesis based models aren't meaningful to the final obtained prediction. However, additional unpresented details of the GPUWattch methodology help justify the detailed modeling. Therefore, we will subsequently discuss some of these details, and conclude with the implications for appropriate model usage.

GPUWattch Power Modeling (as presented) GPUWattch models the cycle-level power of GPU architectures by first using GPGPUSim to obtain activity factors. Then, GPUWattch calculates the dynamic power of a particular benchmark P_{bench}, as the sum of the activity factors $\alpha_{bench,comp}$, multiplied by the maximum power of the component, $PMAX_{comp}$. The $PMAX_{comp}$ power value is obtained through highly detailed modeling using a combination of McPAT-based modeling, empirical models, and synthesis-based models. The GPUWattch authors state that since McPAT is tuned to CPUs, and since there are many undocumented GPU features, they need to correct for this by adding an error term for each component x_{comp}, and use least-squares estimation (linear regression) to estimate the errors. Their final model for dynamic power is:

$$P_{bench} = \sum_{\forall comp} \alpha_{bench,comp} \times PMAX_{comp} \times x_{comp} \qquad (1)$$

In linear regression terminology, $\alpha_{bench,comp} \times PMAX_{comp}$ are the explanatory or input variables, x_{comp} are the regression coefficients and P_{bench} is the dependent variable. At this point, the authors' methodology is as follows:

We iteratively refine the power model on the basis of the sources of the various inaccuracies that LSE [regression] identifies. For instance, in our infrastructure (i.e., McPAT) the power estimation for certain components is biased toward CPU implementations. We narrow the resulting inaccuracy gap for the GPU power model by fixing our initial assumptions about the implementation and then applying the scaling factors that are obtained from LSE.

We contend that a direct interpretation of their methodology would be to run the regression in Equation 1 using measured values of P_{bench} of some microbenchmarks to find the component wise errors x_{comp}, then modify the source code to multiply the original component wise power $\alpha_{bench,comp} \times PMAX_{comp}$ by the "scaling factor" for that component, x_{comp}, to obtain the final power estimate. Note that this procedure is performed on a platform specific basis[4].

[4]Considering the XML files provided in the tool for GTX480 and QuadroFX5600, the scaling coefficients are the series of 32 param names starting at line 31 (`TOT_INST, FP_INT, IC_H, etc.`) In the source code, in `gpgpu_sim_wrapper.c`, these are used in methods like `set_inst_power`, `set_regfile_power` etc. to scale up the McPAT computed values.

If this methodology was directly applied, the computation of *PMAX* is mathematically irrelevant. Performing a linear transformation on the explanatory variables of a linear regression does not affect the error or prediction accuracy. In fact, running the below regression, which does not have $PMAX_{comp}$ values, would be mathematically equivalent, and the resulting regression coefficients are simply scaled as follows: $x'_{comp} = PMAX_{comp} \times x_{comp}$.

$$P_{bench} = \sum_{\forall comp} \alpha_{bench,comp} \times x'_{comp} \qquad (2)$$

What this means is that the $PMAX_{comp}$ variables are mathematically meaningless to the final model. Therefore, users who apply the GPUWattch methodology as written will put unnecessary effort into detailed power modeling (which would include McPAT, empirical model and synthesis model development).

GPUWattch Power Modeling (as implemented) The methodology which is implemented actually does employ the detailed power modeling results during the scaling parameter selection for some purpose, as we explain next[5].

First, instead of scaling the internal power values of McPAT's optimization framework, what is actually scaled are the activity counts which are fed as inputs to McPAT. This assumes that McPAT's choice of components would be unaffected by the different power scaling factor applied to various components.

Second, instead of automatic linear regression, they calculate the root mean square error of their predictions and manually modify the scaling coefficients to reduce the error. This alone would be the manual equivalent to linear regression, and hence would still have the mathematical irrelevance bug. However, the authors also bound the scaling coefficients by between 10× to 50× for on-core and off-core components respectively (here, the authors explain that the bound is chosen based on the confidence in the original detailed model). The authors observe that *without* bounding the scaling coefficients, the error is actually less: a mathematically "better" model. However, the per-component breakdowns with pure linear regression do not match expected intuition (too big or negative scaling factors). Therefore, the bounds on scaling factors serve as a rough guideline in attaining a plausible power distribution. Overall, we believe it is possible to use a purely mathematical approach, applying the same type of rough intuition, to achieve the same quality of results without detailed power modeling like McPAT.

What are GPUWattch's appropriate use cases? The methodology behind the GPUWattch power model has implications for its appropriate usage. Our position is that it that it can only be appropriately employed when a physical artifact with measurable power numbers are available. For the

[5]The authors graciously provided us details on their methodology, and we take the blame if we have made any mistakes in reproducing it here.

two platforms which have configurations now, the GTX480 was released in 2010, and the the Quadro FX5600 is even older. Generating a new GPUWattch configuration requires attaining detailed power measurements, including physically instrumenting the GPU power supply with sensing resistors, followed by an application of the manual error-minimization procedure described above.

The reason why a physical artifact is necessary is that the scaling factors, x_{comp}, are platform specific. As an example, consider the power of register file access in the GTX480 and FX5600. The McPAT scaling between the two designs does not capture their architectural differences, which shows up in the GPUWattch model as the ratio of their register file scaling factors, which is $1.7\times$.

If we want to consider a hypothetical GPU with different configuration parameters, without changing the scaling factors, we should not expect the GPUWattch power model to be valid. To explain, the average scaling factor magnitude is $22\times$ for the GTX480, and $8\times$ for Quadro FX5600. To claim that the hypothetical GPU configuration is valid, the argument that would have to be made is that McPAT gets the power *wrong* by an order of magnitude, but somehow can get the relative scaling of components correct. This is a position we believe is untenable without evidence. The lack of validated configurability would impede an accurate design space exploration.

Good uses for GPUWattch would include estimating the energy impact of policy changes which affect the activity factors, or in adding components which have externally validated power characteristics (again, if the target architecture already has a GPUWattch power model). Revisiting the concept of footprint, these are both small-footprint evaluation scenarios. We clarify here that the authors of GPUWattch never mention design space exploration as something their tool is meant for. So again, our criticism is aimed at tool users rather than developers, and additionally the reviewer who now thinks energy estimation for GPU research is always doable.

3. Pitfalls

This section describes eight pitfalls of modern simulators and simulator usage. For each pitfall, we describe the high-level problem and substantiate our position with empirical evidence. We then give our opinions on how best to avoid the pitfall.

3.1. Pitfall 1: Errors in simulators are inaccessible to users

As outlined above, simulator tools can have significant abstraction, modeling, and specification errors. Furthermore, since simulators are distributed as C/C++ code with little specification, it is difficult for end users to even become aware of these errors. Without understanding whether the simulator is correctly capturing the particular phenomenon a designer is interested in, off-the-shelf usage renders them ineffective for even first-order analysis of effects.

Sometimes, the features of modeling tools are not easily accessible, making bugs difficult to find even with careful data analysis. That is because many of the features are obscured behind implicit assumptions, lack of documentation and lack of good reporting of results. One example is how the pipeline power is reported in McPAT. Since it is implicitly distributed amongst the individual components of the processor, what appears to be a significant error is obfuscated. Errors like this put researchers in a difficult position. Should they go fix the tool which is already validated? And what if another error in the opposite direction is canceling out the effects?

Suggestions: Authors should first validate and sanity check the simulator individually. Further, when it makes sense, they should consider building trace-driven tools that model the first order effects they are aware off, instead of using cycle-accurate tools. We believe it is better to have a tool with known abstraction errors than an unknown black box. Reviewers and the community needs to change its mindset as well – having blind faith in "standard tools," while completely discounting other tools is not appropriate. We revisit the issue of open versus in-house tools in Section 4.

3.2. Pitfall 2: False confidence from validation - over-generalization in simulator papers, or tool misuses

Simulator writers typically make narrow and factually consistent statements about validation, and some examples are below. However, the nature of validation is often misunderstood by users, and these tools are put to use in ways not intended for, including making quantitative generalizations.

gem5's OOO model is widely used, but as observed in a recent paper [13] and our observations above, it has several specification errors. Though the gem5 authors themselves do not claim it as such, some do claim it is a "validated simulator." Clearly, this cannot be taken as all effects modeled. For instance, a technique that works on the instruction front-end must pay attention to gem5's baseline and first fix the specification error described here [13].

Considering McPAT, according to their own documentation and code comments, constants are sometimes chosen to match the validation targets. We agree this is a reasonable decision in some cases, especially when highly customized logic is employed (e.g. functional unit implementations). The danger is when researchers attempt to generalize the results outside the validated processors. These constants will likely not be appropriate.

For GPUWattch, it might be tempting for researchers to perform sensitivity studies by varying McPAT parameters. The path of least resistance would be to use the same scaling factors, instead of measuring the power of a known GPU and deriving new scaling factors using the GPUWattch methodology. For reasons described in the previous section, we argue that without obtaining new scaling factors, this type of sensitivity analysis would be inappropriate.

Suggestions: Use with caution validated simulators. Look for details on the simulator's design and factor those decisions

into the implications of the numbers reported by simulator. For simulator authors, provide guidance on things not *to use the tool for, and provide methodological details in the user documentation, even if they did not appear in the conference publications.*

3.3. Pitfall 3: Architectural abstraction decisions with unknown impact on effects.

In general, simulators model certain things in great detail, while abstracting away others which could have big impact. From an end-user standpoint, the decisions regarding what is modeled in detail may be inappropriate depending on what is required. There is an implicit statement in typical tutorials and documentation that unmodeled things have little first order impact. However, this is often not the case and can misguide researchers. Furthermore, this leads to a fundamental simulator paradox - if getting at first order effects is what is important, why bother with such a low-level quantitative cycle-by-cycle simulation in the first place?

Considering gem5, it is very detailed in some regards, like the full conversion to micro-ops, and "execute in execute" modeling, but looser in other respects, like no modeling of register ports. As observed by Gutierrez et al. [13], gem5 has a specification error in the fetch buffer design. From a simulator user standpoint, this does not lead to any change in performance, however it drastically increases the number of instruction cache accesses. As they describe, if a research idea is purported to reduce the number of cache accesses to save power, without being aware of the fetch buffer, its results would be skewed.

Recently, GPGPUSim V3.x addresses many of the missing modeling features of GPGPUSim V2.x, as outlined in Section 2.3. This raises an interesting question: for the research which used GPGPUSim V2.x to evaluate architectural enhancements involving exactly those missing features, is their evaluation meaningful enough to warrant the effort of simulator modifications and detailed cycle-accurate simulation?

To comment on GPUWattch, it models certain components of power, but lacks others, like the energy of double precision register file access. This particular modeling choice is acceptable if the target GPU is the NVIDIA GTX480, which intentionally has reduced double precision throughput (this is to differentiate between consumer and professional card versions). However, users may want to apply the GPUWattch methodology for products intended for double-precision capability, like the Quadro FX5800 (supported by GPGPUSim). If researchers use this platform to evaluate how their ideas affect the register file power of double-precision benchmarks without modifying the McPAT code, then their results will be skewed. While in general it is acceptable to make these sort of modeling abstractions, they should be made explicit to users in the tool papers and documentation.

Overall, the presence of detail in some places, but lack thereof in others, drives the "trends myth" (Pitfall 7).

Suggestions: For simulator writers, document the sources of abstraction decisions. In many cases, authors are well aware but do not spell-out these decisions, since our community treats these as a weakness. Knowledge of what is missing is far better than ignorance.

3.4. Pitfall 4: Detrimental quantitative evaluation standards

The proliferation of the described tools, and the misconceptions which spread with them, are having harmful effects on the quantitative evaluation standards of our community. The problem here is multifold – it causes first-order modeling to be mislabeled and called cycle-accurate, cycle-accurate modeling to be used when first order modeling would have provided better insight, and prevents researchers who sensibly use higher-level modeling from publishing. We describe these effects in more detail.

Fear of saying first-order model: In many cases, authors appropriately perform a first-order evaluation, but incorrectly label and advertise as cycle-accurate. This is a disservice to our community. An example is the Large-Warp Microarchitecture paper, and we reproduce below the description of their methodology:

We use a cycle accurate simulator that simulates parallel x86 threads, each executing the same compute kernel. In our results, we simulate a single GPU core concurrently executing 1024 threads. Table 1 presents the system parameters used in our simulations for the baseline processor. Since x86 does not have instructions to aid with branch divergence/reconvergence of parallel threads like GPU IS As do [21], we created instrumentation tools to identify conditional branch instructions and their control flow merge points. We used a similar procedure to identify barrier synchronization points since x86 does not support single instruction barrier synchronization present in GPU ISAs [21].

On careful reading this is a first-order model - what else would one call something that uses only 1024 threads, uses x86 ISA to model GPU machine code, and retro-fits effects of branch divergence instead of doing execution driven simulation? However, the authors, presumably to satisfy reviewers, call this cycle-accurate. We question what this type of approach is "cycle-accurate" to. In our opinion such models are sufficient, and our community should allow us to call them what they are: *first order models*. In such cases, the review process is rewarding the rhetorical skills of authors, not scientific exposition.

Misuse of cycle-accurate models: Contrarily, there are cases where we believe a first-order model would suffice, but the authors apply (in our opinion) unnecessary "cycle-accurate" simulators and associated power/energy models. In the recently published "Neural Acceleration for General-Purpose

Approximate Programs", the authors mix cycle accurate simulation + McPAT energy estimation of the core with a high-level energy estimation of the NPU unit [11]. From an external perspective, this mix of detail and abstraction seems arbitrary, and we believe the time spent in performing this detailed evaluation could have been better spent in further analysis of simpler models.

If our community was welcoming of less detailed modeling, such papers could focus on high-level models and well-reasoned arguments, and ultimately provide more insight for readers.

Useful analysis does not see the light of day Often authors use templatized simulator evaluation, not for scientific reasons, and the alternative of analysis using simpler models to fairly capture first-order effects is not considered publication worthy. Some examples of works which *did* get published follow, which 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 [15]. The templatized, simulation-compliant version was ultimately published as an ICCD paper [14]. On the other hand, even when authors take an extremely detailed approach (complete RTL development, synthesis, place-and-route), correctly using non-standard benchmarks and metrics that best capture the research idea can be frowned upon in the review process. ELM [4, 9] is great example of this.

Suggestions: We encourage reviewers, before commenting on tools, to understand in detail what these tools provide. Just skimming the tools paper and filling in the details is insufficient. For authors, when appropriate, we encourage them to boldly go where few have gone before and rely on first-order models for research and publications.

3.5. Pitfall 5: Researchers being guided by poor-quality quantitative findings.

A corollary to the above pitfall is that modeling errors and simulator reliance can lead to reduced-quality findings. Most of us have read papers with technical conceptual flaws that are obfuscated by an inundation of quantitative data. In this paper, we avoid enumerating or substantiating this pitfall with examples to avoid pin-pointing an arbitrary subset of papers. We suspect most readers can readily name such examples.

3.6. Pitfall 6: Emerging error-model-agnostic simulators

There are emerging simulators (like ZSim [22]) which model hardware at a very high level of detail and show validation to commercial products in the final performance estimation. Such simulators present both an opportunity and a danger to our community. For studying some phenomenon, like issues in large multi-core cache design, ZSim is an extremely fast simulator which models the appropriate components of a processor. However, it would not be useful for microprocessor core research, because it eschews its detailed modeling, and trumps as a feature that it can get the end effect correct while hiding and avoiding the details. The authors note on their release notes that "if you are using zsim with workloads or architectures that are significantly different from ours, you should not blindly trust these results" [1], we appreciate this *Architecture Surgeon General* warning, and would add strong caution against using these types of tools for microachitectural core exploration.

Suggestions: As mentioned before, we encourage tool authors to explicitly provide advice on what kinds of things the tool is not meant for.

3.7. Pitfall 7: Trends Myth

There is a myth that simulators have errors, but still get the trends right. We argue that this is flawed. Moreover, if trends are the important factor, this only strengthens the case for first-order models, instead of using simulators which can have large errors.

To substantiate, GPGPUSim's errors are both positive and negative, ranging from -30% to +50% when configured to match a Tesla C1060 as reported by Blem et al [6]. Another example is in a recent ISPASS paper [13], where a "validated" and detailed configuration of gem5 with source code modifications had errors of -40% to +12% on PARSEC and -30% to +45% on SPEC. This is for a total of 7 PARSEC measurements (six +ve, one -ve) and 11 SPEC measurements (three -ve, eight +ve). Without a dog in the fight, one would call such +ve and -ve measurements significant and be cautious about using the tool. However, our community insists such tools must be used. Furthermore, in the interest of claiming such tools are good, the ISPASS authors use a metric that ignores the sign, and reports a low average number. What's worse is that another metric is reported which averages these raw errors (thus canceling +ve and -ve), which is an even lower number. When the ISPASS authors claim that gem5 is "validated," it is on these metrics.

More importantly, when a tool has such divergent +ve and -ve error, what does it imply for the trust we can place in the improvement of a new technique? One argument is that though specific details of simulation are wrong, the overall "trends" will be correct. We postulate that this reasoning is often flawed in practice. That the trends will be correct relies on the assumption that the new technique is inherently insulated and unrelated to the source of these simulation errors. With such large +ve and -ve errors with unknown sources, we believe this position to be untenable in the general case.

One other response to the weaknesses of simulation tools has been that the evaluation does not matter, which brings us to our final pitfall.

3.8. Pitfall 8: Amplified reviewer noise

There is much refrain, especially among industry people and "senior" researchers, that they don't read the evaluation section and go for the big idea. Yet, few reviews reflect this magnanimous attitude (as observed by us from many PC committee

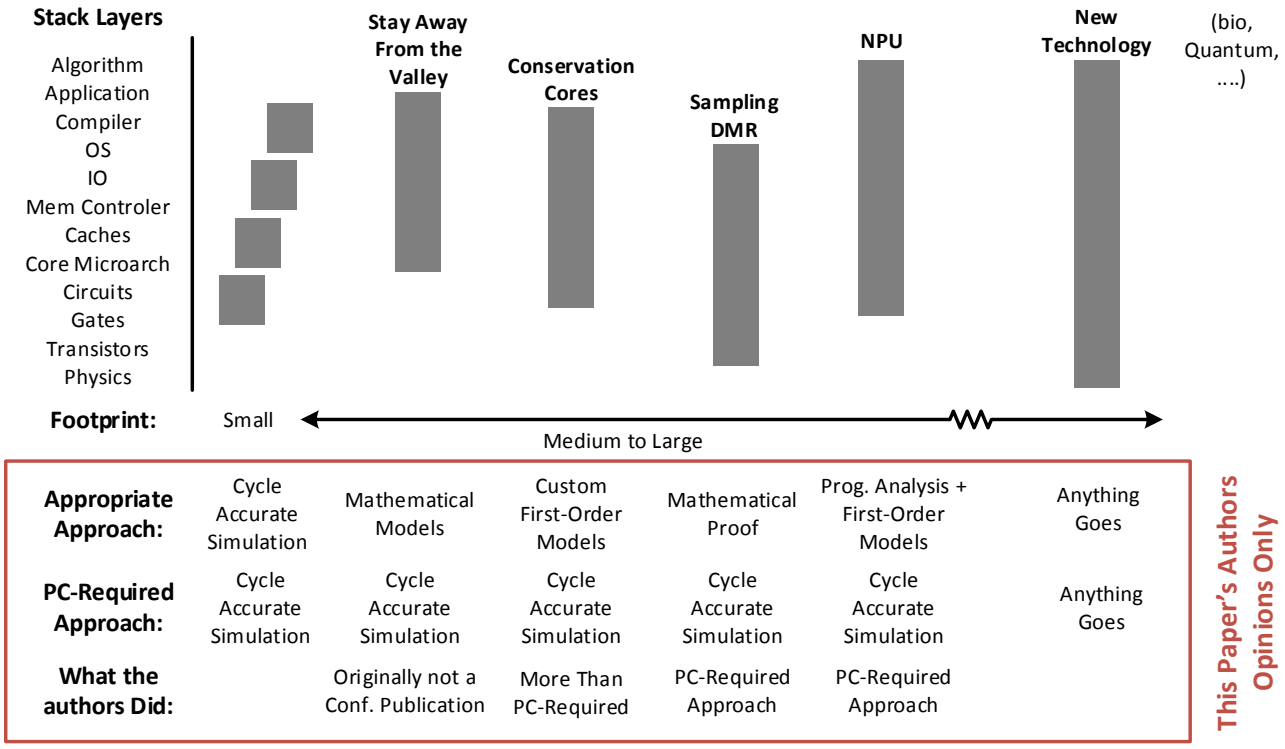

Figure 3: Footprints and approaches for various architectural research.
Gray boxes indicate, for several items of architectural research, the *footprint*, or which layers of the stack the proposed technique influences. For each technique, the figure describes our opinion of the appropriate approach, the PC-branded required approach, and how the authors coped with the discrepancy.

meetings). The bigger issue is an unequal standard and how reviewer assignment unnecessarily plays a role. Why should whether a paper ends up with one of these big-picture people play a significant role in the paper outcome?

Suggestions: Create a common standard on what is expected and is reasonable. In a small community such as ours this is not hard! We should encourage co-reviewers and program chairs to confront the reviewer that harps on quantitative detail early in the review process, not at the PC meeting - at which point scores have been given, and the decision has been made.

4. Suggestions for Escaping Simulate-or-Reject

We first outline a concept called the "footprint," which we believe can help our community calibrate evaluation expectations and avoid the templatized research standards we have adopted. We then comment briefly on open versus in-house tools, in terms of their challenges and the value they provide. Finally, we give suggestions for authors, reviewers and tool developers. Our goal is primarily to provoke thinking and not to provide solutions or strategies for every research direction.

4.1. Research Footprint

The design spectrum of research ideas can be considered in terms of "footprint," which we give a depiction of in Figure 3. An idea's footprint is simply the span of the stack layers which it affects or is influenced by. For any idea, the footprint can span core microarchitecture, circuits, gates, transistors, and physics going down. Going up, the footprint can span caches, the memory controller, IO, OS, compiler, application, and the algorithm.

The first row of Figure 3 describes what we believe would have been the best approach for evaluation. On the left, when the footprint of an idea 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. Some examples (not shown in figure) include the Decoupled Compressed Cache [23], and Sampled Temporal Memory Streaming [25].

Moving to 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, our field does not have this appreciation of an approach's footprint. Instead, in all but the most speculative research based on fundamentally new technology, the reviewer bar for quantitative research is "cycle-accurate" simulation (shown in the second row of the figure).

The final row of Figure 3 shows how each of four research examples dealt with the difference between the most appropriate and templatized PC-required approach. First, the authors of the "Stay away from the Valley" paper [15] were not able

to publish the non-simulator-enhanced version in a conference venue, and we believe this was at least partly because they used mathematical models rather than standard tools and simulation. For Conservation Cores [24], 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 some CACTI modeling) to obtain performance and energy. Their methodology was outside the mainstream and in our opinion exceeds the credibility of cycle-accurate simulation. However, the authors were subjected to reviewer-nitpicking on evaluation details, which we diagnose as mis-calibrated expectations. For the sampling DMR work [20], where simple math showed the technique had almost negligible overhead, authors were compelled by reviewers to spend weeks (3 grad-student summer months) on unimportant simulator implementation and evaluation. Ultimately the simulation "proved" the simple math. Finally, where program analysis and first-order modeling would have worked well for NPU [11], authors used full x86 simulation combined with McPAT models and other high-level energy estimates.

Overall, authors are forced to spend significant time in conforming to templatized reviewer expectations. If reviewers can appreciate the footprint of an approach, and tailor their expectations to match, authors can be free to use the most appropriate approach for their respective problems.

4.2. Community versus In-house Tools

When simulation *is* the best option, researchers must make the decision between using open-source community tools, and using custom in-house tools. Each ostensibly provides its own benefits: researchers developing custom in-house tools will be aware of their implementations and assumptions, while those who rely on open-source tools can benefit from (and contribute to) an active community that continually makes improvements. The fact that we were able to find the errors that we did, is a testament to the value of open tools. With no one to shed light on the bugs of in-house tools, its fair to suspect that they have a similar or higher degree of error. Therefore, we believe that as long as users spend the time to learn about a simulator's assumptions and implementation, they are better off with something open than something closed.

4.3. Our Advice

For Authors As we mentioned earlier on, the most important thing that researchers must realize is that these tools are *not* meant to be used as black boxes. Researchers are often better off building their own first-order model whose details they are aware of, or they should at least internally verify their simulators and understand the details of how they work. Researchers must avoid or be extremely cautious when using these tools for configurations outside the validated point.

For Tool Developers The most important thing for tool developers is to avoid overselling and be explicit when running tutorials on the valid usages of the tool. It is far more important to encourage appropriate use of tools rather than achieve a high citation count.

For Reviewers We have three take-aways for reviewers: i) Don't arbitrarily insist on cycle-level. ii) Don't arbitrarily insist on low-level power-modeling. iii) Before asking for details on a standard tool, please become familiar with what that tool is actually doing.

5. Conclusion

In this work, we have attempted to question the taken-for-granted quantitative evaluation strategy. We have observed many errors in our widely accepted tools and articulated 8 pitfalls that are not commonly acknowledged. Based on these, we encourage authors and reviewers to be welcoming to first-order models and *less* detailed tools where appropriate, and not to blindly insist on standard-tool-based evaluation. Overall, we believe this phase of detailed simulator misuse, which although detrimental, can eventually be overcome and make us stronger than before.

6. Acknowledgments

We would like to thank the anonymous reviewers, our Shepard Gabriel Loh, along with Mark Hill, Nilanjan Goswami, Steve Reinhardt, 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, comments on experiences, criticisms, writing advice, and words of warning. A special thanks to Vamsi Krishna Kodati, Paul Gratz, Yingying Tian, Zhe Wang, Daniel Jimenez, Ping Xiang, Huiyang Zhou, Nilanjan Goswami and Tao Li, who helped verify many of the errors we found. *Please note that the views and statements in the paper are of the authors alone; none of those mentioned in this acknowledgment either implicitly agree or disagree with these views.*

References

[1] "Zsim git homepage." May 2014. [Online]. Available: https://github.com/s5z/zsim

[2] T. M. Aamodt, W. W. L. Fung, and A. Boktor, "Gpgpu-sim 3.x: A performance simulator for many-core accelerator research," iSCA 2012 Tutorial. [Online]. Available: http://www.gpgpu-sim.org/isca2012-tutorial/GPGPU-Sim-Tutorial-ISCA2012.pdf

[3] A. Bakhoda, G. Yuan, W. Fung, H. Wong, and T. Aamodt, "Analyzing cuda workloads using a detailed gpu simulator," 2009.

[4] J. Balfour, W. Dally, D. Black-Schaffer, V. Parikh, and J. Park, "An energy-efficient processor architecture for embedded systems," *IEEE Computer Architecture Letters*, vol. 7, no. 1, pp. 29–32, 2008.

[5] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2024716.2024718

[6] E. Blem, M. Sinclair, and K. Sankaralingam, "Challenge benchmarks that must be conquered to sustain the gpu revolution," in *Proceedings of the 4th Workshop on Emerging Applications for Manycore Architecture*, 2011.

[7] D. Burger, "Future architectures will incorporate hpus, micro 2011 keynote."

[8] D. Burger and T. M. Austin, "The simplescalar tool set, version 2.0," *SIGARCH Comput. Archit. News*, vol. 25, no. 3, pp. 13–25, Jun. 1997. [Online]. Available: http://doi.acm.org/10.1145/268806.268810

[9] W. Dally, J. Balfour, D. Black-Shaffer, J. Chen, R. Harting, V. Parikh, J. Park, and D. Sheffield, "Efficient embedded computing," *Computer*, vol. 41, no. 7, pp. 27–32, July 2008.

[10] E. W. Dijkstra, "Letters to the editor: Go to statement considered harmful," *Commun. ACM*, vol. 11, no. 3, pp. 147–148, Mar. 1968. [Online]. Available: http://doi.acm.org/10.1145/362929.362947

[11] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 449–460. [Online]. Available: http://dx.doi.org/10.1109/MICRO.2012.48

[12] V. Govindaraju, C.-H. Ho, and K. Sankaralingam, "Dynamically specialized datapaths for energy efficient computing," in *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, 2011, pp. 503–514.

[13] A. Gutierrez, J. Pusdesris, R. G. Dreslinski, T. Mudge, C. Sudanthi, C. D. Emmons, M. Hayenga, and N. Paver, "Sources of error in full-system simulation," in *Proceedings ISPASS 14*.

[14] Z. Guz, O. Itzhak, I. Keidar, A. Kolodny, A. Mendelson, and U. Weiser, "Threads vs. caches: Modeling the behavior of parallel workloads," in *Computer Design (ICCD), 2010 IEEE International Conference on*, Oct 2010, pp. 274–281.

[15] Z. Guz, E. Bolotin, I. Keidar, A. Kolodny, A. Mendelson, and U. C. Weiser, "Many-core vs. many-thread machines: Stay away from the valley," *IEEE Computer Architecture Letters*, vol. 8, pp. 25–28, 2009.

[16] A. Jog, O. Kayiran, N. Chidambaram Nachiappan, A. K. Mishra, M. T. Kandemir, O. Mutlu, R. Iyer, and C. R. Das, "Ojog:2013:oct:2451116.245115owl: Cooperative thread array aware scheduling techniques for improving gpgpu performance," in *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '13. New York, NY, USA: ACM, 2013, pp. 395–406. [Online]. Available: http://doi.acm.org/10.1145/2451116.2451158

[17] A. Jog, O. Kayiran, A. K. Mishra, M. T. Kandemir, O. Mutlu, R. Iyer, and C. R. Das, "Orchestrated scheduling and prefetching for gpgpus," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ser. ISCA '13. New York, NY, USA: ACM, 2013, pp. 332–343. [Online]. Available: http://doi.acm.org/10.1145/2485922.2485951

[18] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. S. Kim, T. M. Aamodt, and V. J. Reddi, "Gpuwattch: Enabling energy optimizations in gpgpus," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ser. ISCA '13. New York, NY, USA: ACM, 2013, pp. 487–498. [Online]. Available: http://doi.acm.org/10.1145/2485922.2485964

[19] S. Li, J. Ahn, J. B. Brockman, and N. P. Jouppi, "Mcpat 1.0: An integrated power, area, and timing modeling framework for multicore architectures," *HP Labs*, 2009.

[20] S. Nomura, M. D. Sinclair, C.-H. Ho, V. Govindaraju, M. de Kruijf, and K. Sankaralingam, "Sampling + dmr: Practical and low-overhead permanent fault detection," in *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ser. ISCA '11. New York, NY, USA: ACM, 2011, pp. 201–212. [Online]. Available: http://doi.acm.org/10.1145/2000064.2000089

[21] NSF, "Exploiting parallelism and scalability (xps)."

[22] D. Sanchez and C. Kozyrakis, "Zsim: Fast and accurate microarchitectural simulation of thousand-core systems," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ser. ISCA '13. New York, NY, USA: ACM, 2013, pp. 475–486. [Online]. Available: http://doi.acm.org/10.1145/2485922.2485963

[23] S. Sardashti and D. A. Wood, "Decoupled compressed cache: Exploiting spatial locality for energy-optimized compressed caching," in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-46. New York, NY, USA: ACM, 2013, pp. 62–73. [Online]. Available: http://doi.acm.org/10.1145/2540708.2540715

[24] G. Venkatesh, J. Sampson, N. Goulding, S. Garcia, V. Bryksin, J. Lugo-Martinez, S. Swanson, and M. B. Taylor, "Conservation Cores: Reducing the Energy of Mature Computations," in *ASPLOS '10*.

[25] T. Wenisch, M. Ferdman, A. Ailamaki, B. Falsafi, and A. Moshovos, "Practical off-chip meta-data for temporal memory streaming," in *HPCA 2009*.