Leveraging open source simulators for HPC codesign

Bobby R. Bruce[1], Jason Lowe-Power[1], and Matthew D. Sinclair[2]

[1] *University of California, Davis, {bbruce, jlowepower}@ucdavis.edu*
[2] *University of Wisconsin-Madison, sinclair@cs.wisc.edu*

Topics architecture, modeling and simulation, codesign methodologies

Challenge *We need agile iteration of software and hardware changes in order to achieve hardware-software codesign. As such, we require accurate, modular, and high speed simulators to enable this iteration.* Current simulation techniques are either too high level, too inflexible, or too slow to enable codesign for exascale and post-exascale HPC architectures.

As Moore's Law continues to fade, future performance improvements are going to need to rely more on codesign between multiple layers of the hardware-software stack [3]. Codesign, by definition, involves modifying many levels of the hardware-software stack, such as the architecture, the application runtime, the operating system, compilers, computer networks, and the batch scheduler. Traditionally, many of these layers have relied on proprietary, closed source components, which makes research on next-generation solutions difficult, if not impossible. There is a new opportunity for researching codesign with modern open source platforms such as RISC-V, LLVM, and Linux. These platforms make it possible for researchers to modify *every level* of this stack. However, when making these modifications, researchers need a way to predict their performance impact before undergoing the long and costly process of engineering real-world hardware. We therefore need simulation and modeling which can capture these *full system* effects as the benefits of codesign can only come from understanding the underlying interactions between different parts of the hardware-software stack.

Unfortunately, current modeling and simulation techniques suffer from poor performance, insufficient detail to capture cross-stack optimizations, and are difficult to modify and extend. Full-system cycle-level simulators like gem5 [4] can capture the hardware-software interactions since they support all levels of the stack including the OS, runtime, and unmodified software, but cycle-level simulators are slow (at least 10,000× slowdown) and incapable of simulating large-scale applications in a reasonable time frame. Other systems such as SST [5] scale to modeling full HPC centers, but do so at the cost of modeling detail, often falling back on trace-based or analytic models which do not have enough detail to model important codesign components like the application runtime. Finally, fast FPGA-accelerated simulators like FireSim [2] are capable of running full-systems and mitigate some of the performance issues with software-only simulation systems, but are inflexible, often requiring fully implemented RTL designs. These FPGA-based simulation systems can be useful for tweaking current designs, but are not flexible enough to explore entirely novel architectural designs (e.g., new accelerators, which HPC systems are widely adopting).

Opportunity We need to develop new modeling and simulation techniques to understand the performance impact of novel codesigns at an HPC-center scale and that provide accurate performance predictions in tolerable timeframes. Since no single methodology can provide the required accuracy, detail, or scale, we believe the path forward is through "multi-fidelity simulation." We aim to leverage multi-fidelity simulation in two axes: over time and over space. We envision a unified simulation framework which integrates the best in class models—the fastest with good high-level behavioral accuracy and the most detailed with cycle-level accuracy—in the same simulation instance in both time and space.

Leveraging different fidelity simulation *over time* has been a popular technique to improve the performance and energy of detailed simulation techniques. Examples include SimPoints, SMARTS, and the BarrierPoint sampling methodologies. Recently, researchers have leveraged analytical models to improve cache warmup time which dominates most of the sampled simulation techniques, and used hardware virtualization platforms to fast-forward simulators at native speed (i.e., with no slowdown) to the region of interest or the sampling location. Although many of these techniques have been published, few have been integrated into any widely-used simulation system in a straightforwardly useable fashion. Thus, there is an opportunity to provide codesigners with easy-to-use simulation frameworks which natively support multi-fidelity simulation over time.

While multi-fidelity simulation over time has been previously explored, combining different fidelity models *at the same time* in a single simulation is a new research direction—which we define as multi-fidelity over space. For instance, to simulate an entire compute rack running one application, it is likely unnecessary to simulate all systems at a high fidelity (e.g., cycle level). Instead, we can simulate one or two systems at a cycle level and use higher level models (e.g., trace based or analytical models) for the other systems in the rack. This technique can also be applied within a single node. For instance, we can simulate the processors with a high-level "simple" out-of-order model which instead of providing detailed representation of every structure, provides simple configuration options for components like the instruction window and commit width, while at the same time uses a cycle-level memory system design to investigate the impact of software-cache coherence codesign.

To enable multi-fidelity simulation, we must have a modular, composable, and standardized simulation framework. This framework will allow codesigners to combine different fidelity models in both time and space. There is an opportunity to develop a simulator backplane which will allow modular simulators to share "best in class" models which may operate at different fidelities. We believe that building off of today's modular simulation infrastructures such as gem5 or SST is a viable direction to create this simulator backplane. There is evidence that these simulator systems can integrate other models such as gem5-gpu which integrates GPGPU-Sim with gem5, Emerald which integrates graphics simulation with gem5, gem5-Aladdin and gem5-SALAM which integrate auto-generated accelerators with gem5, and many others. Similarly, SST Elements provides a broad set of simulator integrations for SST. However, these integrated models are not readily available in a centralized easy to use and access location. Additionally, we need to improve the usability and composability of current simulation platforms and expand the availability of multi-fidelity models.

Timeliness or Maturity It is now assumed that future gains in performance are going to come above the level of silicon [3] and performance improvements will increasingly come from hardware-software codesign. Now is the time to build the vital infrastructure to make this possible. This infrastructure will, in large part, consist of simulators which can deliver reliable simulations in acceptable time scales. Without such an infrastructure, it will be extremely difficult for future generations of system designers to develop their ideas, without resorting to time consuming RTL designs or relatively expensive chip tapeouts. Thus, developing this next generation simulation infrastructure will enable system developers to rapidly prototype, test, and iterate on their ideas while being confident that the infrastructure is representative of modern HPC systems.

Work has already began in this area. In 2020 gem5 released it's first stable version as part of an ongoing effort to improve the stability of the project. Documented APIs have been added with guarantees of stability between versions, thereby making a start on the engineering effort necessary for a common architecture backend which other tools may integrate with. This stability will help foster and support a common, community-based infrastructure. Getting value out of this backend will require forming collaborations between open source architecture projects, and ensuring common standards and processes are agreed upon. Moreover, developing such an infrastructure with common standards and processes will enable additional projects, such as those that enable accelerated simulation of RTL [1], to easily integrate into the infrastructure. While difficult to achieve, such an effort would be of immense benefit to researchers investigating codesign and beyond.

References

[1] S. Beamer. A Case for Accelerating Software RTL Simulation. *IEEE Micro*, 40(4):112–119, 2020.

[2] S. Karandikar et al. FireSim: FPGA-accelerated cycle-exact scale-out system simulation in the public cloud. In *ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 29–42. IEEE, 2018.

[3] C. E. Leiserson et al. Theres plenty of room at the Top: What will drive computer performance after Moores law? *Science*, 368(6495), 2020.

[4] J. Lowe-Power et al. The gem5 simulator: Version 20.0+. *arXiv preprint arXiv:2007.03152*, 2020.

[5] A. F. Rodrigues et al. The structural simulation toolkit. *SIGMETRICS Perform. Eval. Rev.*, 38(4):3742, Mar. 2011.