FireSim: FPGA-Accelerated Cycle-Exact Scale-Out System Simulation in the Public Cloud

Sagar Karandikar, Howard Mao, Donggyu Kim, David Biancolin, Alon Amid, Dayeol Lee, Nathan Pemberton, Emmanuel Amaro, Colin Schmidt, Aditya Chopra, Qijing Huang, Kyle Kovacs, Borivoje Nikolić, Randy Katz, Jonathan Bachrach, Krste Asanović

Department of Electrical Engineering and Computer Sciences, University of California, Berkeley
{*sagark, zhemao, dgkim, biancolin, alonamid, dayeol, nathanp, amaro, colins, adichopra, qijing.huang, kylekovacs, bora, randy, jrb, krste*}*@eecs.berkeley.edu*

***Abstract*—We present FireSim, an open-source simulation platform that enables fast cycle-exact microarchitectural simulation of large scale-out clusters by combining FPGA-accelerated simulation of silicon-proven RTL designs with scalable, distributed network simulation, running on a public-cloud host platform. By introducing automation and harnessing cloud FPGAs, FireSim provides the usability and productivity of software full-system simulators with the high-performance and accuracy of FPGA-accelerated simulation, while adding the unprecedented ability to scale to globally-cycle-accurate simulations of thousands of networked nodes. To demonstrate FireSim's scalability, we automatically generate and deploy a target cluster simulation of 1,024 3.2 GHz quad-core server nodes, each with 16 GB of DRAM, interconnected by a 200 Gbit/s network with low latency, which simulates at a 6.6 MHz processor clock rate (<500x slowdown over real-time). In aggregate, this simulation harnesses millions of dollars of FPGAs—at a cost of only hundreds of dollars per simulation-hour to users.**

I. Introduction

The demand for ever more powerful warehouse-scale computers (WSCs) continues to grow, to support new compute-intensive applications deployed on billions of edge devices as well as conventional computing workloads migrating to the cloud. While the first few generations of WSCs were built with standard servers, hardware and application trends are pushing datacenter architects towards building warehouse-scale machines that are increasingly specialized and tightly integrated [1]. These hardware trends include the end of general-purpose processor performance scaling, the continued scaling of network performance, new memory technologies, and new *disaggregated* datacenter architectures. To support modern web-scale services, application and systems framework developers expect the ability to deploy fine-grained tasks, where task latencies are measured in microseconds.

These trends push the boundaries of hardware-software co-design at-scale. Architects can no longer simply simulate individual nodes and leave the issues of scale to post-silicon measurement. Additionally, the introduction of custom silicon in the cloud to augment general-purpose processing means that architects must model emerging hardware, not only well-understood processor microarchitectures. Hardware-software co-design studies targeting next-generation WSCs are hampered by a lack of a scalable and performant simulation environment. Modifying microarchitectural software simulators to model scale-out systems [2], [3] is hampered by the low simulation speeds (5–100 KIPS) of the underlying single-server software simulator. Fast custom-built simulation hardware has been proposed [4], but is difficult to modify and involves considerable capital expense, which limits access for most academic and industrial researchers.

To address these limitations, we present FireSim[1,2] [5], a fast, open-source, cycle-exact FPGA-accelerated simulation framework that can simulate large clusters, including high-bandwidth, low-latency networks, on a public-cloud host platform.

II. Productive FPGA-Accelerated Simulation with Cloud-Hosted FPGAs

Architects experience many challenges when building and using FPGA-accelerated simulation platforms. FPGA platforms are unwieldy, especially compared to commodity servers used by software simulators. Traditional FPGA platforms are constrained by high prices, individual platform quirks that make reproducibility difficult, the need to provision for maximum utilization at time of initial purchase, and long build times. Even when FPGA pricing is insignificant to a project, building a custom rack of large FPGAs requires significant operations experience and makes it extremely difficult to share and reproduce research prototypes.

Several cloud providers have recently integrated FPGAs into their cloud services, including Amazon, Microsoft, Huawei, and Alibaba. Amazon makes FPGAs available as part of its EC2 F1 *public* cloud offering, allowing developers

[1]https://fires.im
[2]https://github.com/firesim/firesim

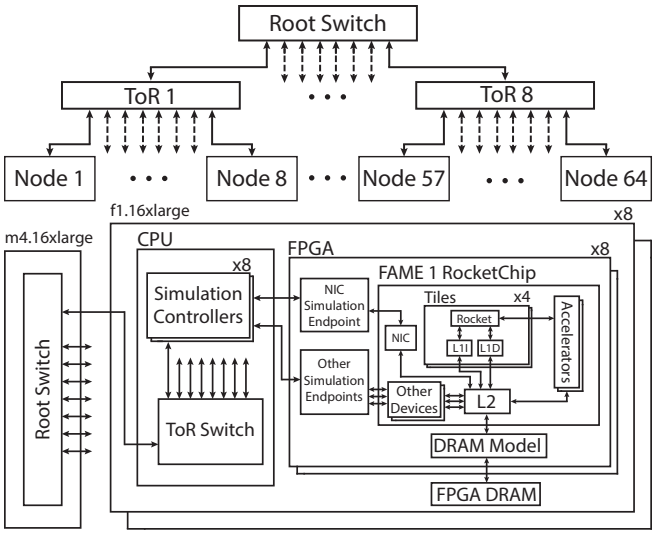

Figure 1. Target view (top) and mapping (bottom) of a 64-node simulation to EC2 F1 in FireSim.

to directly design FPGA-based applications that run in the cloud. Using an FPGA-enabled public cloud platform addresses many of the traditional issues with FPGA-based hardware simulation by providing elasticity, scalability, and reduction in capital expenditure.

Because of these benefits, Amazon EC2 F1 forms a natural platform on which to build the scalable FireSim environment. Amazon's EC2 F1 offering provides three new EC2 instance types, `f1.2xlarge`, `f1.4xlarge`, and `f1.16xlarge`, which consist of a powerful host instance attached by PCIe to 1, 2, or 8 Xilinx Virtex UltraScale+ FPGAs. Each FPGA contains 64 GB of DRAM onboard across 4 channels, making it an ideal platform for prototyping servers. FireSim can automatically provision and scale across large numbers of these host instances to run simulations and build FPGA images.

III. FireSim

FireSim [5] models a target system containing a collection of server blades connected by some form of network. The target server blades are modeled using FAME-1 models [6] automatically derived from the RTL of the server SoCs and mapped onto FPGA instances, while the target network is modeled with high-performance, cycle-by-cycle C++ switch models running on host server instances. These two target components are interconnected by a high-performance simulation token transport that models target link characteristics. Figure 1 shows the target topology and target-to-host mapping for a 64-node simulation with 8 top-of-rack (ToR) switches and one root switch, which we use as an example throughout this section.

A. Server Blade Simulation

1) Target Server Design: FireSim compute servers are derived from the Rocket Chip SoC generator [7], which is an SoC generation library written in Chisel. Rocket Chip can produce Verilog RTL for a complete processor system, including the RISC-V Rocket CPU, L1 and L2 caches, custom accelerators, and I/O peripherals. Concretely, the server blades we model in this work each consist of 1 to 4 RISC-V Rocket Cores modeled at 3.2 GHz, 16 KiB private L1 I/D Caches, a 256 KiB shared L2, 16 GiB of DDR3, a 200 Gbit/s Ethernet NIC, optional RoCC accelerators, and a disk controller. When we refer to a particular frequency f for Rocket Chip, for example 3.2 GHz, this implies that all models that require a notion of target time in the simulation (e.g., the network) assume that 1 cycle is equivalent to $1/f$ seconds. The "FAME-1 Rocket Chip" box in Figure 1 provides a sample block diagram of a Rocket Chip server node. To produce a complete server blade, we implement two new hardware components as tapeout-ready Chisel RTL: a block device controller to interface with a disk model (e.g. to boot custom Linux distributions with large root filesystems) and an on-die Ethernet network interface controller (NIC) with a corresponding RISC-V Linux driver, described in detail in our full paper [5].

2) Cycle-Exact Server Simulations from RTL: We use the FAME-1 [6] transforms provided by the MIDAS/Strober frameworks [8], [9] to translate the server designs written in Chisel into RTL with decoupled I/O interfaces for use in simulation. Each target cycle, the transformed RTL on the FPGA expects a token on each input interface to supply input data for that target cycle and produces a token on each output interface to feed to the rest of the simulated environment. If any input of the SoC does not have an input token for that target cycle, simulation stalls until a token arrives. This allows for timing-accurate modeling of I/O attached to custom RTL on the FPGA. To provide a cycle-accurate DDR3 DRAM model for our target servers, we use a synthesizable DRAM timing model, FASED [8], backed by the host FPGA's on-board DRAM. Other I/O interfaces (UART, Block Device, NIC) communicate with a software driver ("simulation controller" in Figure 1) on the host CPU core, which implements both timing and functional request handling (e.g. fetching disk blocks). Since in this work we are primarily interested in scaling to large clusters and network modeling, we focus on the implementation of the network token-transport mechanism used to globally coordinate simulation target time between the FAME-1-transformed server nodes.

3) Improving Scalability and Utilization: In addition to the previously described configuration, FireSim includes an additional "supernode" configuration, which simulates multiple complete target designs on each FPGA to provide improved utilization and scalability, restricted only by FPGA

resources. In our 1024-node simulation (Section V-B), we pack four quad-core server blade simulations onto each FPGA, allowing us to model a 32-node rack on each `f1.16xlarge` instance.

B. Network Simulation

1) Target Switch Modeling: Switches are modeled in software using a high-performance C++ switching model that processes network flits cycle-by-cycle. The switch models have a parameterizable number of ports, each of which interact with either a port on another switch or a simulated server NIC on the FPGA. Port bandwidth, link latency, amount of buffering, and switching latency are all parameterized and runtime-configurable.

The simulated switches implement store-and-forward switching of Ethernet frames. At ingress into the switch, individual simulation tokens that contain valid data are buffered into full packets, timestamped based on the arrival cycle of their last token, and placed into input packet queues. This step is parallelized using host OpenMP threads, with one thread per-port. The timestamps are also incremented by a configurable minimum switching latency to model the minimum port-to-port latency of datacenter switches. These timestamps are later used to determine when a packet can be released to an output buffer. A global switching step then takes all input packets available during the switching round, pushes them through a priority queue that sorts them on timestamp, and then drains the priority queue into the appropriate output port buffers based on a static MAC address table (since datacenter topologies are relatively fixed). In this step, packets are duplicated as necessary to handle broadcasts. Finally, in-parallel and per-port, output ports "release" packets to be sent on the link in simulation token form, based on the switch's notion of simulation time, the packet timestamp, and the amount of available space in the output token buffer. In essence, packets can be released when their release timestamp is less than or equal to global simulation time. Since the output token buffers are of a fixed size during each iteration, congestion is automatically modeled by packets not being able to be released due to full output buffers. Dropping due to buffer sizing and congestion is also modeled by placing an upper bound on the allowable delay between a packet's release timestamp and the global time, after which a packet is dropped. The switching algorithm described above and assumption of Ethernet as the link-layer is not fundamental to FireSim—a user can easily plug-in their own switching algorithm or their own link-layer protocol parsing code in C++ to model new switch designs.

2) High-performance Token Transport: From the target's view, endpoints on the network (either NICs or ports on switches) should communicate with one another through a link of a particular latency and bandwidth. On a simulated link, the fundamental unit of data transferred is a token that represents one target cycle's worth of data. Each target cycle, every NIC expects one input token and produces one output token in response. Each port on every switch also behaves in the same way. For a link with link latency of N cycles, N tokens are always "in-flight" on the link at any given time. That is, if a particular network endpoint issues a token at target cycle M, the token arrives at the other side of the link for consumption at target cycle $M + N$.

To simulate the 200 Gbit/s links we use throughout this work, the width of the data field in each token is set to 64 bits, since we assume that our simulated processor frequency is 3.2 GHz. In a distributed simulation as in FireSim, different host nodes are decoupled and can be executing different target cycles at the same time, but the exchange of these tokens ensures that each server simulation computes each target cycle deterministically, since all NICs and switch ports are connected to the same network and do not advance unless they have input tokens to consume.

In a datacenter topology, there are two types of links to model: links between a NIC and a switch port and links between two ports on different switches. Since we model switches in software and NICs (and servers) on FPGAs, these two types of links map to two different types of token transport. Transport between NICs and switch models requires two hops: a token must first cross the PCIe interface to an individual node's simulation driver, then be sent to a local switch model through shared memory or a remote switch model over a socket. To improve performance without causing deadlock, tokens are moved across host transports in batches, of up to the number of cycles of link latency.

3) Deploying/Mapping Simulations to EC2 F1: At this point, we have outlined each component necessary to build a large-scale cluster simulation in FireSim. However, without automation, the task of stitching together all of these components in a reliable and reproducible way is daunting. To overcome this challenge, the FireSim infrastructure includes a simulation manager that automatically builds and deploys simulations given a programmatically defined datacenter topology. That is, a user can write a configuration in a few lines of Python that describes a particular datacenter topology and server types for each server blade. The FireSim cluster manager takes this configuration and automatically runs the desired RTL through the FPGA build flow and generates the high-performance switch models and simulation controllers with the appropriate network token transports (shared memory, socket, PCIe transport). In particular, based on the given topology, simulated servers are automatically assigned MAC and IP addresses and the MAC switching tables in the switch models are automatically populated for each switch in the simulation. Once all component builds are complete, the manager flashes FPGAs on each F1 instance with the desired server configurations, deploys simulations and switch models as described by the user, and finally boots Linux (or other software) on the simulated nodes.

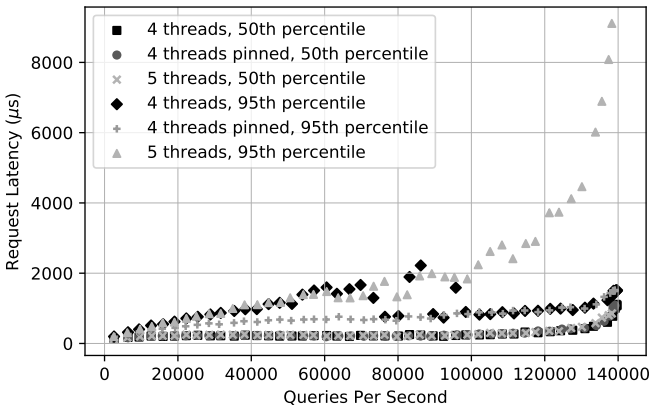

Figure 2. Reproducing the effect of thread imbalance on tail latency in memcached.

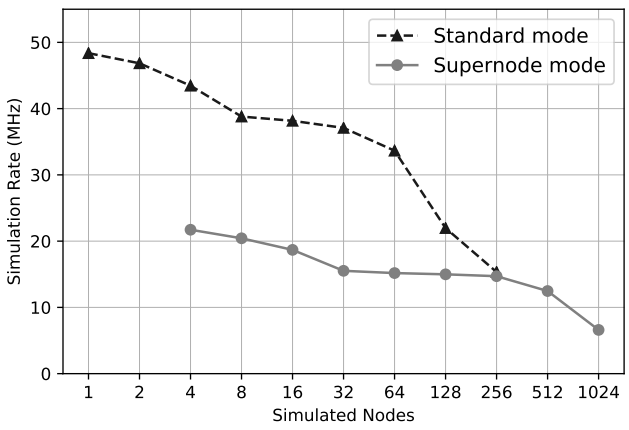

Figure 3. Simulation rate vs. # of simulated target nodes.

At the root switch, a special port can be added to the network that allows for direct ingress into the simulated datacenter network over SSH. That is, a user can *directly ssh* into the simulated system from the host machine and treat it as if it were a real cluster to deploy programs and collect results. Alternatively, a second layer of the cluster manager allows users to describe jobs that *automatically* run on the simulated cluster nodes and *automatically* collect result files and host/target-level measurements for analysis outside of the simulation. For example, the open release of FireSim includes reusable workload descriptions used by the manager to automatically run various versions of SPECint, boot other Linux distributions such as Fedora, or reproduce the experiments described later in this article[3], among others.

IV. Reproducing memcached QoS Phenomena from Deployed Commodity Clusters in FireSim

As an end-to-end validation of FireSim running a realistic datacenter workload, we run the `memcached` key-value store and use the `mutilate` memcached load-generator from Leverich and Kozyrakis [10] to benchmark our simulated system. While much simulation work has focused on reproducing well-known phenomena like the long-latency tail, we go further to validate a finer-grained phenomenon: thread imbalance in memcached servers when memcached uses more threads than the number of cores in the system. Reproducing this result involves interaction between the core microarchitecture, operating system, and network. Under thread imbalance, a sharp increase in tail latency has been shown, while median latency is relatively unchanged [10]. To replicate this result, we simulate an 8-node cluster in FireSim interconnected by a 200 Gbit/s, 2 μs latency network, where each simulated server has 4 cores. We provision one server in the simulation as a memcached host. We run the `mutilate` load generator on the remaining seven simulated blades to generate load on the memcached server. On the serving node, we configure memcached to run with either 4 or 5 threads and report median and tail (95th-percentile) latencies based on achieved queries per second. Figure 2 shows the results of this experiment. As expected from the literature [10], we observe thread imbalance when running with 5 threads—the tail latency is significantly worsened by the presence of the extra thread, while median latency is essentially unaffected. Our full paper describes several other interesting phenomena shown in Figure 2.

[3]http://docs.fires.im/en/latest/Advanced-Usage/Workloads/ISCA-2018-Experiments.html

V. Simulation Performance

A. Performance vs. target scale and link-latency

To show the overhead of token-based synchronization of all simulated nodes in clusters of varying size interconnected by a simulated 2 μs, 200 Gbit/s network, we run a benchmark that boots Linux to userspace, then immediately powers down the nodes in the cluster and reports simulation rate. Despite the lack of network traffic on the target, the network model must still exchange tokens on each link for each cycle to maintain global cycle-accuracy. This benchmark shows the overhead of distributing simulations, first between FPGAs on one instance, and then between FPGAs in different instances. Figure 3 shows the results of this benchmark, both for "standard" and "supernode" FPGA configurations. Our full paper also runs a similar benchmark varying link-latency rather than simulation scale to demonstrate that FireSim performs well even with links of varying latency (from 2.25 MHz at 50 ns link-latency to 50 MHz at 10 μs link latency).

B. Thousand-Node Datacenter Simulation

To demonstrate the scale achievable with FireSim, we run a simulation that models 1024$\times$3.2 GHz quad-core nodes, with 32 top-of-rack switches, 4 aggregation switches, and

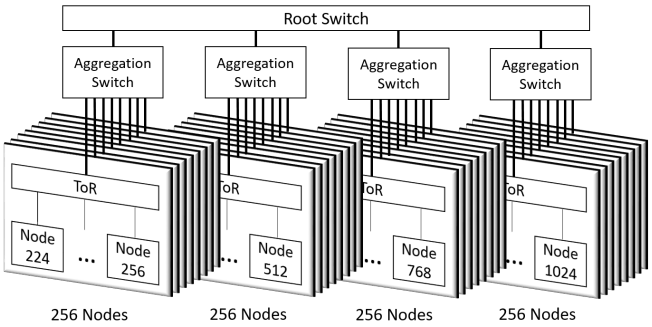

Figure 4. Topology of 1024-node datacenter simulation.

one root switch, all interconnected by a 2 μs, 200 Gbit/s network and arranged in a tree-topology, at a simulation rate of 6.6 MHz. This design represents a more realistic target design point than the example design used in Section III, since we make use of FireSim's "supernode" feature to pack four simulated nodes per FPGA, giving a total of 32 simulated nodes attached to each ToR switch. Figure 4 shows this topology in detail. Each ToR switch has 32 downlinks to nodes and one uplink to an aggregation switch. Each aggregation switch has eight downlinks, each to one ToR switch and one uplink to the root switch. Finally, the root switch has 4 downlinks to the 4 aggregation switches in the target topology. This topology is specified to the FireSim simulation manager with around 10 lines of configuration code. More complicated topologies, such as a fat-tree, can similarly be described in the manager configuration. Compared to existing software simulators, this instantiation of FireSim simulates an order of magnitude more nodes, with several orders of magnitude improved performance.

To map this simulation to EC2, we run 32 `f1.16xlarge` instances, which host ToR switch models and simulated server blades, and 5 `m4.16xlarge` instances to serve as aggregation and root-switch model hosts. The cost of this simulation can be calculated for two EC2 pricing models: spot instances (bidding on unused capacity) and on-demand (guaranteed instances). To calculate the spot price of this simulation, we use the longest stable prices in recent history, ignoring downward and upward spikes. This results in a total cost of ≈$100 per simulation hour. Using on-demand instances, which have fixed instance prices, this simulation costs ≈$440 per simulation hour. Using publicly listed retail prices of the FPGAs on EC2 (≈$50K each), this simulation harnesses ≈$12.8M worth of FPGAs. We expect that users will use cluster-scale experiments to prototype systems, with datacenter-scale experiments to analyze behavior at-scale once a system is already stable at cluster-scale.

VI. Related Work

In this abbreviated article, we cover a limited set of related work. Our full paper [5] discusses related work in detail.

A. Software Simulators

One approach to simulating warehouse-scale computers is to scale-out existing cycle-accurate full-system software simulators. For example, dist-gem5 [3] is a distributed version of the popular architectural simulator gem5. Software-based simulators are extremely flexible at the expense of performance—being several orders of magnitude slower than FPGA-accelerated simulation platforms. Software models of processors are also notoriously difficult to validate and calibrate against a real design [11], and do not directly provide reliable power and area numbers. By utilizing FPGAs in a cloud service and directly deriving simulations from silicon-proven RTL, FireSim matches many of the traditional flexibility advantages of software simulators, while maintaining cycle-exactness and high simulation performance.

B. Hardware-accelerated Simulators

Several proprietary tools exist for hardware-accelerated system simulation, such as Cadence Palladium, Mentor Veloce, and Synopsys Zebu. These systems are generally prohibitively expensive (≈millions of dollars) and thus only used by industrial design teams for single SoC projects.

Several prior projects used FPGAs to accelerate simulation of computer systems. The RAMP collaboration [12] pushed towards fast, productive FPGA-based evaluation for multi-core systems, and one of the RAMP simulators, DIABLO [4] is the most similar simulator to FireSim. Although DIABLO also uses FPGAs to simulate large scale-out systems, there are several significant differences between DIABLO and FireSim:

Automatically transformed RTL vs. Abstract Models. In DIABLO, servers are modeled using handwritten abstract RTL models. Authoring abstract RTL models is considerably more difficult than developing an actual design in RTL, and abstract RTL cannot be run through an ASIC flow to gain realistic power and area numbers. FireSim's simulated servers are built by directly applying FAME-1 transforms to tapeout-ready RTL to yield a simulator that has the exact cycle-by-cycle bit-by-bit behavior of the user-written RTL. Simulated switches in DIABLO are also abstract RTL models. In FireSim, users write abstract switch models in C++, making them considerably easier to modify.

Specialized vs. Commodity Host Platform. DIABLO used a custom-built FPGA platform that cost ≈$100K at publication time, excluding operation and maintenance costs. This cost and platform-dependency makes it difficult for other researchers to use DIABLO and reproduce results. In contrast, the entire FireSim codebase is open-source with substantial documentation and automation, which allows any user to easily deploy simulations on EC2 without the high cost of purchasing large numbers of FPGAs.

VII. Discussion and Future Work

Productive and reproducible FPGA-accelerated simulation for non-WSC targets. The large scale of FireSim experiments required us to build a simulation management framework to enable reliable and reproducible experimentation with thousands of nodes via automation. This capability is also useful in improving the productivity of FPGA-accelerated simulation for non-WSC targets, for example running workloads like SPECint on single-node systems. Harnessing FireSim's ability to distribute jobs to many parallel single-node simulations, users can run the entire SPECint17 benchmark suite on Rocket Chip-like systems with *full reference* inputs, and obtain cycle-exact results in roughly one day. In the future, we plan to also re-use the FireSim network simulation transport to support partitioning larger chip designs across many FPGAs.

Open-sourcing and adoption. FireSim is BSD-licensed open-source (https://github.com/firesim/firesim) and comes with 100+ pages of documentation (https://docs.fires.im). FireSim has a growing user community across industry (as a pre-silicon validation tool) and academia (with several user publications in ISCA, MICRO, and workshops). In addition to research usage, FireSim is used in Berkeley's undergraduate computer architecture course, allowing students to directly work with real implementations of fundamental architectural concepts. We plan to release these course materials to enable instructors to integrate FireSim into their curricula.

New target designs and new simulator features. Because FireSim automatically transforms RTL designs into FPGA simulators, supporting new user designs is straightforward. In addition to the RISC-V Rocket in-order core used in this work, FireSim now also supports the Berkeley Out-of-Order Machine, a superscalar OoO RISC-V implementation and Hwacha, a vector accelerator. Verilog designs have also been simulated in FireSim, including the NVIDIA Deep Learning Accelerator (NVDLA) and PicoRV32, a Verilog RISC-V design. FireSim also contains new debugging tools, including automatic logic analyzer insertion and commit log tracing, among others, which allow more introspection into designs running on the FPGA. We aim to continue improving FireSim's introspection and automation capabilities to provide the flexibility and ease-of-use of software simulators with the high-performance and accuracy of FPGA-accelerated simulation.

VIII. Conclusion

The open-source FireSim simulation platform represents a new approach to warehouse-scale architectural research, simultaneously supporting an unprecedented combination of *fidelity* (cycle-exact microarchitectural models derived from synthesizable RTL), *target scale* (4,096 processor cores connected by network switches), *flexibility* (modifiable to include arbitrary RTL and/or abstract models), *reproducibility*, target *software support*, and *performance* (less than 500× slowdown over real time), while using a *public FPGA-cloud platform* to remove upfront costs and provide large cluster simulations on-demand.

Acknowledgments

Research partially funded by DARPA Award Number HR0011-12-2-0016, ARPA-E Award Number DE-AR0000849, RISE Lab sponsor Amazon Web Services, ADEPT Lab industrial sponsor Intel, and ADEPT Lab affiliates Google, Huawei, Siemens, SK Hynix, and Seagate. Any opinions, findings, conclusions, or recommendations in this article are solely those of the authors and do not necessarily reflect the position or the policy of the sponsors.

References

[1] L. A. Barroso, U. Hölzle, and P. Ranganathan, "The datacenter as a computer: Designing warehouse-scale machines, third edition," *Synthesis Lectures on Computer Architecture*, vol. 13, 2018. [Online]. Available: https://doi.org/10.2200/S00874ED3V01Y201809CAC046

[2] S. Novakovic, A. Daglis, E. Bugnion, B. Falsafi, and B. Grot, "Scale-out NUMA," in *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '14. New York, NY, USA: ACM, 2014.

[3] A. Mohammad, U. Darbaz, G. Dozsa, S. Diestelhorst, D. Kim, and N. S. Kim, "dist-gem5: Distributed simulation of computer clusters," in *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, April 2017.

[4] Z. Tan, Z. Qian, X. Chen, K. Asanović, and D. Patterson, "DIABLO: A Warehouse-Scale Computer Network Simulator Using FPGAs," in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '15. New York, NY, USA: ACM, 2015.

[5] S. Karandikar, H. Mao, D. Kim, D. Biancolin, A. Amid, D. Lee *et al.*, "FireSim: FPGA-accelerated cycle-exact scale-out system simulation in the public cloud," in *Proceedings of the 45th Annual International Symposium on Computer Architecture*, ser. ISCA '18. Piscataway, NJ, USA: IEEE Press, 2018. [Online]. Available: https://doi.org/10.1109/ISCA.2018.00014

[6] Z. Tan, A. Waterman, H. Cook, S. Bird, K. Asanović, and D. Patterson, "A Case for FAME: FPGA Architecture Model Execution," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ser. ISCA '10. New York, NY, USA: ACM, 2010.

[7] K. Asanović, R. Avižienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio *et al.*, "The Rocket Chip Generator," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, Apr 2016.

[8] D. Biancolin, S. Karandikar, D. Kim, J. Koenig, A. Waterman, J. Bachrach *et al.*, "FASED: FPGA-accelerated simulation and evaluation of DRAM," in *The 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'19)*, ser. FPGA '19. New York, NY, USA: ACM, 2019. [Online]. Available: https://doi.acm.org/10.1145/3289602.3293894

[9] D. Kim, A. Izraelevitz, C. Celio, H. Kim, B. Zimmer, Y. Lee *et al.*, "Strober: Fast and Accurate Sample-based Energy Simulation for Arbitrary RTL," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16. Piscataway, NJ, USA: IEEE Press, 2016.

[10] J. Leverich and C. Kozyrakis, "Reconciling high server utilization and sub-millisecond quality-of-service," in *Proceedings of the Ninth European Conference on Computer Systems*, ser. EuroSys '14. New York, NY, USA: ACM, 2014.

[11] A. Gutierrez, J. Pusdesris, R. G. Dreslinski, T. Mudge, C. Sudanthi, C. D. Emmons *et al.*, "Sources of error in full-system simulation," in *ISPASS*. IEEE Computer Society, 2014.

[12] J. Wawrzynek, D. Patterson, M. Oskin, S. Lu, C. Kozyrakis, J. C. Hoe *et al.*, "RAMP: Research accelerator for multiple processors," *IEEE Micro*, vol. 27, March 2007.