Theme Article

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 Howard Katz, Jonathan Bachrach, and Krste Asanović
University of California,Berkeley

***Abstract*—In this article, 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 1024 3.2-GHz quad-core server nodes, each with 16 GB of DRAM, interconnected by a 200 Gb/s network with low latency, which simulates at a 6.6-MHz processor clock rate ($<500\times$ 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.**

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 toward 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

Digital Object Identifier 10.1109/MM.2019.2910175
Date of publication 11 April 2019; date of current version 8 May 2019.

Published by the IEEE Computer Society

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 codesign 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.

We present FireSim (https://fires.im, https://github.com/ firesim/ firesim), 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.

To address these limitations, we present FireSim (https://fires.im, https://github.com/firesim/firesim),[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.

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 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 four 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.

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 eight top-of-rack (ToR) switches and one root switch, which we use as an example throughout this section.

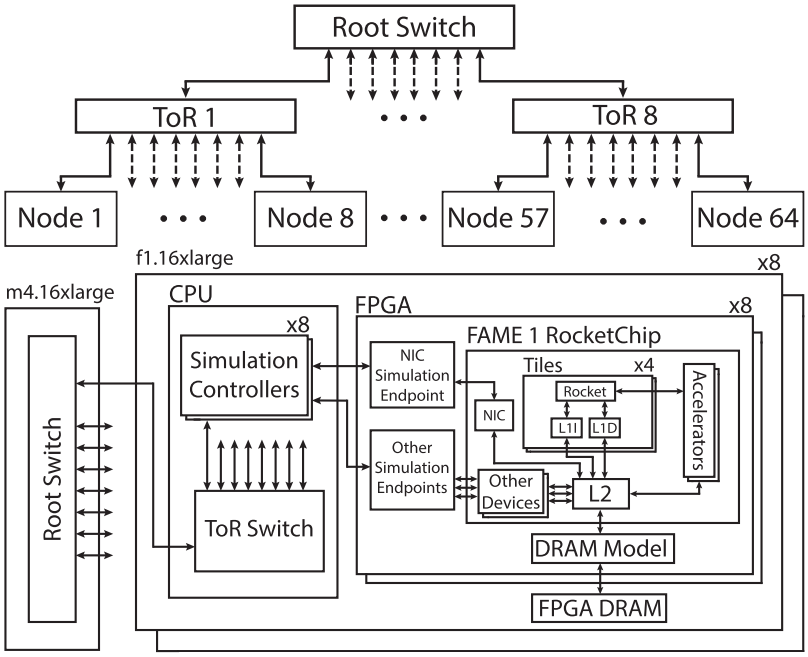

Figure 1. Target view (top) and mapping (bottom) of a 64-node simulation to EC2 F1 in FireSim.

Server Blade Simulation

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 article all 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-Gb/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]

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 article 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.

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 (see the "Thousand-Node Datacenter Simulation" section), we pack four quad-core server blade simulations onto each FPGA, allowing us to model a 32-node rack on each `f1.16xlarge` instance.

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.

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-Gb/s links we use throughout this article, 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.

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. 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 (http://docs.fires.im/en/latest/Advanced-Usage/Workloads/ISCA-2018-Experiments.html), among others.

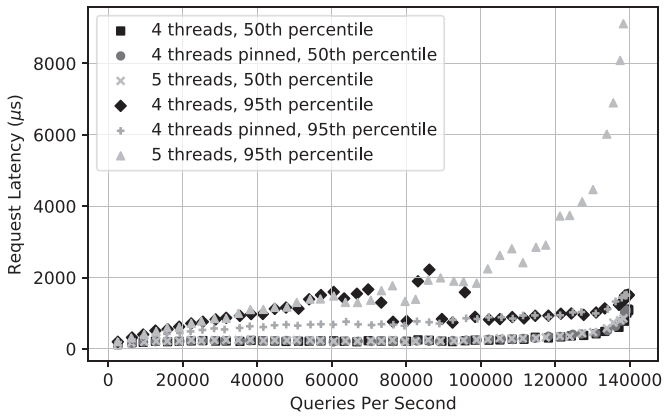

Figure 2. Reproducing the effect of thread imbalance on tail latency in memcached.

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 eight-node cluster in FireSim interconnected by a 200-Gb/s, 2-μs latency network, where each simulated server has four 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 four or five 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 five threads—the tail latency is significantly worsened by the presence of the extra thread, while median latency is essentially unaffected. Our full paper[5] describes several other interesting phenomena shown in Figure 2.

SIMULATION PERFORMANCE

Performance versus 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-Gb/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[5] 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).

Thousand-Node Datacenter Simulation

To demonstrate the scale achievable with FireSim, we run a simulation that models 1024-× 3.2-GHz quad-core nodes, with 32 ToR switches, four aggregation switches, and one root switch, all interconnected by a 2-μs, 200-Gb/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 the "FireSim" section, 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

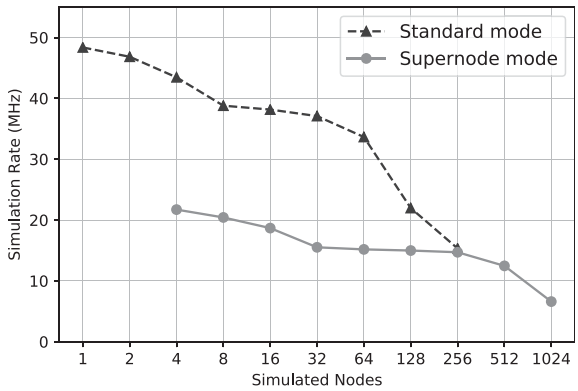

Figure 3. Simulation rate versus the number of simulated target nodes.

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 four downlinks to the four aggregation switches in the target topology. This topology is specified to the FireSim simulation manager with around ten lines of configuration code. More complicated topologies, such as 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 five 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

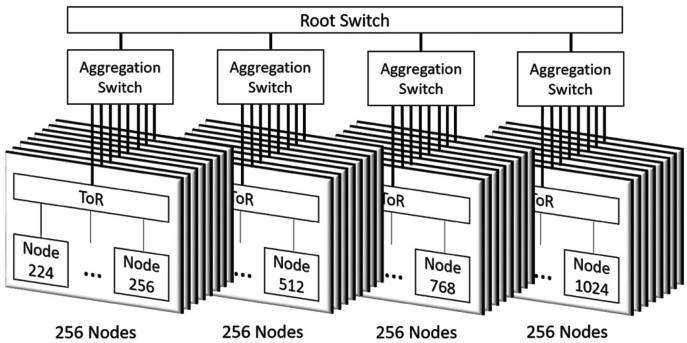

Figure 4. Topology of 1024-node datacenter simulation.

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 (≈ $50 000 each), this simulation harnesses ≈ $12.8 million 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.

RELATED WORK

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

Software Simulators

One approach to simulating WSCs 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.

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 toward fast, productive FPGA-based evaluation for multicore 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 versus 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 tape-out-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 versus Commodity Host Platform. DIABLO used a custom-built FPGA platform that cost ≈ $100000 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.

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 reuse 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.

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.

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 article, 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.

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* (4096 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.

ACKNOWLEDGEMENTS

This work was supported 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 Comput. Archit.*, vol. 39, no. 3, pp. i–189, 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 *Proc. 19th Int. Conf. Archit. Support Program. Lang. Operating Syst.*, 2014, pp. 3–18.
3. A. Mohammad, U. Darbaz, G. Dozsa, S. Diestelhorst, D. Kim, and N. S. Kim, "dist-gem5: Distributed simulation of computer clusters," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, Apr. 2017, pp. 153–162.

4. Z. Tan, Z. Qian, X. Chen, K. Asanović, and D. Patterson, "DIABLO: A warehouse-scale computer network simulator using FPGAs," in *Proc. 20th Int. Conf. Archit. Support Program. Lang. Operating Syst.*, 2015, pp. 207–221.
5. S. Karandikar *et al.*, "FireSim: FPGA-accelerated cycle-exact scale-out system simulation in the public cloud," in *Proc. 45th Annu. Int. Symp. Comput. Archit.*, 2018, pp. 29–42. [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 *Proc. 37th Annu. Int. Symp. Comput. Archit.*, 2010, pp. 290–301.
7. K. Asanović *et al.*, "The rocket chip generator," Electr. Eng. Comput. Sci. Dept., Univ. California, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2016-17, Apr. 2016.
8. D. Biancolin *et al.*, "FASED: FPGA-accelerated simulation and evaluation of DRAM," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, 2019, pp. 330–339. [Online]. Available: https://doi.acm.org/10.1145/3289602.3293894
9. D. Kim *et al.*, "Strober: Fast and accurate sample-based energy simulation for arbit accurate sample-based energy simulation for arbit," in *Proc. 43rd Int. Symp. Comput. Archit.*, 2016, pp. 128–139.
10. J. Leverich and C. Kozyrakis, "Reconciling high server utilization and sub-millisecond quality-of-service," in *Proc. 9th Eur. Conf. Comput. Syst.*, 2014, pp. 4:1–4:14.
11. A. Gutierrez *et al.*, "Sources of error in full-system simulation," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, 2014, pp. 13–22.
12. J. Wawrzynek *et al.*, "RAMP: Research accelerator for multiple processors," *IEEE Micro*, vol. 27, no. 2, pp. 46–57, Mar. 2007.

Sagar Karandikar is currently a PhD student in the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. His research focuses on exploring hardware-software co-design in warehouse-scale machines. He has a BS and an MS in electrical engineering and computer sciences from the University of California, Berkeley. He is a member of the the Association for Computing Machinery (ACM) and the IEEE. Contact him at sagark@eecs.berkeley.edu.

Howard Mao is currently a PhD student at the University of California, Berkeley. He is a member of the ADEPT lab and is interested in designing microarchitectures for datacenter systems. Contact him at zhemao@eecs.berkeley.edu.

Donggyu Kim is currently a PhD student at the University of California, Berkeley. He has an MS in computer science from UC Berkeley, and previously studied at Pohang University of Science and Technology. Contact him at dgkim@berkeley.edu.

David Biancolin is currently a PhD student in the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. He has a BASc in engineering science from the University of Toronto. Contact him at biancolin@berkeley.edu.

Alon Amid is currently a PhD student in the Electrical Engineering and Computer Sciences Department, University of California, Berkeley. His current research focus includes parallel and distributed computing, energy-efficient processors and architectures, and hardware-software co-design. He has a BSc in electrical engineering from Technion—Israel Institute of Technology. Contact him at alonamid@eecs.berkeley.edu.

Dayeol Lee is currently a PhD student in the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. He is interested in hardware/system-level security as well as warehouse-scale computer systems. He has a BS and an MS in computer science and engineering from Pohang University of Science and Technology. Contact him at dayeol@berkeley.edu.

Nathan Pemberton is currently a PhD student in the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, studying computer architecture and operating systems for warehouse-scale computers. He has a BS in computer engineering from the University of California Santa Cruz, and an MS in computer science from the University of California, Berkeley. He is a member of the Association for Computing Machinery (ACM). Contact him at nathanp@berkeley.edu.

Emmanuel Amaro is currently a PhD student in computer science at the University of California, Berkeley. His research interests include systems and computer architecture with a focus in computer disaggregation. Contact him at amaro@berkeley.edu.

Colin Schmidt is currently a PhD student at the University of California, Berkeley, where he works on

architecting, implementing, and building software for vector accelerators. He has a BS in electrical and computer engineering and computer science from Cornell University. He is a student member of the Association for Computing Machinery (ACM). Contact him at colins@berkeley.edu.

Aditya Chopra is currently a software engineer at Google, where he works on the Dynamic Search Ads team. He has an MS from the University of California, Berkeley, where his primary research interest is in low latency serving systems. Contact him at adichopra@google.com.

Qijing Huang is currently a PhD student in the Department of Computer Sciences, University of California, Berkeley. Her current research interests focus on hardware accelerators, high-level synthesis, and machine learning for hardware design. She has a BS from the University of Toronto. Contact her at qijing.huang@berkeley.edu.

Kyle Kovacs is currently a Master's student in electrical engineering and computer sciences at the University of California Berkeley, where he completed his undergraduate degree as well. His academic interests include computer architecture and embedded systems. Contact him at kylekovacs@berkeley.edu.

Borivoje Nikolić is the National Semiconductor Distinguished Professor of Engineering at the University of California, Berkeley. He has a PhD in electrical and computer engineering from the University of California, Davis. He is a Fellow of the IEEE. Contact him at bora@eecs.berkeley.edu.

Randy Howard Katz is currently a United Microelectronics Corporation Distinguished Professor in Electrical Engineering and Computer Science at the University of California, Berkeley. He has published more than 300 refereed technical papers, book chapters, and books. He has supervised 59 MS theses and 48 PhD dissertations. He has received 16 best paper awards, including the triple "test of time" RAID paper and one paper selected for a 50 year retrospective on IEEE Communications publications), and three best presentation awards. In the late 1980s, with colleagues at Berkeley, he developed Redundant Arrays of Inexpensive Disks (RAID), a $15 billion per year industry sector. His current research interests include data analytics, computational mobility, and cloud memory systems. He has an undergraduate degree from Cornell University and an MS and a PhD from the University of California, Berkeley. He joined the Berkeley faculty in 1983. He is a Fellow of the Association for Computing Machinery (ACM), the IEEE, and the American Association for the Advancement of Science, and a member of the National Academy of Engineering and the American Academy of Arts and Sciences. Contact him at randykatz@berkeley.edu.

Jonathan Bachrach is currently an adjunct assistant professor in the Department of Electrical Engineering and Computer Sciences at the University of California, Berkeley. He has a PhD in computer science from the University of Massachusetts, Amherst. Contact him at jrb@berkeley.edu.

Krste Asanović is currently a professor in the Department of Electrical Engineering and Computer Sciences at the University of California, Berkeley. He has a PhD in computer science from the University of California, Berkeley. He is a Fellow of the IEEE and the Association for Computing Machinery (ACM). Contact him at krste@berkeley.edu.