1.0 The Goal: Infrastructure for Diverse System Modeling

In the decade that I have been part of the computer architecture community, computer system modeling has advanced dramatically. Through the generous efforts of researchers, tools such as SimOS, RSIM, Dinero, and SimpleScalar have worked to accelerate the pace of research by lowering the barriers to entry into the research community and by providing a common substrate upon which researchers could share results, workloads, and even code. Each of these tools grew out of a growing vector of research in which early work in the area defined the requirements of the infrastructure and eventually the design of the tool. Popular tools, like those listed earlier, are so because they serve these research agendas well. Looking forward, performance modeling tools and techniques must continue to serve their potential users’ research agenda.

For computer system modeling tools, demands will likely grow as the landscape of research is changing at a faster pace than in recent decades. While much of the work of the 80’s and 90’s was focused on performance optimization of general purpose processors, it is clear from the literature and funding sources that computer architecture research has become more diverse. Future performance analysis tools will need to address the increased demands of these new endeavors. A quick perusal of the literature reveals many concerns beyond general purpose processor performance:

- **Physical Artifact Effects** including cycle-timing impacts due to wire latency, the effects of circuit critical path modifications, and area costs for detailed cost-performance analysis.

- **Embedded System Design and Analysis** including support for graphics and digital signal processing, application-specific processors, heterogeneous multiprocessors, and a wide array of sensors and devices.

- **Power Analysis and Optimization** for both static and dynamic power dissipation with the added requirement that analysis infrastructure support tradeoffs in accuracy and speed.

- **Reliable System Design and Analysis** that requires realistic fault injection models with support for fault classification and coverage analysis.

To meet these diverse modeling goals, tool developers will have focus their efforts on creating modeling infrastructure, rather than a particular tool or model. The difference between a tool and an infrastructure lies in the care taken in designing the internal modules and interfaces. For an infrastructure, internal modules and interfaces must be organized in a way that permits their reuse over a wide range of modeling tasks. An infrastructure that supports modeling of diverse devices will undoubtedly require a large collection of reusable modules. Moreover, module
interfaces must be sufficiently expressive and well documented, and these interfaces must be extensible to accommodate future device requirements. Finally, a user-friendly infrastructure will require a high-level specification mechanism that permits users to efficiently specify module requirements, their shared datapaths, and the accompanying control structures. A pernicious tension exists here in that efficient specification frameworks require high-level abstractions, but high-level abstractions reduce the flexibility of the infrastructure and make it difficult to perform low-level design optimizations. Crafting a successful balance between all these concerns will in itself be a rich research effort.

2.0 The Challenge: Design Effort and Cost

I doubt that there will be much disagreement among us as to the need to increase the degree of diversity supported in system modeling, and to do so in a way that provides a reusable component-based infrastructure. Moreover, I have full confidence that my colleagues in the computer architecture research community have the expertise necessary to define a viable infrastructure. The key challenge, as I see it, is a non-technical one in that the design effort and cost necessary to create the next generation of infrastructures is immense.

There are three primary investments that will have to be made to create future modeling infrastructure: development, verification, and support. For SimpleScalar, code development has been a large investment, although, not the largest. For example, the SimpleScalar/x86 target being developed by SimpleScalar LLC, took nearly two person-years to develop. A potential pitfall that can extend development time unnecessarily is too strong a focus on modeling accuracy. While increased model detail leads to more accurate results, it is always at the expense of design time and ease of use for the resulting model. As a point of reference, I had the opportunity to conduct research at Intel using the Pentium 4 product simulator. In one effort, I ported many of the ideas from my thesis, which were written for SimpleScalar, to the Pentium 4 model. Working with the highly detailed product simulator (at more than 200k lines of code) was a glacial process. Since returning to academia, I have found it quite refreshing to return to SimpleScalar where our simpler, albeit less accurate, models (all under 5k lines of code) have accelerated the pace of my work. As an alternative to high accuracy, I would encourage researchers to focus on drawing clear insights from their work. It is these insights that can most effectively communicate ideas from those that do research to those that use research to build products.

Verification often requires more effort than code development, especially if one wishes to create robust implementations that work for a wide array of workloads on different host platforms. For example, University of Michigan is developing an ARM target for SimpleScalar that supports device-level emulation of a Compaq iPaq PDA. The model is sufficiently detailed that it can boot a stock version of Linux. The implementation of the device models for the SimpleScalar/ARM target took about 4 person-months to implement; the verification of this development has been ongoing for 4 person-months at the time of this writing, and we’ve budgeted an additional 6 person-months to complete the effort. Moreover, I now budget a significant amount of development resources toward verification tools, as they can dramatically reduce the cost of building robust infrastructure. For example, during the implementation of SimpleScalar/ARM at University of Michigan, we constructed a random test generator that could compare the results of random instructions on the simulator to actual hardware results. The tester took weeks to develop, but likely saved us months of debug time.

Finally, the amount of effort required to support an open infrastructure like SimpleScalar is striking. The current user base is estimated to be approximately 500 researchers and likely twice
as many students. The support load manifests in two forms: direct support for user questions and code maintenance. Direct user support is by far the largest demand; the primary developers of SimpleScalar answer about 10 e-mails per week, many more just before popular conference deadlines. Most questions are from new users experiencing difficulty installing or running SimpleScalar. More experienced users will often write the SimpleScalar developers seeking advice on how to implement a particular new feature. A smaller portion of the support load is the continual process of code maintenance. To keep the support load as light as possible, we have devised a number of proactive mechanisms to support users including FAQs, mailing lists and e-mail support.

3.0 One Remedy: A Healthy Economy of Tool Innovation

Given the high cost of developing performance analysis infrastructure and the likely higher cost of next generation infrastructures that supports more diverse models, how can we create an economy of innovation that makes it profitable for would-be developers to contribute? By “economy”, I do not mean a monetary one. I can assure my colleagues that there is little money to be made in computer system simulation; the market is simply too small to create any sustainable revenue stream. Instead, we need to create an economy based on intellectual and academic “currencies”. What follows is a list of suggested goals that I believe would lead to a more healthy economy of tool innovation:

Reduce development cost by building on existing infrastructures. SimpleScalar version 4.0, for example, represents nearly 20 person-years of development. I want to encourage future infrastructure developers to adapt and modify what they can from SimpleScalar. The flexible licensing and open source distribution makes this possible without undue burden to the developers, as evidenced by the growing body of tools (e.g., SimplePower, Wattch, HydraScalar) based on SimpleScalar.

Improve return on investment for those that contribute to the building of next generation infrastructure. As such, I would like to strongly encourage funding agencies, private funding institutions, and proposal reviewers to give strong favor to performance infrastructure development proposals, for both existing tools and newer speculative endeavors. NSF and DARPA have been quite generous in this regard, in particular in the case of SimpleScalar, and I feel that the return has been positive and easily measurable. The research community also shares a responsibility to place value on the output of those that create infrastructure through publication of tools papers and recognition of these contributions as equally valuable as the research they enable. This remedy need not cost a dime, and its return would benefit all.

I am optimistic that within a healthy economy of innovation, the next generation of performance analysis infrastructure will emerge and serve an even more vital role to the research endeavors that inspired it.