A Roadmap and Plan of Action for Community-Supported Empirical Evaluation in Computer Architecture*

Bruce R. Childers
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA USA
childers@cs.pitt.edu

Alex K. Jones
Department of Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, PA USA
akjones@pitt.edu

Daniel Mossé
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA USA
mosse@cs.pitt.edu

ABSTRACT

A framework of open interoperable simulators for computer architecture is long overdue. Today there are many separate, uncoordinated efforts to develop simulation and modeling artifacts (tools) for computer architecture research. The artifacts are used to empirically evaluate new computer architecture innovations and compare them with the state of the art. The artifacts are usually developed by individual groups, often for a specific purpose, and may not be publicly released. Consequently, it is difficult to leverage investment in artifact development and to repeat or reproduce experiments. In this position paper, we present recommendations and a roadmap for sharing and building open-source, interoperable simulation and modeling artifacts. The recommendations are the outcome of a community workshop involving industry, government and academia to determine how to coordinate effort, share tools and improve methodology.

Keywords

Repository, Repeatability, Computer Architecture

1. INTRODUCTION

Software simulation, hardware emulation, analytic modeling and benchmarking are pivotal to the development and evaluation of all types of computer architectures and systems from simple microcontrollers in deeply embedded applications to sophisticated and parallel general-purpose and graphics processing units in exascale systems. These techniques are used to analyze existing systems and their bottlenecks, develop new hardware capabilities, and study and evaluate design trade-offs, leading to faster, lower energy and higher reliability computers. Many research groups have made a large investment in computer architecture *artifacts*, i.e., evaluation tools and workloads. A primary purpose of these artifacts is to enable *experiments*, or observational studies of proposed architectural capabilities.

While artifacts are fundamental to empirical evaluation of computer architectures, the collection of artifacts suffers from a few important problems. They are often fragmented, ad hoc, and internal (i.e., not released publicly) efforts. Further investment (e.g., to produce an artifact suitable for external release) is perceived contrary to expediency of the particular research effort, where the focus is on the end research results, rather than the artifacts and experiments themselves. There is often little to no incentive to build, release and maintain soundly developed and robust artifacts for the broader community. As a result, the artifacts that are developed are many times brittle, difficult to reuse and extend for new purposes, scalable to only a few design and benchmark sizes, insufficiently documented, and minimally tested and validated. It is onerous to repeat or reproduce results for comparison or trust the conclusions drawn with these artifacts, which impedes the quality and pace of computer architecture innovation.

Given this state of affairs, researchers may use an existing artifact without a good understanding of how choices made in the artifact interact with their new models. Alternatively, they may resort to "rolling their own" to have control and detailed understanding of the artifact. The proliferation of artifacts for CPU/cache (e.g., Simics, SimpleScalar, M5, PTLsim, Opal, QEMU, gem5, Sniper), main memory (e.g., DRAMsim2, MemSim, Ruby), storage (e.g., DiskSim, FlashSim, DiskSim-SSD, FRP) and network-on-a-chip (e.g., GARNET, Noxim, DARSIM, Hornet) simulation and emulation (e.g., RAMP, FAST, ProtoFlex), reflects the state of the tools, where a significant investment has been made that cannot be readily recouped. These exemplars represent only the ones released to the community and do not include the copious effort in internally developed (i.e., point solution) tools and benchmarks.

The challenge *is recognized* by the architecture community and efforts are underway to begin to address it. In this paper, we report on the outcomes of a workshop, "Community Supported Computer Architecture Design and Evaluation Framework" (CSA), that brought together artifact developers and users to assess the current situation and to establish strategies and build a community for coordinated development and experimentation. The workshop fostered the formation and growth of a community of users (i.e., researchers that use the artifacts) and implementers dedicated to building open-source, extensible, scalable, tested, and interoperable infrastructure for processor, cache, network-on-a-chip (NoC), main memory, and storage. The CSA workshop engaged developers and users from academia, government and industry.

*The workshop and associated efforts described in this report were supported by the National Science Foundation through grants CCF-1148646 and CNS-1305220.

In this position paper, we report on the CSA workshop activities and the outcomes. Through a series of talks, intensive whole and small group discussion, the participants developed a set of directions, formulated as a roadmap, and tangible recommendations for working together as a community. The roadmap and recommendations were intended as guidance to the community on how to leverage investment, improve artifact quality, and share results.

2. CONVENING THE COMMUNITY

The two-day CSA workshop was sponsored by the National Science Foundation. Participation was by invitation. The thirty-six participants represented artifact users and developers from academia, government and industry. Participation from representatives of these different groups was important to ensure a broad perspective, taking into account relevant concerns of the groups (e.g., simulators for academic research versus industrial chip development). An archive of participant presentations and discussion transcripts are available from the CSA web site, `http://csa.cs.pitt.edu`.

The workshop had four tasks: 1) assess current architecture simulation, hardware emulation, analytic modeling and benchmarking infrastructure; 2) determine the needs and development priorities for the simulation framework, including what capabilities should be added to future artifacts; 3) identify ways to leverage, combine and coordinate disparate investment in the simulation framework by multiple groups; and, 4) develop strategies and actions to build a community that supports a suite of well tested, validated and interoperable artifacts, and the experimental results collected with the artifacts. These tasks were done through break-out sessions and group discussion to (a) form a cohesive view of today's infrastructure for simulation and emulation; (b) develop the roadmap; and, (c) identify near-term actions.

3. THE CHALLENGE

CSA examined the state of artifact development to learn why the community has generally not worked together on interoperable artifacts and shared experiments. The assessment identified several issues, which we report below, that frame the challenge of building and maintaining open-source interoperable tools.

The assessment considered various different evaluation methodologies used in computer architecture, namely:

- *Analytic modeling* is used in early designs to ask early "what-if" questions when many choices need to be considered quickly. Analytic modeling is also useful to scale design evaluation to large systems and long-running applications (e.g., statistical sampling to reduce simulation time).

- *Software simulation* relies on a computer program to mimic hardware behavior. It is used in middle design stages to behaviorally model new architecture ideas for coarse-grain study and evaluation because the simulators can be developed reasonably quickly, but a tradeoff between accuracy and speed is typically required.

- *Hardware emulation* implements actual circuits in a programmable device, such as a field-programmable gate array (FPGA), to prototype design choices. It provides increased accuracy and speed but with the need for additional development investment. Hardware emulation is often done later than software simulation, once design choices are narrowed.

- *Hybrid simulation–emulation* achieves the benefits of simulation (earlier design through behavioral modeling) and emulation (fast and detailed modeling), which enables the consideration of more architecture choices and better decisions. Hybrid software simulation and hardware emulation techniques are becoming more useful as computer architectures, benchmarks, and input data sets grow larger.

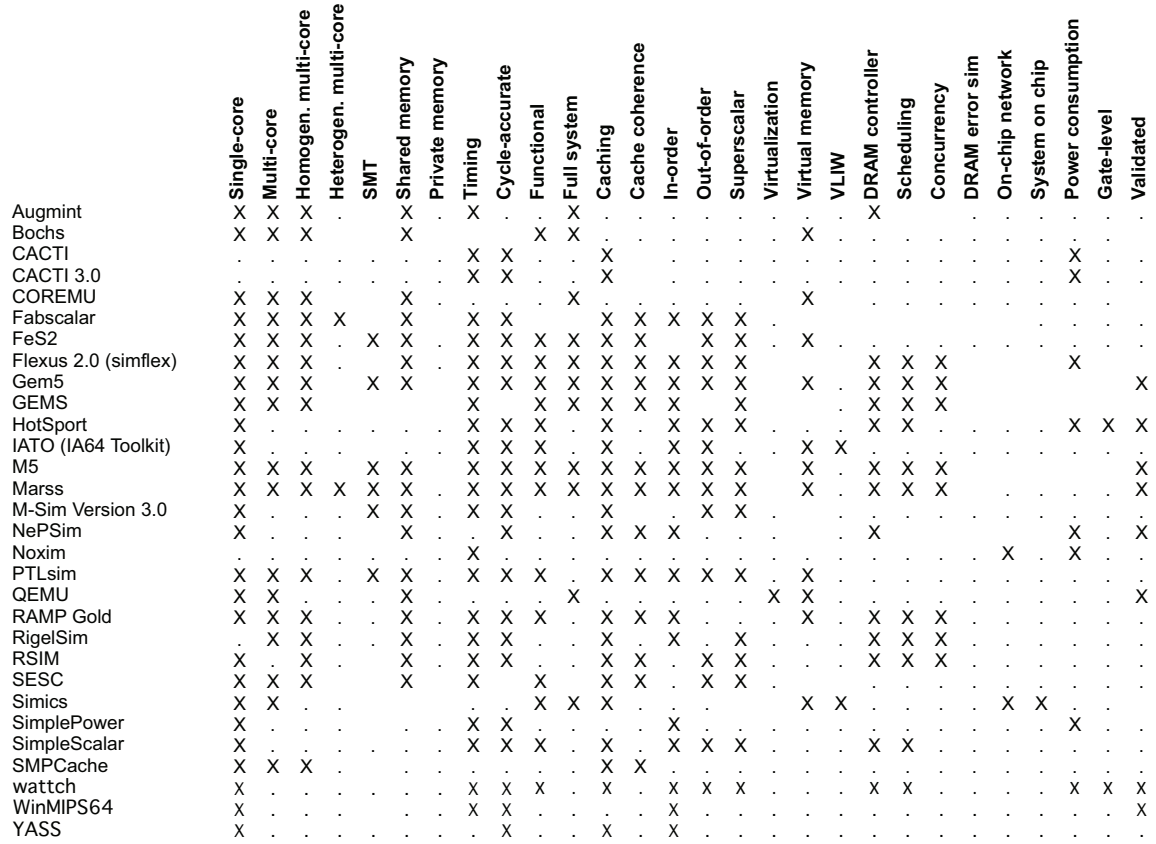
For each assessment method, an additional dimension to the experimental space is the evaluation workload. This requires benchmarking artifacts. *Benchmarking* evaluates design choices under a range of software programs and data sets to determine relative benefit and cost. Benchmarks are the stimulus that drives analytic modeling, software simulation and hardware emulation.

- Benchmarks may be actual software applications, simplified variants exhibiting similar behavior as full applications ("mini-apps"), execution traces of a particular application behaviors (e.g., MPI calls, memory addresses), or analytic models (e.g., state machines).

Several issues exist when creating and using such tools for evaluation of computer architecture capabilities. Below we discuss the issues of (a) the breadth of tools, (b) fragmentation of development, (c) longevity of the tool, (d) interoperability, (e) performance and scalability, and (f) perceived value/reward by the community.

The first issue is the breadth of characteristics exhibited by each artifact category that inherently divides tool development. For example, the concerns of benchmark developers are different than the concerns of simulator developers. Benchmarks tend to be relatively stable, which avoids the "moving target syndrome" faced by developers of an actively used simulator. Even in a single category, there are competing needs driven by separate subcommunities. For example, high-performance computing is concerned with modeling systems "at scale", often for a small set of specific applications. This focus strongly influences how evaluation is approached (i.e., abstraction and fidelity) and the underlying mechanisms that are used (e.g., parallel discrete event simulation). For general-purpose evaluation, there may be less emphasis placed on scale, but more emphasis placed on low-level detail (e.g., interaction of performance and thermals). Furthermore, in each category, the tools differ in several dimensions: levels of integration; scalability and coverage; performance; methods and techniques supported; fidelity and validation; flexibility and modularity; metrics gathered and reported; and, functional vs. real data-driven workloads.

The second issue is the size of the community and fragmentation of the development teams working towards individual goals. Across the spectrum of categories, the amount of investment is simply huge. Indeed,



	Single-core	Multi-core	Homogen. multi-core	Heterogen. multi-core	SMT	Shared memory	Private memory	Timing	Cycle-accurate	Functional	Full system	Caching	Cache coherence	In-order	Out-of-order	Superscalar	Virtualization	Virtual memory	VLIW	DRAM controller	Scheduling	Concurrency	DRAM error sim	On-chip network	System on chip	Power consumption	Gate-level	Validated
Augmint	X	X	X	.	.	X	.	X	.	.	X	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.
Bochs	X	X	X			X				X	X	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	
CACTI	.	.	.	.	.	.	.	X	X	.	.	X		.	.	.	.	.	.	.	.	.	.	.	.	X	.	.
CACTI 3.0	.	.	.	.	.	.	.	X	X	.	.	X		.	.	.	.	.	.	.	.	.	.	.	.	X	.	.
COREMU	X	X	X			X	.	.	.	.	X	.	.	.	.	.		X		.	.	.	.	.	.	.	.	
Fabscalar	X	X	X	X		X		X	X			X	X	X	X	X	.								.	.	.	.
FeS2	X	X	X	.	X	X	.	X	X	X	X	X	X		X	X	.	X	.	.	.	.	.	.	.	.	.	.
Flexus 2.0 (simflex)	X	X	X	.		X	.	X	X	X	X	X	X	X	X	X				X	X	X				X		
Gem5	X	X	X		X	X		X	X	X	X	X	X	X	X	X		X	.	X	X	X						X
GEMS	X	X	X					X		X	X	X	X	X		X			.	X	X	X						
HotSport	X	.	.	.	.	.	.	X	X	X	.	X	.	X	X	X	.	.	.	X	X	.	.	.	.	X	X	X
IATO (IA64 Toolkit)	X	.	.	.		.	.	X	X	X	.	X	.	X	X	.	.	X	X	.	.	.	.	.	.	.	.	.
M5	X	X	X		X	X		X	X	X	X	X	X	X	X	X		X	.	X	X	X						X
Marss	X	X	X	X	X	X	.	X	X	X	X	X	X	X	X	X		X	.	X	X	X		.	.	.	.	X
M-Sim Version 3.0	X	.	.	.	X	X	.	X	X	.	.	X		.	X	X	.		.	.	.	.	.	.	.	.	.	.
NePSim	X	.	.	.		X	.	.	X	.	.	X	X	X	.	.	.		.	X						X	.	X
Noxim	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	X	.	X	.	.
PTLsim	X	X	X	.	X	X	.	X	X	X	.	X	X	X	X	X	.	X		.	.	.	.	.	.	.	.	.
QEMU	X	X	.	.	.	X	.	.	.	.	X	.	.	.	.	.	X	X	.	.	.	.	.	.	.	.	.	X
RAMP Gold	X	X	X	.	.	X	.	X	X	X	.	X	X	X	.	.	.	X	.	X	X	X	.	.	.	.	.	.
RigelSim	.	X	X	.	.	X	.	X	X	.	.	X	.	X	.	X	.	.	.	X	X	X	.	.	.	.	.	.
RSIM	X	.	X	.		X	.	X	X	.	.	X	X	.	X	X	.	.	.	X	X	X	.	.	.	.	.	.
SESC	X	X	X			X		X		X		X	X	.	X	X	.			.	.	.	.	.	.	.	.	.
Simics	X	X	.	.				.	.	X	X	X	.	.	.			X	X	.	.	.	.	X	X	.	.	
SimplePower	X	.	.	.		.	.	X	X	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	X	.	.
SimpleScalar	X	.	.	.	.	.	.	X	X	X	.	X	.	X	X	X	.	.	.	X	X	.	.	.	.	.	.	.
SMPCache	X	X	X	.		.	.	.	.	.	.	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
wattch	X	.	.	.	.	.	.	X	X	X	.	X	.	X	X	X	.	.	.	X	X	.	.	.	.	X	X	X
WinMIPS64	X	.	.	.		.	.	X	X	.	.	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	X
YASS	X	.	.	.	.	.	.	.	X	.	.	X	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Figure 1: A selection of simulators and their features (rows). We use a 'X' to mean that a simulator has the feature, a '.' that it does not have the feature, and a blank when it is unclear.

research conferences for computer architecture collectively attract thousands of attendees each year who have all certainly developed, extended, or used artifacts and conducted experiments in their own work. The community has been astonishingly prolific in artifact production. Figure 1 shows a *small* selection of software simulation-based artifacts and their features. This snapshot demonstrates the diversity in even a small collection of tools—there is overlap in some capabilities by multiple tools, but not in others.

The differences and duplication in features is symptomatic of fragmented, ad hoc and internal (i.e., not released publicly) nature of existing effort and investment. With isolated development spread across many groups, there is no single "meeting place", or *exchange*, where artifact developers and users can gather to learn about the tools and coordinate effort. There is no general catalog of the artifacts, their capabilities, development status, and where to find them. Consequently, it is difficult to locate artifacts and to know whether what is found is appropriate for a particular need. The information in Figure 1 serves as an example of how a catalog might be developed for the community. This partial index is only a starting point; a useful and complete listing requires the community to identify and categorize the tools. The central exchange could provide a "home base" for developing and accessing such a catalog.

The third issue is the useful lifetime of the artifacts—historically, they have tended to exhibit either short lifetimes or very long ones. An artifact may prove useful to an individual research group, which then releases the tool but does not maintain or continue to promote it. This issue is typical of the "student graduation problem": A student may actively develop his/her artifact during thesis research and release it to others out of altruism. However, when the student graduates, the artifact quickly becomes an orphan — the former student has new priorities, his/her previous research group's focus shifts, etc. There is no vested interest to keep the artifact up-to-date by the original authors, and the artifact becomes the responsibility of the community to "own". More likely, the artifact becomes another one that is discarded in the long history of useful but forgotten tools.

At the other extreme, an artifact may have a much longer than anticipated lifetime; it needs to be robust and flexible to last its entire life, and needs to be maintained and documented sufficiently to hand it off from one set of developers to the next. Figure 2 illustrates this issue. The figure shows the lifetime of several publicly available artifacts. The x-axis shows a period of 15 years and the y-axis shows citation count[1]. The rise and fall in citation count for an artifact defines its lifetime. To determine the curves, we examined four conference proceedings (ISCA, HPCA, MICRO and ASPLOS) from 1997 to 2011. We counted how often a particular tool was mentioned as being used in the

[1]Citation count is but a proxy for real usage, popularity and the impact of the research created with the tool.

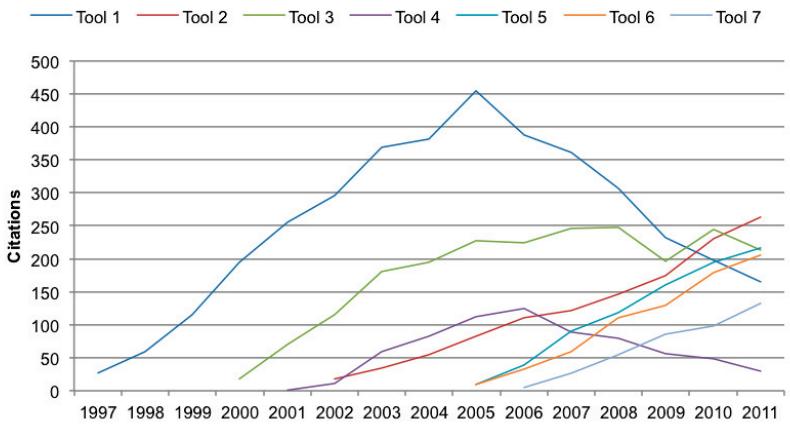


Figure 2: Lifetime of some popular evaluation artifacts.

experimental methodology of each paper. In the graph, the names of the artifacts are anonymized because we intend this figure to only illustrate the issue of lifetime rather than making a statement about specific artifact importance.

Tool 1 is a single core simulator, which can model aggressive superscalar designs. As measured by the rising and falling citation counts, it had a lifetime that spanned the full 15 shown in the chart. This example suggests that the community should be prepared to develop tools for longevity and to maintain them. For a tool with a long lifetime, multiple groups probably will need to support the tool; it is an unreasonable expectation, and possibly an undesirable one, that a single group will take on the responsibility to maintain a tool for such a long time period. As Dennard scaling reached its conclusion and power and thermal effects of frequency scaling became evident, Tool 1 was gradually replaced by Tool 2, which focused on multi-core simulation. The shift in technology brought about a tool change, and the community should anticipate emerging technologies and the implications to current and future artifact development and maintenance. A new artifact can appear and gain quick use by the community as illustrated by Tool 3. Similarly, a tool can be widely used, and then have a quick drop-off in usage. Tool 4 exhibits this behavior: A much improved version was made available several years after the tool's first introduction. In general, the development and maintenance of the artifacts will follow changes in research priorities, and investment in artifacts needs to shift with the change.

The fourth issue is the technical differences between tools that limits integration—whether they have compatible interfaces, or can be meaningfully integrated. The available artifacts have varying levels of integration, and cover a huge spectrum of capabilities, which compounds the difficulty of using existing artifacts and having them interoperate. Current artifacts address different scales as well, including datacenter- and cluster-scale, servers and PCs, and mobile devices. Most tools exist for the middle of the spectrum, i.e., systems with single to a few nodes. There are current simulators, such as gem5 [7], SST [43] and Manifold [32], that have flexibility and modularity, but their interfaces have not been standardized, which is necessary for long-term interoperability.

Additional interoperability challenges are to combine cycle-accurate, functional-driven, and trace-driven tools, to reconcile the performance vs. fidelity tradeoff (e.g., integrating cycle accurate component models into more abstract models for full systems), and to reconcile the many metrics—of varying types and accuracy—provided by existing tools, which must be composed, including traditional (cycles per instruction, performance, energy, reliability, etc) and non-traditional (i.e., application-dependent such as QoS).

A fifth issue is the performance and single-view limitations associated with existing tools. Simulator performance is problematic with increasing data set sizes and complexity of modern multi-core/heterogeneous architectures. Today's tools are not, largely, parallel simulations. This is a critical limitation, as available computers are largely parallel, yet the simulators that model them are not. Some counter examples are Sniper [10, 17] a parallel multi-core simulator and Hornet [40], a parallel network-on-chip simulator. Sniper is not a cycle accurate simulator and uses abstract modeling for key functions of modern processors (e.g., out-of-order execution, cache access times, network-on-chip latencies). Hornet is a cycle accurate network simulator. However, both simulators exhibit scalability concerns making them only able to leverage four or eight host cores effectively.

As a result, the methods and techniques that are most commonly used to improve performance are statistical sampling and functional emulation combined with detailed ("cycle accurate") sequential discrete event simulation. Current artifacts rarely allow trade-offs between fidelity and validation due to performance and integration limitations. That is, the artifacts model systems only at a single level, and usually do not allow 'zooming' in and out on different aspects to get a broader or narrower understanding of design trade-offs being evaluated. This capability is desirable to improve the speed of design explorations (e.g., abstract modeling of less important design aspects to gain enough time to model in detail the choices where accuracy is critical). This also creates interface difficulties between tools with different fidelities.

A final issue is lack of value perceived by the community in building and distributing artifacts. There

is tremendous pressure to produce new results at an increasingly rapid pace. Consequently, research expediency to innovate takes a much higher priority than developing artifacts. The value put on dissemination of results (the "ends") far outweighs the value put on producing results with robust, carefully validated and shared artifacts (the "means"). The scientific methodology of producing results has also suffered; there is minimal accountability in computer architecture research, especially in academic environments, because the artifacts and metadata needed for repeatability and reproducibility are rarely disseminated since there is minimal perceived value in doing so.

With the need to produce results, an "island model" of research dominates, where individual researchers develop artifacts and extensions to existing artifacts (possibly to an open source artifact) strictly within their own group. The artifacts are not typically released beyond the group; given the time and effort it takes to develop such artifacts, this is to be expected. Importantly, the island model also preserves competitive advantage: The capability offered by a tool developed in-house, or perhaps a combination of tools put together in some unique way, is a powerful means for an academic or industry research group to "carve out" a space for their own research. This "island" is then difficult difficult for other researchers to enter without making a similar investment in tools (i.e., "setup your own island"). The island model is perceived to have more value to individual research groups than a community collaborative model.

The balance of value between the island and community models must shift to enable open-source interoperable artifacts. Services, capabilities, recognition and access to needed capabilities can be used to create more value from the community approach. The value needs to be high enough to achieve critical mass in contribution and to sustain the effort. This may require an incentive model or reward structure for users to share their artifacts and experiments with the larger community. Some compromise may also be necessary, such as limiting access to tools and the research innovations implemented with them until some time period has elapsed after the research has been described. Such "time access rights" can help mitigate the conflict between competitive advantage and openness for credible scientific methodology. For industrial research, especially centering around products, it may be unrealistic to expect that actual simulators embodying commercial designs will be provided. However, research innovations, including ones made by industry research labs, that point the way to new directions—not actual designs developed and evaluated in industry—can and should be released as part of sound science.

4. ROADMAP FOR INVESTMENT

From the assessment, the workshop participants proposed several approaches to begin to address the challenges. To lay the groundwork for a well maintained, robust, interoperable infrastructure, requires several aspects to be addressed. The approaches are formulated as a "roadmap".

First, a needs document describing the range of what the community requires should be developed. This document would capture how artifact infrastructures would be used. It could also indicate the simulation, emulation, and benchmarking capabilities that are necessary to provide for community requirements. It would also identify emerging technologies and approaches that will need to be modeled in the future. This will help ensure the artifacts proactively anticipate future needs. This will help researchers avoid having to implement their own simulators because current ones are limited or unsuitable for their needs. It would also help direct investment by the community on existing and newly proposed artifacts.

Second, a model of governance for consensus and decision-making is necessary. This will help ensure that the entire community can participate. It will also keep the community "on the same page". A governance structure should be formed to establish community-recommended practices and methodologies for artifacts. The issues to be addressed by this governance body are enumerated in some detail in Table 1. Policies and procedures for these issues must ensure the artifacts are beneficial to a wide selection of community members to attract participation. The policies and procedures developed by this governance body must also be developed with due consideration of the burden placed on the administration for such policies.

Third, a central repository is needed. This repository could be managed under the model of governance. The repository will need full-time staff support and should not solely be an effort of graduate students. It will also need some way to validate and test the objects (simulators, emulators, experiments) contributed to the repository, including ways to curate, index and search the objects. This repository can also serve as the gateway to an experimentation instrument. Researchers may then use the instrument to run artifacts and create new experiments, perhaps taking advantage of compute resources not locally available from multiple server instances to specialized hardware (e.g., GPUs and FPGAs. As a result, experiments may be evaluated with additional workloads and/or become the baseline for direct comparisons with newly proposed experiments without necessarily revealing underlying details. Artifact modifications can be available to other users and developers with various levels of validation from certified to "use at your own risk". Thus, an important component of managing the repository will be ensuring these capabilities to be compatible with policies for use as established by the community governance.

Fourth, the "glue logic" between simulator and emulator components and models needs to be defined and developed. This glue logic would define simulation layers and their interfaces. It should be simple and must support existing artifacts to make it easier to boot-strap interoperability. It should be intuitive and uniform, supporting many different design choices and people that will use it. It should not hurt simulation performance or the possibility of parallelization. It will require varying abstractions handled by the artifacts. It will be seamless at different levels of abstraction, including parallelization, virtualized execution, and execution on accelerators such as GPUs and FPGAs. It will define the data formats for communication across interfaces, including semantics, statistics, report generation, automated analysis (particularly anomalies), and visualization. The interfaces must be adaptable and extensible for future change, as nothing in computer architecture is static.

Resource Management	Determine policies for fair allocation of local and distributed development, including access to resources; credit and recognition strategies for contributing resources, development, and/or obtaining higher reward seals for creating quality artifacts; financial contribution toward maintenance and sustainability of development.
Methodology	Define best practices of development, testing, validation; surveying users for their feedback (similar to EBay or Amazon) on artifacts; reward types, their meaning and the criteria for reward of noteworthy artifacts.
Education	Identify needs for training materials of best practices for software engineering practices relevant to artifact development by computer architecture researchers; encourage community to contribute materials (e.g., presentation slides, recording of presentations, etc.); organize and distribute the materials (e.g., indexing presentations from FAQs)
Assessment	Define statistics (#users, #citations, etc.) for assessing impact of artifacts and determining development priorities and needs; design user satisfaction survey; analyzes statistics; define and refine strategy for future development.
Ontology	Define common terminology, classifications and characterizations of artifacts and expressing the needs of community in a common language.
Licensing	Evaluate license choices, characterize licenses and the impact of selecting a particular license for a tool developer, recommend licensing choices to allow open development by academia, industry and government; address issues surrounding properties artifacts (e.g., tools from industry) and competitiveness.

Table 1: Governance issues for community-supported computer architecture artifacts

Finally, the interoperable artifacts must build on a foundation that is acceptable to the broader community. The foundation must migrate and wrap existing artifacts in the APIs and interfaces to achieve interoperability. An overarching infrastructure should be developed to implement basic components and runtimes, provide functional and metric validation. It must address parallelization directly to move beyond current performance limitations and restrictions in the scales of the systems that can be modeled. Development guidelines may be necessary in the foundation to govern modules at different fidelities and accuracy to allow better abstraction and valid composition.

5. PLAN OF ACTION

The roadmap identifies the general directions toward improving computer architecture artifacts. At these meetings, a "plan of action" was also developed to determine immediate and tangible steps that can and should be taken by the community to begin achieving the directions stated in the roadmap. Short and long term steps were both recommended, which we outline below.

First, small working groups should be developed to continue the CSA effort and to address technical details. The working groups should particularly address the governance issues (Table 1) as these affect the whole community and require broad support. The working groups will solicit and aggregate needs and requirements from the community. The groups will define standard layers, and eventually, the groups will develop definitions of standard simulation and emulation interfaces. Multiple discussion sessions among community participants, including those outside the architecture community (e.g., application developers), must be undertaken to ensure everyone is comfortable with the results of the working groups and ensure the groups are synchronized and speaking the same language. Participation is needed from academia, industry and government to ensure the issues of the different stakeholders are brought to the forefront and collectively addressed.

This step is in motion through Bird of a Feather sessions at Supercomputing in 2012, 2013 and 2014 (SC12, SC13, SC14). A meeting, attended by forty-five participants, was held to plan the working groups at the International Symposium on Microarchitecture (MICRO-45), Vancouver, Canada, 2012. The REPRODUCE workshop, held during the Conference on High-Performance Computer Architecture (HPCA), Orlando, FL, 2014, served as an interdisciplinary forum for researchers, practitioners and developers in computer engineering to discuss ideas, experience, trusted and reproducible research methodologies, practical techniques, tools and repositories. A similar workshop, TRUST, to bridge between architecture and the compiler communities was held at the Conference on Programming Language Design and Implementation (PLDI), Edinburgh, Scotland, 2014.

Second, a central repository should be established. It could be built iteratively, with existing tools like gem5 and SST, before interfaces are defined. An education and training portal could be started to include tutorials, videos, slideshows and webinars on best practices for constructing and using the simulation and emulation tools. Community backing should be sought for the repository and grown gradually to ensure traction and long-term sustainability. The ultimate goal is to have the repository become integral to computer architecture research and development.

The Open Curation for Computer Architecture Modeling (OCCAM) project is undertaking the infrastructure, community and education development needed for the central repository. OCCAM is a community-supported instrument for open-access to tools and experiments for computer architecture research (`http://www.occamportal.org`). The project is a pilot to learn community requirements and concerns in sharing artifacts and experimental results. It builds and engages an active community of users, establishes governance and access policies, and determines the requirements for software services of the repository that will create value to encourage researchers to leave their island model and participate in more open ways. OCCAM aims to show that individual effort can be saved by sharing simulators and experiments to avoid the burden of re-implementing and re-creating tools, experiments, data sets, benchmarks, etc. These savings can then be directed on innovating new architectural techniques for better computer systems. Unique

capabilities can also be created from the tools and experiments provided in the exchange from the whole community, which cannot be done through an island model. The OCCAM pilot gives an opportunity for community members to comment and give suggestions to evolve and refine the approach, and find the most compelling ways to change the value balance from an island model to one with more openness.

Third, the community needs to be educated on the benefits of developing and sharing artifacts. To be successful, any effort to develop shared infrastructure in computer architecture will require a concentrated and sustained education and training component. The barriers to adoption, contribution and use of the infrastructure should be addressed with training on (1) best practices for simulation and emulation techniques, software development practices, quality documentation, testing practices, and validation against real hardware, when possible; (2) ways shared effort can actually *increase* research productivity by leveraging investment, rather than impede productivity due to perceived hurdles or burdens on development practices; (3) needs and benefits of scientifically sound and accountable experimental methodologies to provide open access to all metadata for repeatability and reproducibility, including access to the source for modeling, simulation, emulation and benchmark artifacts used in evaluation.

The education effort should address both senior (influential leaders) and junior (students) members of the community. The working group activities provide an opportunity to directly educate members of the community; but materials, such as online video tutorials, should also be created for wider community distribution.

Fourth, incentives need to be provided to attract and sustain participation. A critical key to improving the state of the artifacts is to create value behind contribution and use of the infrastructure. This value can be provided by producing quality, simulation and emulation artifacts. The artifacts must be thoroughly documented and validated. The artifacts should separate the simulation framework from the simulator components to ensure it can be used by different groups. The artifacts may involve multiple distributions built from the bottom-up. An approach similar to the one taken for Linux may be a good model, allowing multiple groups to participate. Through the central repository, individual parts of the infrastructure could be vetted and integrated. The community must also be incentivized to contribute, including experimental outcomes and designs evaluated with the infrastructure. The incentives can be altruistic ones, but other incentives will be necessary, such as visibility, responsibility, stamps of approval, access to models and data in return for contributing, and delaying the release of artifacts and experiments to allow leveraging one's own research.

Finally, the infrastructure requires implementers and manpower behind the community activities and the working groups. Government and industry funding for a center with engineers whose specific job is interface maintenance, organizing and curating the contributions, conducting testing and integration development, etc., is critical to the long term success of this effort. The infrastructure and center must also be supported by industry, as they use and contribute to the artifacts. Industry may be convinced to dedicate some engineering effort (again, similarly to Linux), and contribute their research results and simulators used in their publicly disseminated conference and journal articles to the repository. Graduate students will be inevitably involved, but the artifacts will demand professional engineering and support staff to achieve the goal. The center should be organized and supported from its outset with an eye towards its sustainability.

6. AVAILABLE ARTIFACTS

There have been many related efforts for open-access repositories, open-access testbeds and open source artifacts that can help inform and guide the computer architecture community as it embarks on similar endeavors. Below, we list some examples as points of reference for the community, as well as to begin developing a catalog of what is available.

Open-access Repositories: nanoHUB is an effort centered around a community of researchers and educators for nanotechnology [34]. This community has come together to construct (and financially support) an online resource (`nanoHUB.org`) and a software framework (`hubzero.org`) to host communities, including online simulation tools. nanoHUB has been wildly successful for nanotechnology because it created capabilities that could not be done by any one individual group, such as tool frameworks, data sets, GUIs for building tools, experiment management, and coordinated education and infrastructure development. nanoHUB's success is evidence that providing a compelling service, education and community building can indeed lead to betterment in the way science is conducted. This report suggests a way for computer architecture researchers to reproduce this success for computer architecture evaluation. Another successful repository is `arXiv.org`, which is an open collection of scientific articles, including computer science and engineering. The CMU Artificial Intelligence repository is one of the oldest online open-access collections of research (started in 1993) [16]. There are also community-building efforts, such as HiPEAC [24] and ArtistDesign [3], that foster research specialties, but do not provide digital curation.

Open-access Testbeds: There are several open-access instruments for science and engineering. Open Cirrus supports cloud computing research across systems, applications, and services for heterogeneous datacenters [8, 35]. PlanetLab is a testbed for developing and deploying network services (e.g., file sharing, overlay networks, and object location) [13, 38, 39]. XSEDE [45] offers access to computers, services, data stores and education required for high-performance scientific simulation, visualization, and modeling. CloudLab and Chameleon are reconfigurable testbeds for cloud computing research and development open to the academic community [15, 11]. Amazon, Google and IBM offer scalable computing, including VMs, databases, and storage, on a pay-as-you-go basis.

There are also focused open-access testbeds that have been developed and maintained by a communities related to computer architecture. These testbeds are typically devoted to

specific needs and do not curate experiments. EmuLab is a framework for network research and education [23, 18]. It also has an education component to use the framework as virtual laboratory in coursework. Intel provided grant-driven access and training for their parallel Single Chip Cloud Computer through the MARC program [25], which demonstrates one way in which industry can offer access to proprietary capabilities. Seattle is a user-supported infrastructure for networking and distributed systems research and education [9, 42]. FAbRIC is a testbed [19] to create a shared reconfigurable substrate for hardware emulation. It will be used to share these expensive, unique resources with the community.

Simulation and Emulation Artifacts: Many open-source simulators have been developed for computer architecture. Due to space limitations, we focus on example artifacts from recent conference tutorials to give an indication of what is currently available. GPU Ocelot provides tracing for CUDA and PTX for GPGPUs [27, 28]. GPGPU-Sim is a GPU simulator for CUDA and OpenCL workloads [2, 1, 4]. MacSim is a simulator for heterogeneous architectures employing both GPGPUs and general-purpose Intel x86 CPUs [29, 30]. multi2sim is another heterogeneous CPU-GPU simulator [44]. MARSSx86 is a full-system Intel x86-64 simulator for multi-core architectures [22], with direct execution (QEMU [6]) and detailed models for coherent caches and on-chip networks. Sniper is a multi-core simulator [10, 17] built on interval core modeling [21] and the Graphite parallel simulation engine [33].

The Structural Simulation Toolkit (SST) is a modular framework from Sandia National Laboratories for "extreme-scale architectures" to "plug in" different simulators into a parallel simulation engine [43, 41, 26]. Manifold is a toolkit to construct scalable heterogeneous many-core simulators [32]. Another modular toolkit is gem5 [20, 7, 5], which supports multiple processor types, different instruction sets, multi-core architectures, and full-system simulation.

With the rapid advances of FPGA technology, increasingly more researchers are accelerating full-system simulation by splitting tasks within the simulation between FPGAs and collaborating software modules [12, 14, 36, 37]. UT-FAST [12], for example, partitions simulators into a speculative functional model (software) that simulates the instruction set architecture and a timing model (FPGA) that predicts performance. ProtoFlex [14] takes a different approach to partitioning and uses "transplanting" to dynamically reassign a simulated entity, such as a processor, from FPGA hardware to software simulation and vice versa. BlueSPARC models CPU pipelines in the FPGA and simulates I/O devices in Simics on a host PC. The BlueSPARC simulator, part of ProtoFlex, has performance comparable to or better than the Simics full-system simulator [31] in functional mode. Penry et al. [37] automatically generate a parallelized multicore processor simulator from a concurrent, structural model of the processor. Thanks to the structural modeling, their framework allows individual model components to be replaced with equivalent hardware. Finally, Pellauer et al. [36] reduced development effort and complexity for FPGA simulation by splitting a simulator into timing and functional partitions.

7. CONCLUSION

Computer architecture research continues to suffer from use of ad-hoc methodologies for innovation, sometimes limiting advancement. A symptom of this is the fragmented collection of evaluation artifacts that are used to evaluate new ideas in this field. By creating a community shift in how research is conducted, evaluated, and quantitatively shared—as has been previously demonstrated in other fields such as nanotechnology—can allow higher productivity in the computer architecture and related fields. In particular, this paper describes an effort to address this challenge through a community generated roadmap and action plan towards a more productive set of tools and methodologies in the computer architecture research space as determined through community-wide discussion at several workshops.

The actionable items are the following: The development of a community governance conducted through small topical working groups, possibly similar to standards committees, to define requirements based on community need and to articulate appropriate policies and interfaces. The construction of a central repository to host existing tools and to provide a backbone for the development of future artifacts as specified by the working groups. A concerted effort promoting the value for developing and sharing artifacts to the wider community. The development of an incentive ladder to promote adoption and enable sustainability of the approach. Identification of and support for manpower to spearhead the effort.

Although this effort is being done by and for the computer architecture community, it is clearly transferable to other communities that rely on simulations to evaluate their work, especially if there are many tools in play.

8. ACKNOWLEDGMENTS

The participants at the CSA workshop all contributed to the directions reported in this article. A list of participants appears in the Appendix. In addition, we thank Noel Wheeler (Laboratory for Physical Sciences, University of Maryland), Grigori Fursin (cTuning Foundation/INRIA), Sangyeun Cho (University of Pittsburgh) and Rami Melhem (University of Pittsburgh) for discussion on simulation and modeling. We also thank Tim Parenti and Brian Dicks for organizing the workshop and characterizing simulators and their lifetimes.

9. REFERENCES

[1] T. M. Aamodt and A. Boktor. GPGPU-Sim 3.x: A performance simulator for many-core accelerator research. *International Symposium on Computer Architecture (ISCA)*, `http://www.gpgpu-sim.org/isca2012-tutorial/`, 2012. [Online; accessed 30-Sep-2012].

[2] T. M. Aamodt and W. W. Fung. GPGPU-Sim 3.x: A performance simulator for manycore accelerator research. *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, `http://www.gpgpu-sim.org/pact2012-tutorial/`, 2012. [Online; accessed 30-Sep-2012].

[3] ArtistDesign NoE — European Network of Excellence on High Performance and Embedded Architecture and Compilation. `http://www.artist-embedded.org/`, 2012. [Online; accessed 21-Sep-2012].

[4] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt. Analyzing cuda workloads using a detailed GPU simulator. In *2012 IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS)*, pages 163–174, 2009.

[5] B. Beckmann, N. Binkert, A. Saidi, J. Hestness, G. Black, K. Sewell, and D. Hower. gem5: A multiple-ISA full system simulator with detailed memory modeling. Tutorial at the *International Symposium on Computer Architecture (ISCA)*,`http://www.m5sim.org/`. [Online; accessed 12-Jul-2012].

[6] F. Bellard. QEMU, a fast and portable dynamic translator. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '05, pages 41–41, Berkeley, CA, USA, 2005. USENIX Association.

[7] 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*, 39(2):1–7, Aug. 2011.

[8] R. Campbell, I. Gupta, M. Heath, S. Y. Ko, M. Kozuch, M. Kunze, T. Kwan, K. Lai, H. Y. Lee, M. Lyons, D. Milojicic, D. O'Hallaron, and Y. C. Soh. Open Cirrus(TM) cloud computing testbed: federated data centers for open source systems and services research. In *Proceedings of the 2009 conference on Hot topics in cloud computing*, HotCloud'09, Berkeley, CA, USA, 2009. USENIX Association.

[9] J. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson. Seattle: a platform for educational cloud computing. In *Proceedings of the 40th ACM technical symposium on Computer science education*, SIGCSE '09, pages 111–115, New York, NY, USA, 2009. ACM.

[10] T. E. Carlson, W. Heirman, and L. Eeckhout. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov. 2011.

[11] Chameleon. `http://www.chameleoncloud.org/`, 2014. [Online; accessed 13-Nov-2014].

[12] D. Chiou, D. Sunwoo, J. Kim, N. A. Patil, W. Reinhart, D. E. Johnson, J. Keefe, and H. Angepat. Fpga-accelerated simulation technologies (fast): Fast, full-system, cycle-accurate simulators. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 40, pages 249–261, Washington, DC, USA, 2007. IEEE Computer Society.

[13] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12, July 2003.

[14] E. S. Chung, E. Nurvitadhi, J. C. Hoe, B. Falsafi, and K. Mai. A complexity-effective architecture for accelerating full-system multiprocessor simulations using fpgas. In *Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays*, FPGA '08, pages 77–86, New York, NY, USA, 2008. ACM.

[15] CloudLab. `http://www.cloudlab.us/`, 2014. [Online; accessed 13-Nov-2014].

[16] CMU Artificial Intelligence Repository. `http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/0.html`, 2012. [Online; accessed 02-Jan-2012].

[17] L. Eeckhout, W. Heirman, T. E. Carlson, and I. Hur. Sniper: Fast, accurate and scalable multicore simulation. Tutorial at the *International Symposium on Computer Architecture (ISCA)*,`//http://snipersim.org/w/Tutorial:ISCA_2012`. [Online; accessed 30-Sep-2012].

[18] Emulab — total network testbed. `http://www.emulab.net`, 2012. [Online; accessed 18-Sep-2012].

[19] FPGA-Accelerator Research Infrastructure Cloud (FAbRIC). `http://users.ece.utexas.edu/~derek/FABRIC.html`. [Online; accessed 29-Sep-2012].

[20] The gem5 simulator system. `http://www.gem5.org/`. [Online; accessed 12-Jul-2012].

[21] D. Genbrugge, S. Eyerman, and L. Eeckhout. Interval simulation: Raising the level of abstraction in architectural simulation. In *16th International Conference on High-Performance Computer Architecture (HPCA)*, pages 1–12, 2010.

[22] K. Ghose, A. Patel, F. Afram, H. Zheng, J. Tringali, and J. Greensky. Marssx86 tutorial. Tutorial at *Conference on Parallel Architecture and Compilation Techniques (PACT)*, `http://marss86.org/~marss86/index.php/PACT_Tutorial_2012`. [Online; accessed 24-Sep-2012].

[23] M. Hibler, R. Ricci, L. Stoller, J. Duerig, S. Guruprasad, T. Stack, K. Webb, and J. Lepreau. Large-scale virtualization in the emulab network testbed. In *USENIX 2008 Annual Technical Conference on Annual Technical Conference*, ATC'08, pages 113–128, Berkeley, CA, USA, 2008. USENIX Association.

[24] HiPEAC Compilation Architecture — European Network of Excellence on High Performance and Embedded Architecture and Compilation. `http://www.hipeac.net/`, 2012. [Online; accessed 27-Sep-2012].

[25] Intel Many-core Applications Research Community (MARC). `http://communities.intel.com/community/marc`, 2012. [Online; accessed 17-Mar-2012].

[26] C. L. Janssen, H. Adalsteinsson, and J. P. Kenny. Using simulation to design extremescale applications and architectures: programming model exploration. *SIGMETRICS Perform. Eval. Rev.*, 38(4):4–8, Mar. 2011.

[27] A. Kerr, G. Diamos, and S. Yalamanchili. A characterization and analysis of PTX kernels. In *International Symposium on Workload Characterization (IISWC)*, 2009.

[28] A. Kerr, G. Diamos, and S. Yalamanchili. GPU application development, debugging, and performance tuning with GPU Ocelot. In Wen-mei Hwu, editor,

GPU Computing GEMS, vol. 2. Morgan Kaufmann, 2011.

[29] H. Kim, S. Yalamanchili, J. Lee, N. Lakshminarayana, A. Kerr, A. Rodrigues, and G. Hsieh. Tutorial on Ocelot and SST-MacSim Simulator. *International Symposium on Computer Architecture*, `http://comparch.gatech.edu/hparch/isca12_gt.html`, 2012. [Online; accessed 15-Sep-2012].

[30] H. Kim, S. Yalamanchili, J. Lee, N. Lakshminarayana, A. Kerr, A. Rodrigues, and G. Hsieh. Tutorial on Ocelot and SST-MacSim Simulator. *Conference on High-Performance Computer Architecture*, `http://comparch.gatech.edu/hparch/OcelotMacSim_tutorial.html`, 2012. [Online; accessed 15-Sep-2012].

[31] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hållberg, J. Högberg, F. Larsson, A. Moestedt, and B. Werner. Simics: A full system simulation platform. *Computer*, 35(2):50–58, Feb. 2002.

[32] Manifold parallel discrete event simulation. `http://manifold.gatech.edu`. [Online; accessed 29-Sep-2012].

[33] J. Miller, H. Kasture, G. Kurian, C. Gruenwald, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal. Graphite: A distributed parallel simulator for multicores. In *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, pages 1–12, jan. 2010.

[34] nanoHUB.org — Online Simulation and More for Nanotechnology. `nanohub.org`. [Online; accessed 20-Sep-2012].

[35] OpenCirrus(TM) the HP/Intel/Yahoo! Open Cloud Computing Research Testbed. `http://opencirrus.org`, 2012. [Online; accessed 15-Sep-2012].

[36] M. Pellauer, M. Vijayaraghavan, M. Adler, Arvind, and J. Emer. A-ports: an efficient abstraction for cycle-accurate performance models on fpgas. In *Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays*, FPGA '08, pages 87–96, New York, NY, USA, 2008. ACM.

[37] D. A. Penry, D. Fay, D. Hodgdon, R. Wells, G. Schelle, D. I. August, and D. Connors. Exploiting parallelism and structure to accelerate the simulation of chip multi-processors. In *12th International Symposium on High-Performance Computer Architecture HPCA*, pages 29–40, 2006.

[38] L. Peterson, A. Bavier, M. E. Fiuczynski, and S. Muir. Experiences building planetlab. In *Proceedings of the 7th symposium on Operating systems design and implementation*, OSDI '06, pages 351–366, Berkeley, CA, USA, 2006. USENIX Association.

[39] PlanetLab — An Open Platform for Developing, Deploying and Accessing Planetary-Scale Services. `http://www.planet-lab.org/`, 2012. [Online; accessed 27-Sep-2012].

[40] P. Ren, M. Lis, M. H. Cho, K. S. Shim, C. Fletcher, O. Khan, N. Zheng, and S. Devadas. Hornet: A cycle-level multicore simulator. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(6):890–903, June 2012.

[41] A. F. Rodrigues, K. S. Hemmert, B. W. Barrett, C. Kersey, R. Oldfield, M. Weston, R. Risen, J. Cook, P. Rosenfeld, E. CooperBalls, and B. Jacob. The structural simulation toolkit (SST). *SIGMETRICS Perform. Eval. Rev.*, 38(4):37–42, Mar. 2011.

[42] Seattle — Open Peer-to-Peer Computing. `http://seattle.cs.washington.edu`, 2012. [Online; accessed 18-Sep-2012].

[43] SST: The Structural Simulation Toolkit. `http://sst.sandia.gov`, 2012. [Online; accessed 12-Aug-2012].

[44] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli. Multi2Sim: A Simulation Framework for CPU-GPU Computing . In *Proc. of the 21st International Conference on Parallel Architectures and Compilation Techniques*, Sep. 2012.

[45] XSEDE: Extreme Science and Engineering Discovery Environment. `http://www.xsede.org`, 2012. [Online; accessed 15-Sep-2012].

APPENDIX

A. WORKSHOP PARTICIPANTS

Ahmed Amer (Santa Clara University)
Christopher Batten (Cornell University)
Nathan Binkert (Hewlett-Packard Labs)
Bruce Childers (University of Pittsburgh)
Derek Chiou (University of Texas at Austin)
Sangyeun Cho (University of Pittsburgh)
Almadena Chtchelkanova (National Science Foundation)
John Davis (Microsoft Corporation)
Lieven Eeckhout (Ghent University)
Jean-Luc Gaudiot (University of California – Irvine)
James Hoe (Carnegie Mellon University)
Engin Ipek (University of Rochester)
Alex Jones (University of Pittsburgh)
Hyesoon Kim (Georgia Institute of Technology)
Martha Kim (Columbia University)
Michael Kistler (IBM Austin Research Laboratory)
Jack Lange (University of Pittsburgh)
Benjamin Lee (Duke University)
Ahmed Louri (National Science Foundation)
Nicolas Maillard (Federal University of Rio Grande do Sul)
Jason Mars (University of Virginia)
Chris Mineo (Laboratory for Physical Sciences, University of Maryland / Department of Defense)
Daniel Mossé (University of Pittsburgh)
Steve Poole (Oak Ridge National Laboratory / Department of Defense)
Steven Reinhardt (Advanced Micro Devices, Inc.)
Arun Rodrigues (Sandia National Laboratories)
Eric Rotenberg (North Carolina State University)
Sonia Sachs (Department of Energy)
Ali Saidi (ARM, Ltd.)
Kevin Skadron (University of Virginia)
Evan Speight (IBM Austin Research Laboratory)
James Tringali (Rambus, Inc.)
David Wood (University of Wisconsin – Madison)
Sudhakar Yalamanchili (Georgia Institute of Technology)
Li Zhao (Intel Corporation)