



An Open-Source Platform
for Scalable FPGA-
Accelerated Hardware
Simulation in the Cloud
(and now, local FPGAs too!)

<https://fires.im>

 @firesimproject

Speaker: Sagar Karandikar



Berkeley Architecture Research



The architect/chip-developer's design flow

1. High-level Simulation
2. Write RTL + Software, plug into your favorite ecosystem (e.g. Chipyard)
3. Co-design in software RTL sim (e.g. Verilator, VCS, etc.)
 - Run microbenchmarks
4. Co-design in FPGA-accelerated simulation
 - Boot an OS and run the complete software stack,
obtain realistic performance measurements
5. Tapeout → Chip
 - Boot OS and run applications, but no more opportunity for co-design





The architect/chip-developer's design flow

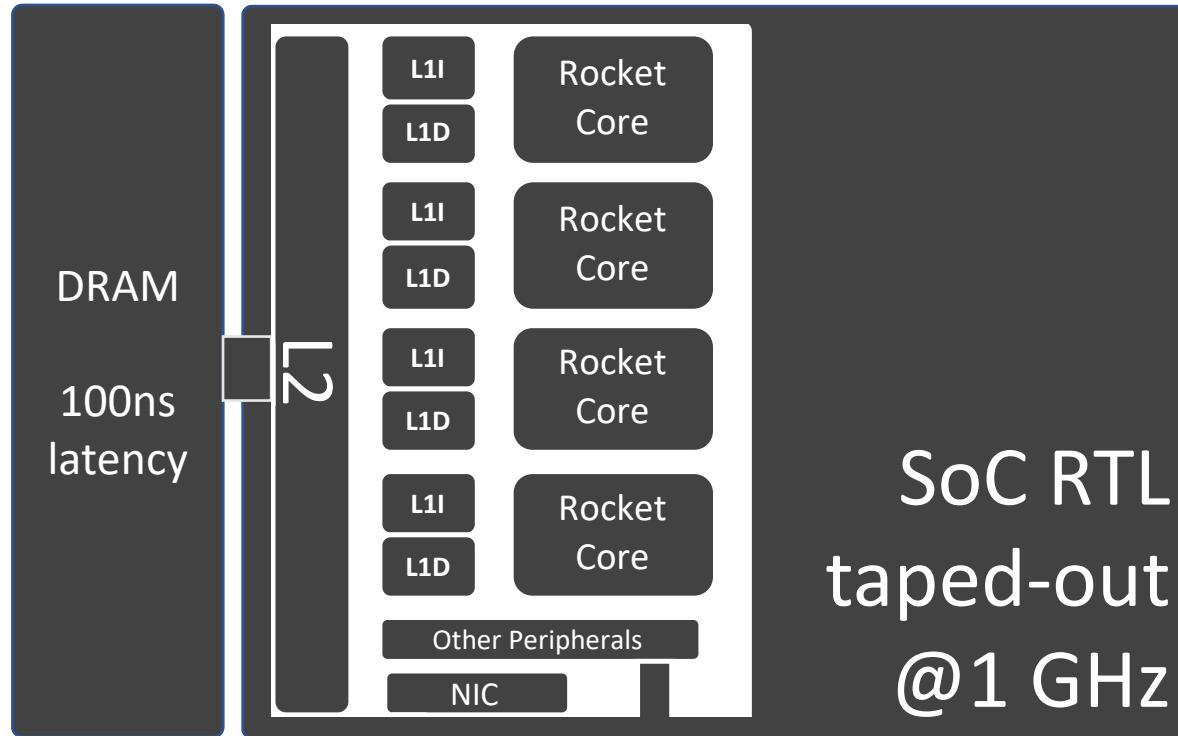
1. High-level Simulation
2. Write RTL + Software, plug into your favorite ecosystem (e.g. Chipyard)
3. Co-design in software RTL sim (e.g. Verilator, VCS, etc.)
 - Run microbenchmarks
4. **Co-design in FPGA-accelerated simulation**
 - Boot an OS and run the complete software stack, obtain realistic performance measurements
5. Tapeout → Chip
 - Boot OS and run applications, but no more opportunity for co-design





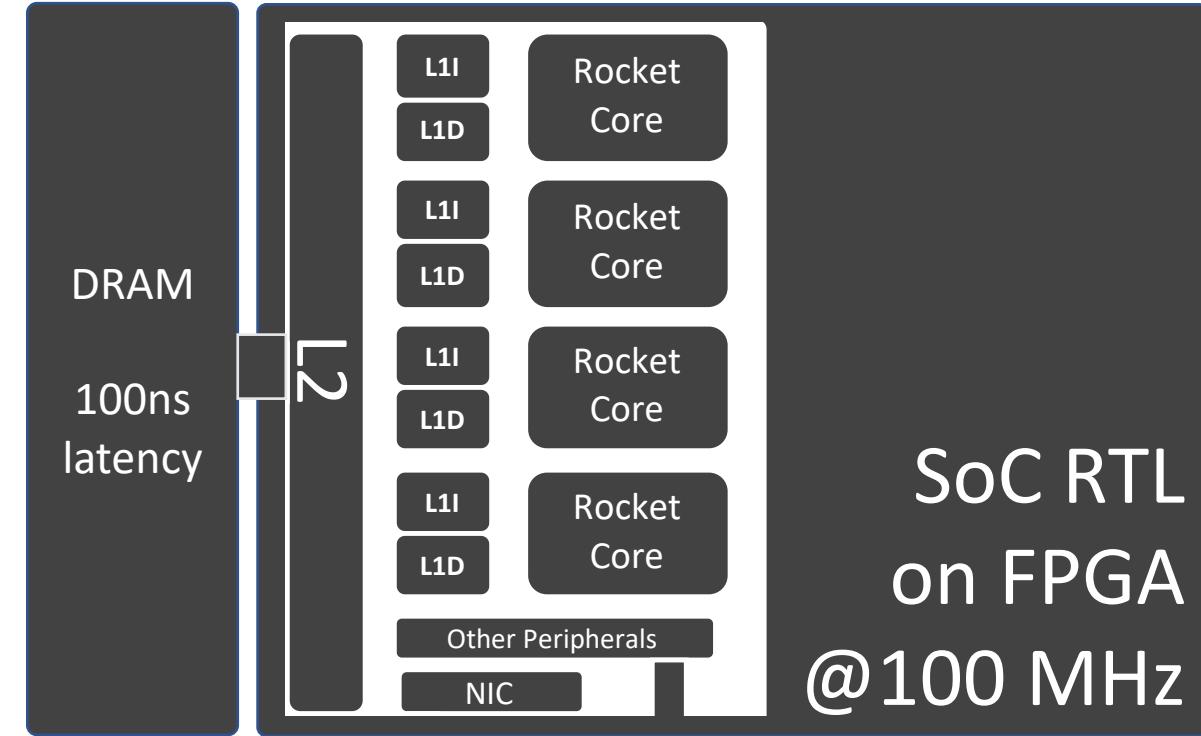
What about FPGA prototyping?

Taped-out SoC



SoC sees 100 cycle DRAM latency

FPGA Prototype of SoC



SoC sees 10 cycle DRAM latency





The Difficulty with FPGA Prototypes

- Every FPGA clock executes one cycle of the simulated machine
- Exposes latencies of FPGA resources to the simulated world.

Three problems:

- 1) FPGA resources may not be an accurate model (ex. previous slide)
- 2) Simulations are non-deterministic
- 3) Different host FPGAs produce different simulation results





Want HW simulators that:

- Are as fast as silicon
- Are as detailed as silicon
- Have all the benefits of SW-based simulators
- Are low-cost

Our Thesis:

- FPGAs are the only viable basis technology
→ Build *FPGA-accelerated* simulators with
SW-like flexibility using an *open-source* tool





How? Useful Trends Throughout the Stack

Open ISA



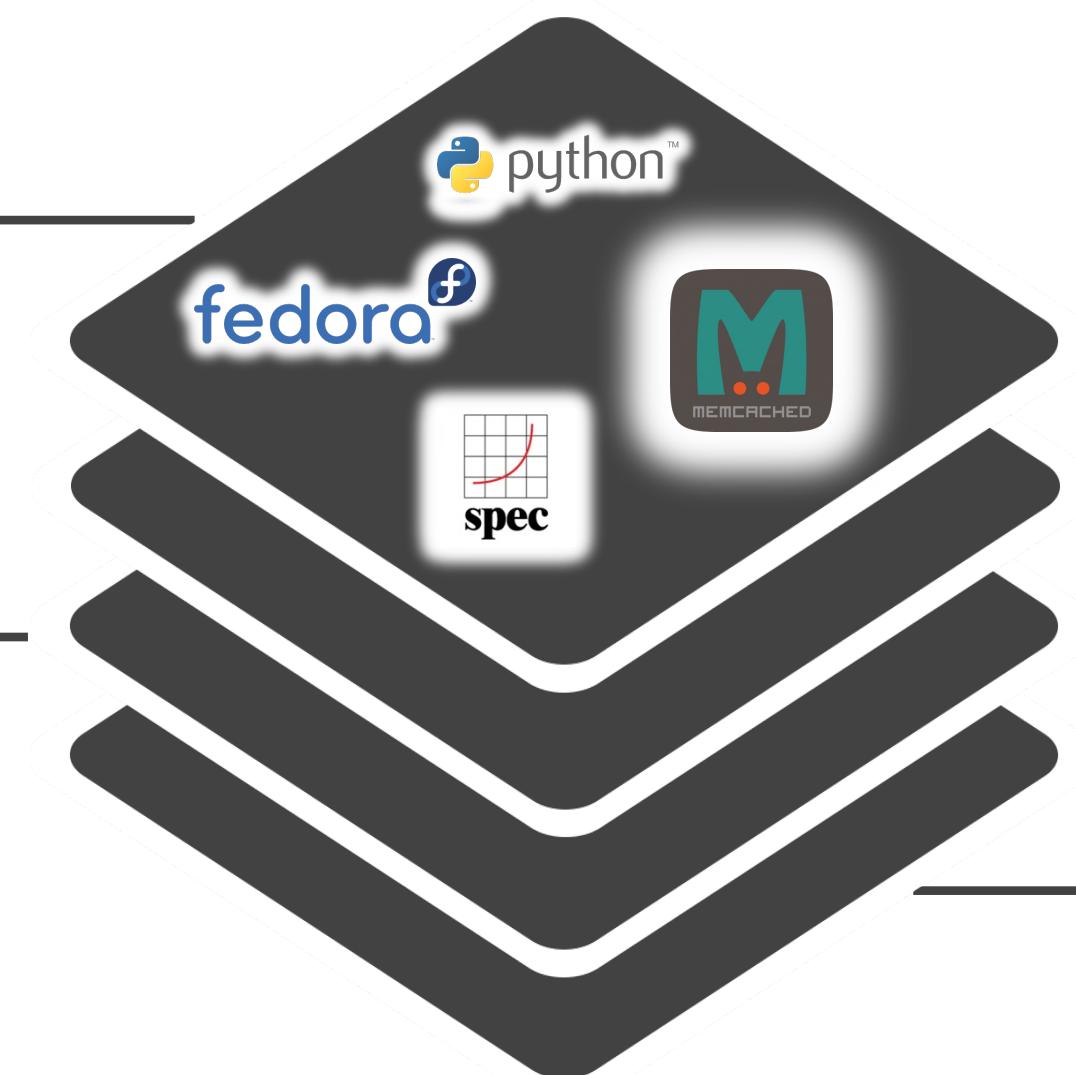
CHISEL

High-Productivity
Hardware Design

Language & IR



Berkeley Architecture Research



Open, Silicon-Proven
SoC Implementations



FPGAs in the Cloud



Amazon EC2 F1 Instances

Run Customizable FPGAs in the AWS Cloud



FireSim at 35,000 feet

- Open-source, fast, automatic, deterministic FPGA-accelerated hardware simulation for pre-silicon verification and performance validation
- Ingests:
 - Your RTL design: FIRRTL (Chisel or Verilog via Yosys), blackbox Verilog
 - Or Chipyard-generated designs with Rocket Chip, BOOM, NVDLA, PicoRV32, and more
 - HW and/or SW IO models (e.g. UART, Ethernet, DRAM, etc.)
 - Workload descriptions
- Produces:
 - Fast, cycle-exact simulation of your design + models around it
 - Automatically deployed to cloud FPGAs (AWS EC2 F1)
 - And now, local FPGAs too (e.g. Xilinx Alveo U250)





Three Distinguishing Features of FireSim

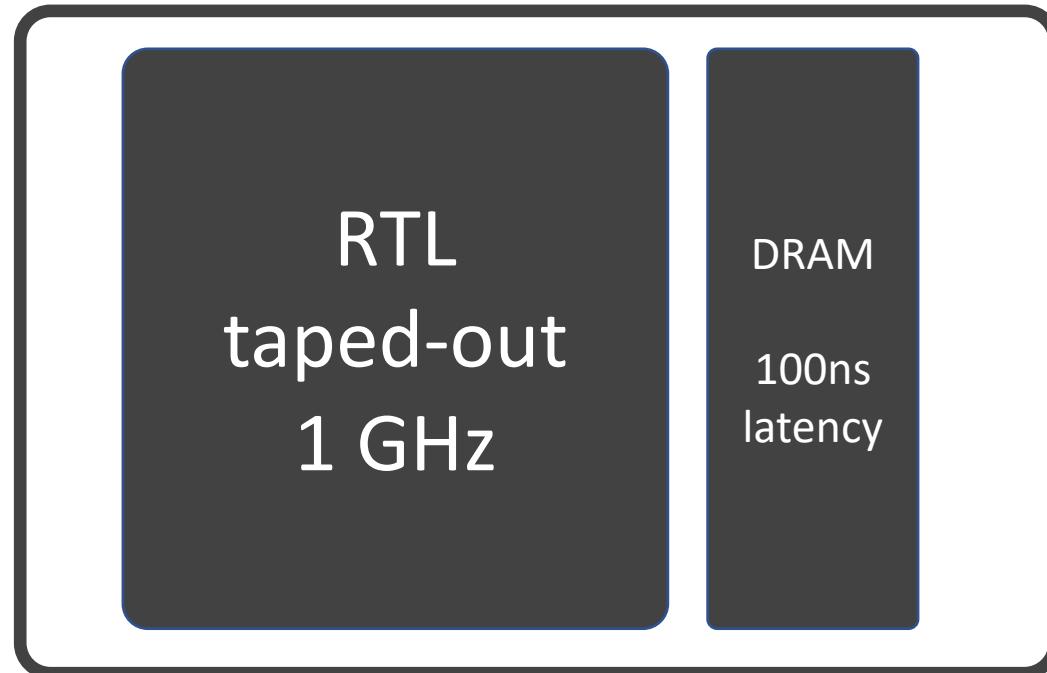
- 1) Not FPGA prototypes, rather FPGA-accelerated simulators
 - Automatic transformation of designs into FPGA-accelerated simulators
 - Enables new debugging, resource optimization, and profiling capabilities
- 2) Uses cloud FPGAs
 - Inexpensive, elastic supply of large FPGAs
 - Easy to collaborate with other researchers
 - Heavy automation to hide FPGA complexity
- 3) Open-source (<https://fires.im>)





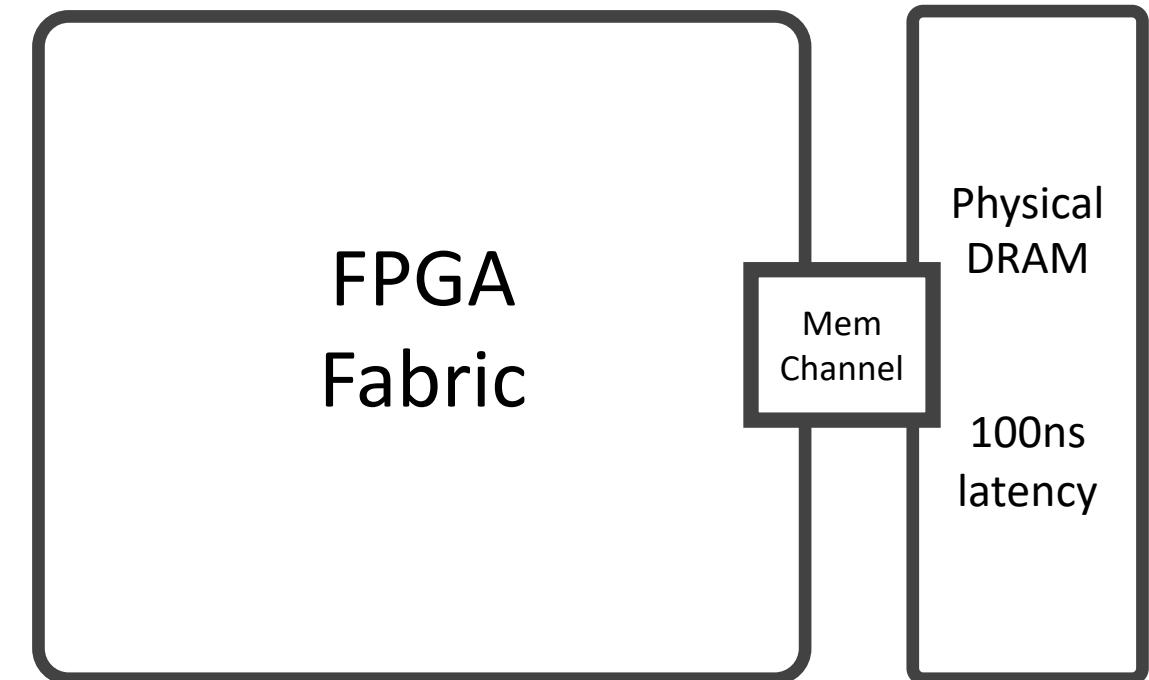
Separating Target and Host

Target: the machine under simulation



Closed simulation world.

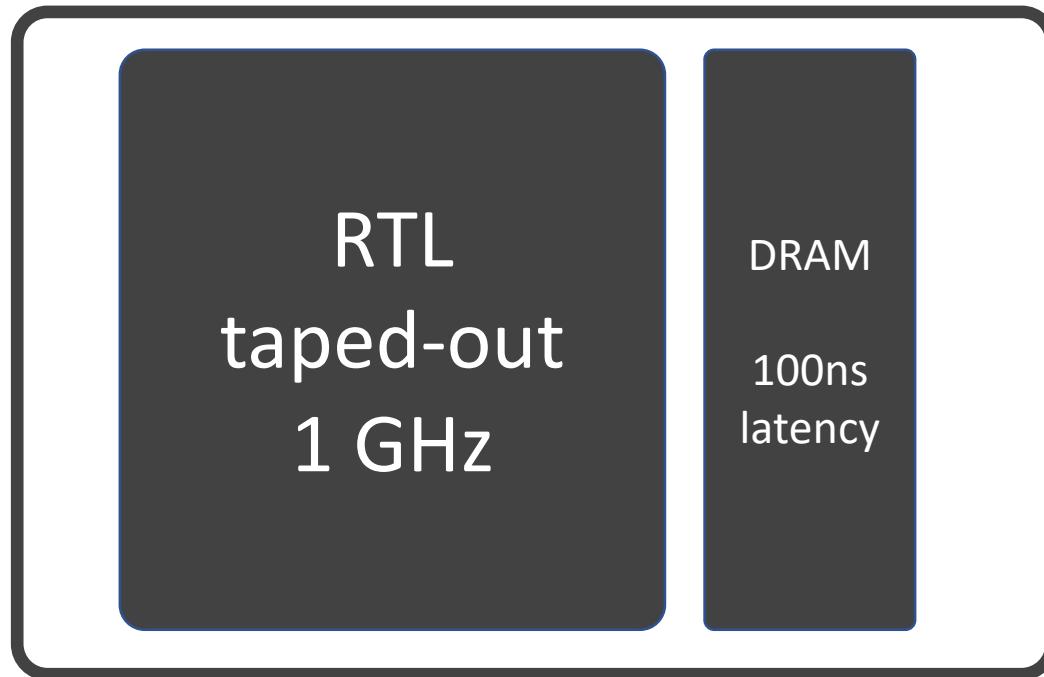
Host: the machine executing (*hosting*) the simulation





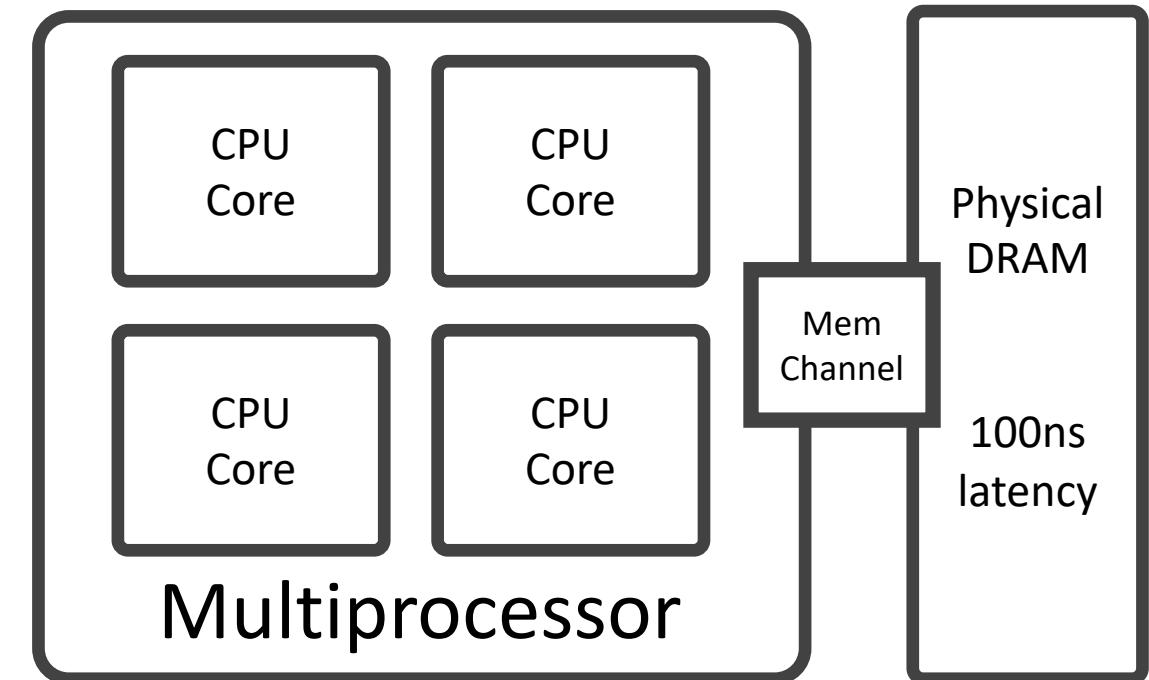
Separating Target and Host

Target: the machine under simulation



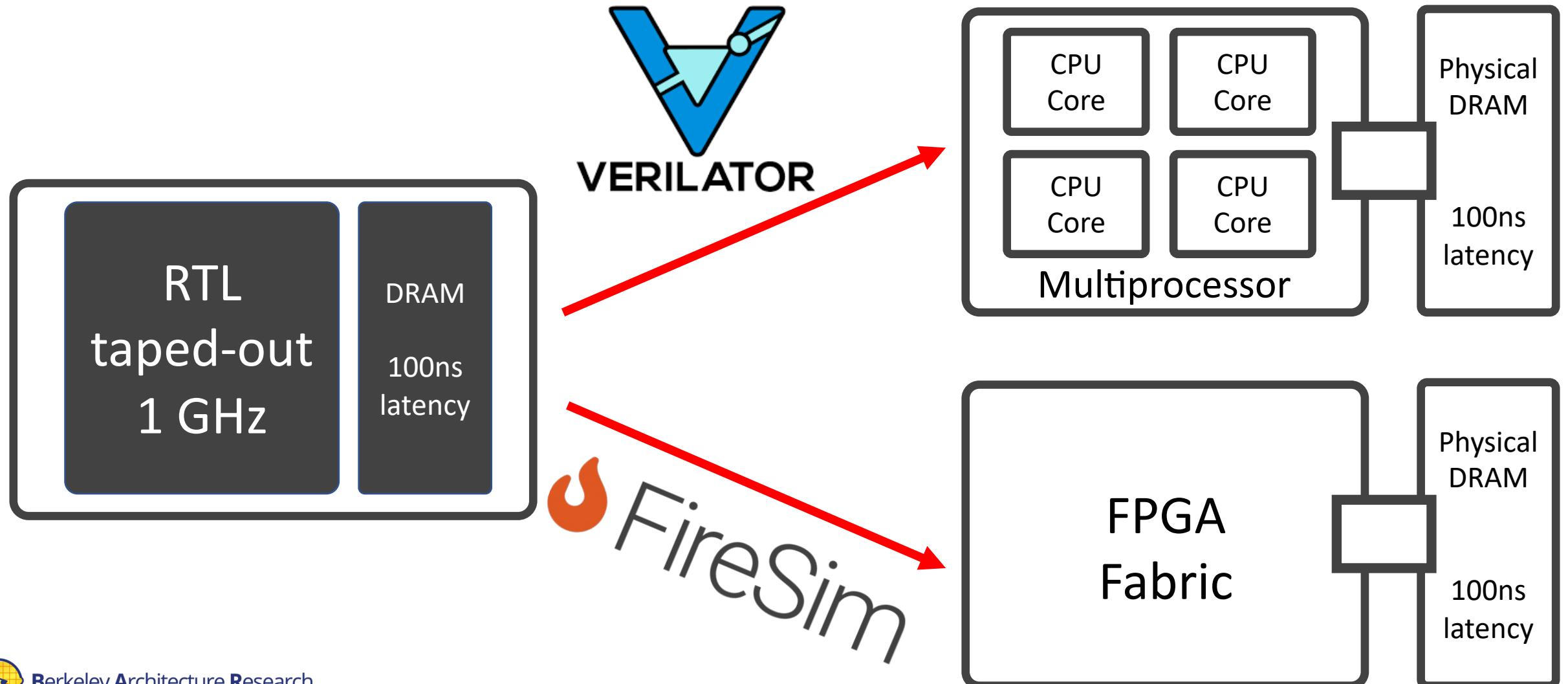
Closed simulation world.

Host: the machine executing (*hosting*) the simulation





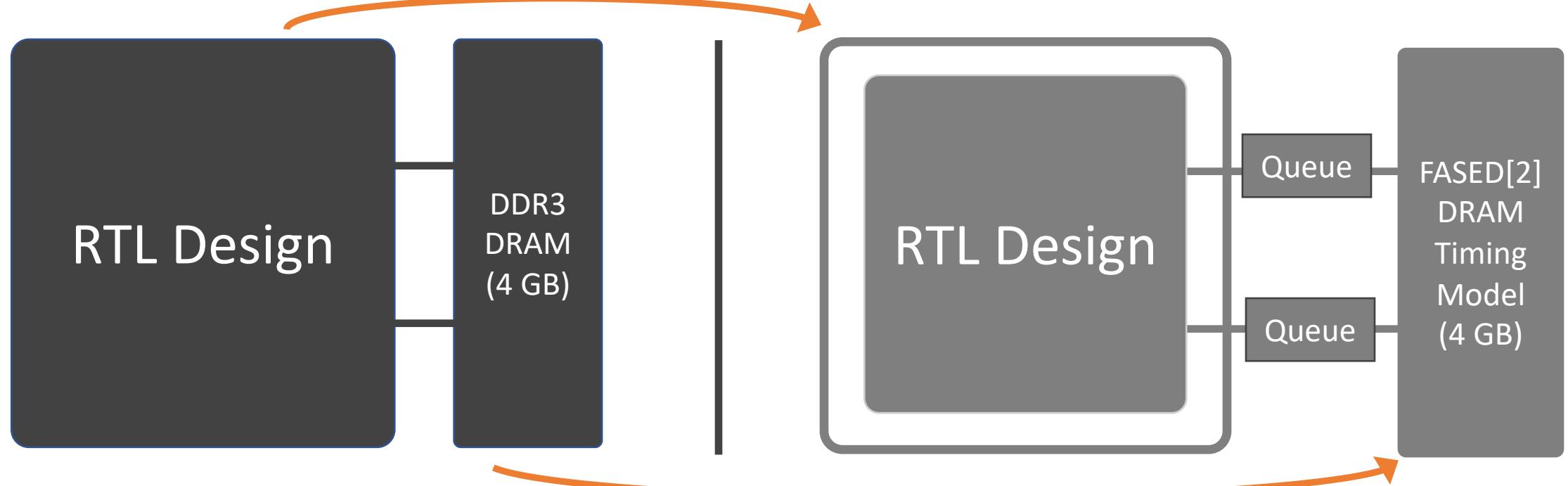
FireSim Generates FPGA-Hosted Simulators





Host Decoupling in FireSim: Transforming the Target

- 1) Convert RTL into a latency-insensitive [1] model using FIRRTL transform



- 2) Generate FPGA-hosted model for DRAM [2] (think DRAMSim on an FPGA)
- 3) Generate queues (token channels) to connect the target models

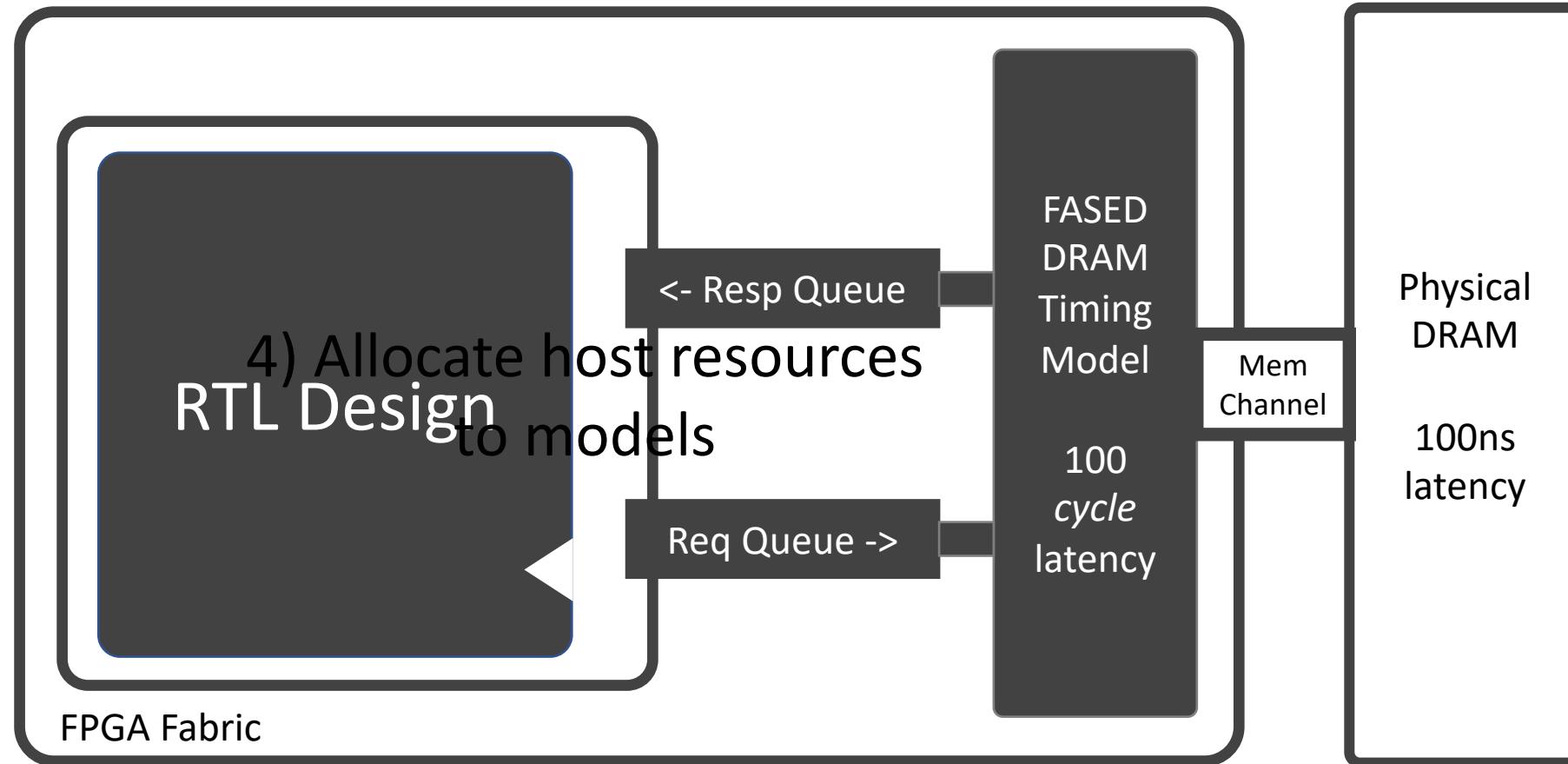
[1] *Theory of Latency Insensitive Design*, Carloni et al, also see: RAMP

[2] FASED: FPGA-accelerated Simulation and Evaluation of DRAM, Biancolin et al





Host Decoupling in FireSim: Mapping to the FPGA



SoC sees realistic DRAM latency





Benefits of Host Decoupling on FPGAs

Simulations:

- Execute deterministically
- Produce identical results on different hosts (FPGAs & CPUs)

This enables support for:

1. SW co-simulation (e.g. block device, network models)
2. Simulating large targets over distributed hosts (ISCA '18, Top Picks '18)
3. Non-invasive debugging and instrumentation (FPL '18, ASPLOS '20)
4. Multi-cycle resource optimizations (ICCAD '19)





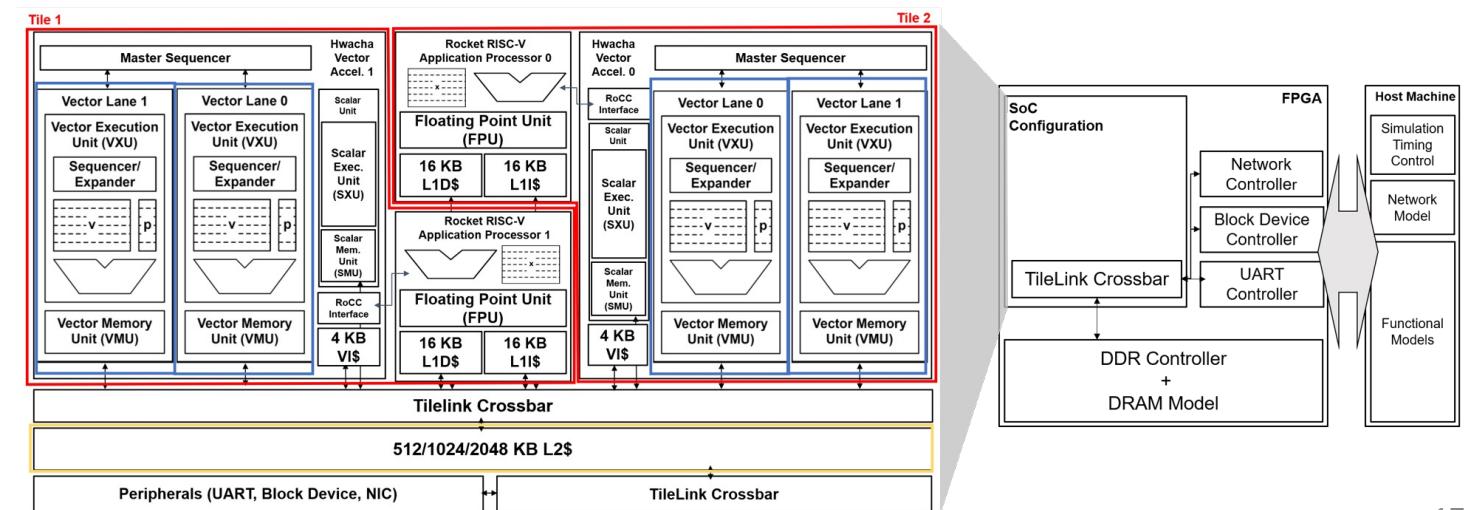
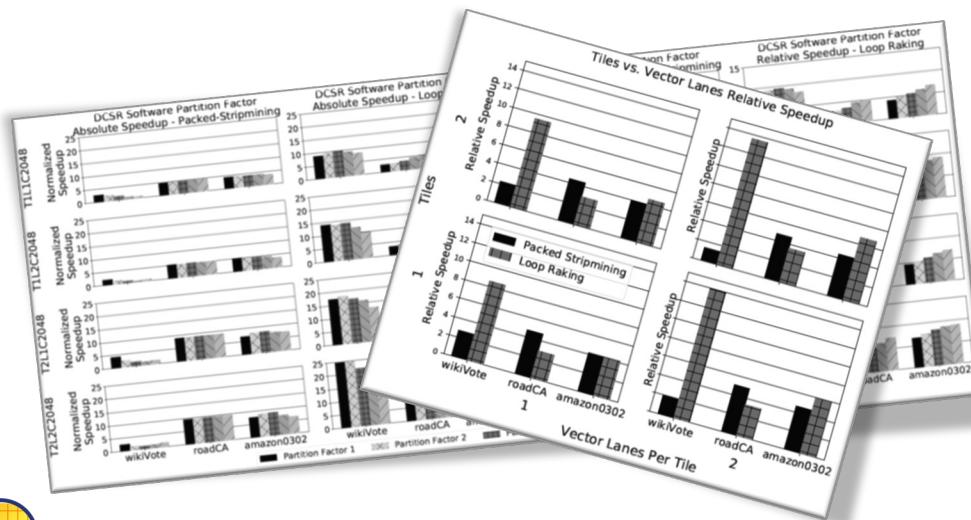
What Can You Do With FireSim?





Example use cases: Evaluating SoC Designs

- Performance Measurement
 - Run SPECint 2017 with reference inputs on Rocket Chip in parallel on ~10 FPGAs within a day (e.g., in D. Biancolin, et. al., *FASED*, FPGA '19)
- Rapid Full-System Design Space Exploration
 - Data-parallel accelerators (Hwacha) and multi-core processors
 - Complex software stacks (Linux, OpenMP, GraphMat, Caffe)





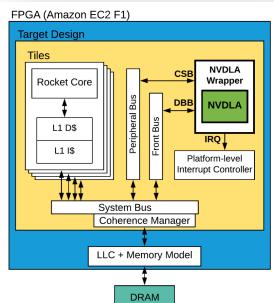
Example use cases: Evaluating SoC Designs

- Security:
 - BOOM Spectre replication
 - A. Gonzalez, et. al., *Replicating and Mitigating Spectre Attacks on an Open Source RISC-V Microarchitecture*, CARRV '19
 - Keystone Enclave performance evaluation
 - D. Lee, et. al., *Keystone*, EuroSys '20
 - Accelerator evaluation
 - Chisel-based accelerators:
 - Machine learning (H. Genc, et. al., *Gemmini*, DAC 2021)
 - Garbage collection (M. Maas, et. al., *A Hardware Accelerator for Tracing Garbage Collection*, ISCA '18)
 - Integrating Verilog-based accelerators:
 - NVDLA (F. Farshchi, et. al. *Integrating NVIDIA Deep Learning Accelerator (NVDLA) with RISC-V SoC on FireSim*. EMC2 '19)
 - HLS-based rapid prototyping (Q. Huang, et. al., *Centrifuge*, ICCAD '19)
 - Scale-out accelerators
 - nanopU NIC-CPU co-design (S. Ibanez, et. al., *nanopU*, OSDI '21)
 - Protobuf Accelerator (S. Karandikar, et. al., *A Hardware Accelerator for Protocol Buffers*, MICRO '21. MICRO-54 Distinguished Artifact Winner.)



Farzad Farshchi
University of Kansas
farshchi@ku.edu

Qijing Huang
University of California
qijing.huang@berkeley.edu



Example use cases: Debugging and Profiling SoC Designs



- Debugging a Chisel design at FPGA-speeds
 - Many FireSim debugging features: Assertion synthesis, printf synthesis, ILA insertion, etc.
 - e.g. FireSim Debugging Docs

The screenshot shows the FireSim documentation website with a blue header featuring the FireSim logo and the word "stable". A search bar is present. The main content area has a dark sidebar with navigation links: "GETTING STARTED:", "AWS EC2 F1 TUTORIAL:", and "ADVANCED DOCS:". The main content area is titled "Debugging and Profiling on the FPGA" and contains several sections with sub-links, such as "Capturing RISC-V Instruction Traces with TracerV", "Assertion Synthesis: Catching RTL Assertions on the FPGA", "Printf Synthesis: Capturing RTL printf Calls when Running on the FPGA", and "AutoILA: Simple Integrated Logic Analyzer (ILA) Insertion".

» Debugging and Profiling on the FPGA

Edit on GitHub

Debugging and Profiling on the FPGA

A common issue with FPGA-prototyping is the difficulty involved in trying to debug and profile systems once they are running on the FPGA. FireSim addresses these issues with a variety of tools for introspecting on designs once you have a *FireSim* simulation running on an FPGA. This section describes these features.

Debugging and Profiling on the FPGA:

- Capturing RISC-V Instruction Traces with TracerV
 - Building a Design with TracerV
 - Enabling Tracing at Runtime
 - Selecting a Trace Output Format
 - Setting a TracerV Trigger
 - Interpreting the Trace Result
 - Caveats
- Assertion Synthesis: Catching RTL Assertions on the FPGA
 - Enabling Assertion Synthesis
 - Runtime Behavior
 - Related Publications
- Printf Synthesis: Capturing RTL printf Calls when Running on the FPGA
 - Enabling Printf Synthesis
 - Runtime Arguments
 - Related Publications
- AutoILA: Simple Integrated Logic Analyzer (ILA) Insertion
 - Enabling AutoILA
 - Annotating Signals
 - Setting a ILA Depth
 - Using the ILA at Runtime
- AutoCounter: Profiling with Out-of-Band Performance Counter Collection
 - Chisel Interface
 - Enabling AutoCounter in Golden Gate
 - Rocket Chip Cover Functions
 - AutoCounter Runtime Parameters
 - AutoCounter CSV Output Format
 - Using TracerV Trigger with AutoCounter



Example use cases: Debugging and Profiling SoC Designs



- Debugging a Chisel design at FPGA-speeds
 - Many FireSim debugging features: Assertion synthesis, printf synthesis, ILA insertion, etc.
 - e.g. FireSim Debugging Docs

The screenshot shows a dark-themed sidebar of a documentation site. At the top, there's a navigation bar with links like "Assertion Synthesis", "printf Synthesis", "AutoILA", "AutoCounter", "TracerV + Flame Graphs", "Dromajo Co-simulation", "Debugging a Hanging Simulator", "Non-Source Dependency Management", "Supernode", "Miscellaneous Tips", "FireSim Asked Questions", and "(Experimental) Using On Premise FPGAs". Below this is a section titled "COMPILER (GOLDEN GATE) DOCS:" with links to "Overview & Philosophy" and "Read the Docs". A dropdown menu indicates the version is "v: stable".

- Assertion Synthesis: Catching RTL Assertions on the FPGA
- printf Synthesis: Capturing RTL printf Calls when Running on the FPGA
 - Enabling printf Synthesis
 - Runtime Arguments
 - Related Publications
- AutoILA: Simple Integrated Logic Analyzer (ILA) Insertion
 - Enabling AutoILA
 - Annotating Signals
 - Setting a ILA Depth
 - Using the ILA at Runtime
- AutoCounter: Profiling with Out-of-Band Performance Counter Collection
 - Chisel Interface
 - Enabling AutoCounter in Golden Gate
 - Rocket Chip Cover Functions
 - AutoCounter Runtime Parameters
 - AutoCounter CSV Output Format
 - Using TracerV Trigger with AutoCounter
 - AutoCounter using Synthesizable Prints
 - Reset & Timing Considerations
- TracerV + Flame Graphs: Profiling Software with Out-of-Band Flame Graph Generation
 - What are Flame Graphs?
 - Prerequisites
 - Enabling Flame Graph generation in `config_runtime.yaml`
 - Producing DWARF information to supply to the TracerV driver
 - Modifying your workload description
 - Running a simulation
 - Caveats
- Dromajo Co-simulation with BOOM designs
 - Building a Design with Dromajo
 - Running a FireSim Simulation
 - Troubleshooting Dromajo Simulations with Meta-Simulations
- Debugging a Hanging Simulator
 - Case 1: Target hang.
 - Case 2: Simulator hang due to FPGA-side token starvation.
 - Case 3: Simulator hang due to driver-side deadlock.
 - Simulator Heartbeat PlusArgs

Previous

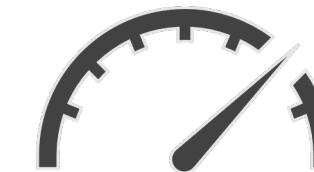
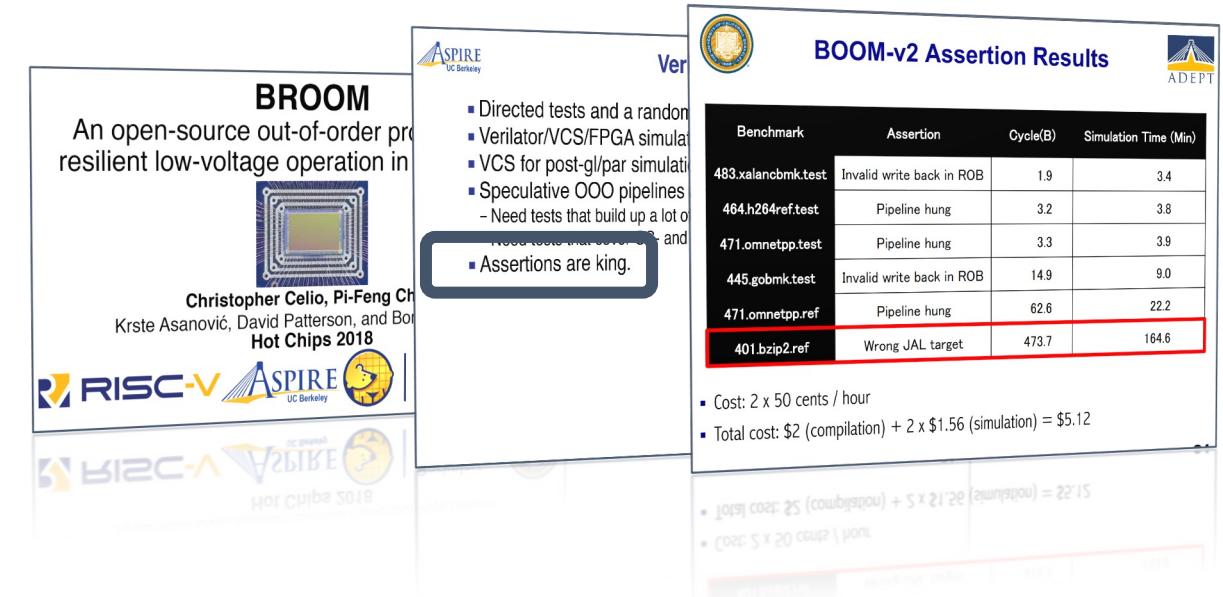
Next





Example use cases: Debugging and Profiling SoC Designs

- Debugging a Chisel design at FPGA-speeds
 - Many FireSim debugging features: Assertion synthesis, printf synthesis, ILA insertion, etc.
 - e.g. FireSim Debugging Docs
 - e.g. Fixing BOOM Bugs (D. Kim, et. al., *DESsert*, FPL '18)
- Profiling a custom RISC-V SoC at FPGA-speeds
 - e.g. HW/SW Co-design of a networked RISC-V system (S. Karandikar, et. al., *FirePerf*, ASPLOS 2020)



FirePerf





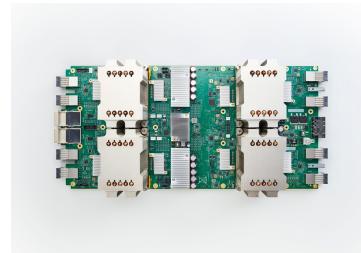
How-to-build a *datacenter-scale* FireSim simulation

- [1] S. Karandikar et. al., "FireSim: FPGA-Accelerated Cycle-Exact Scale-Out System Simulation in the Public Cloud." *ISCA 2018*
- [2] S. Karandikar et. al., "FireSim: FPGA-Accelerated Cycle-Exact Scale-Out System Simulation in the Public Cloud." *IEEE Micro Top Picks 2018*



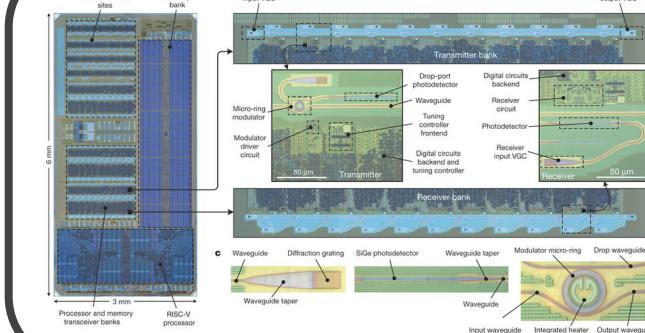


The new datacenter hardware environment



**The end of
Moore's Law**

Custom Silicon
in the Cloud

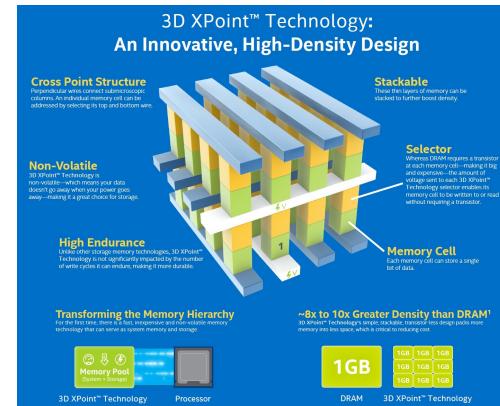


**Faster
networks**

e.g. Silicon
Photonics

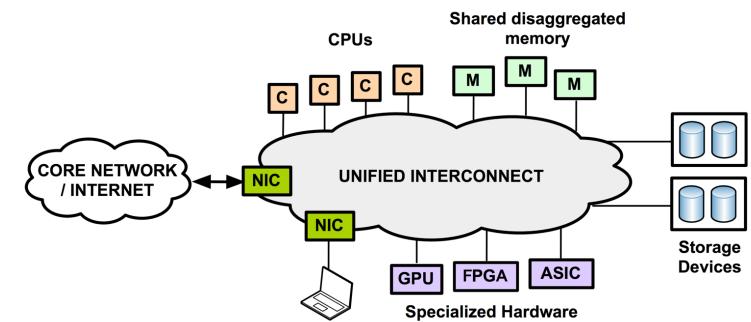
**Deeper
memory/storage
hierarchies**

e.g. 3DXPoint, HBM



**New datacenter
architectures**

e.g. disaggregation



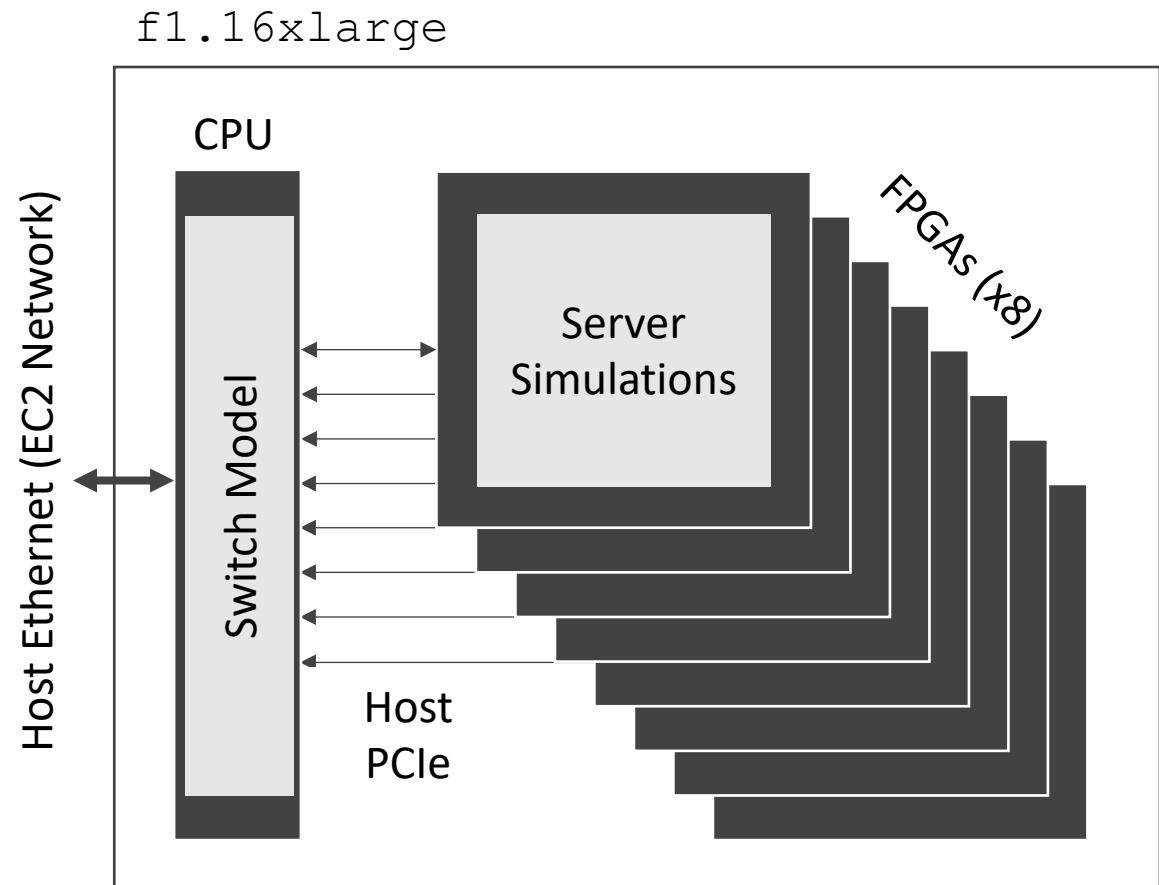
[1]





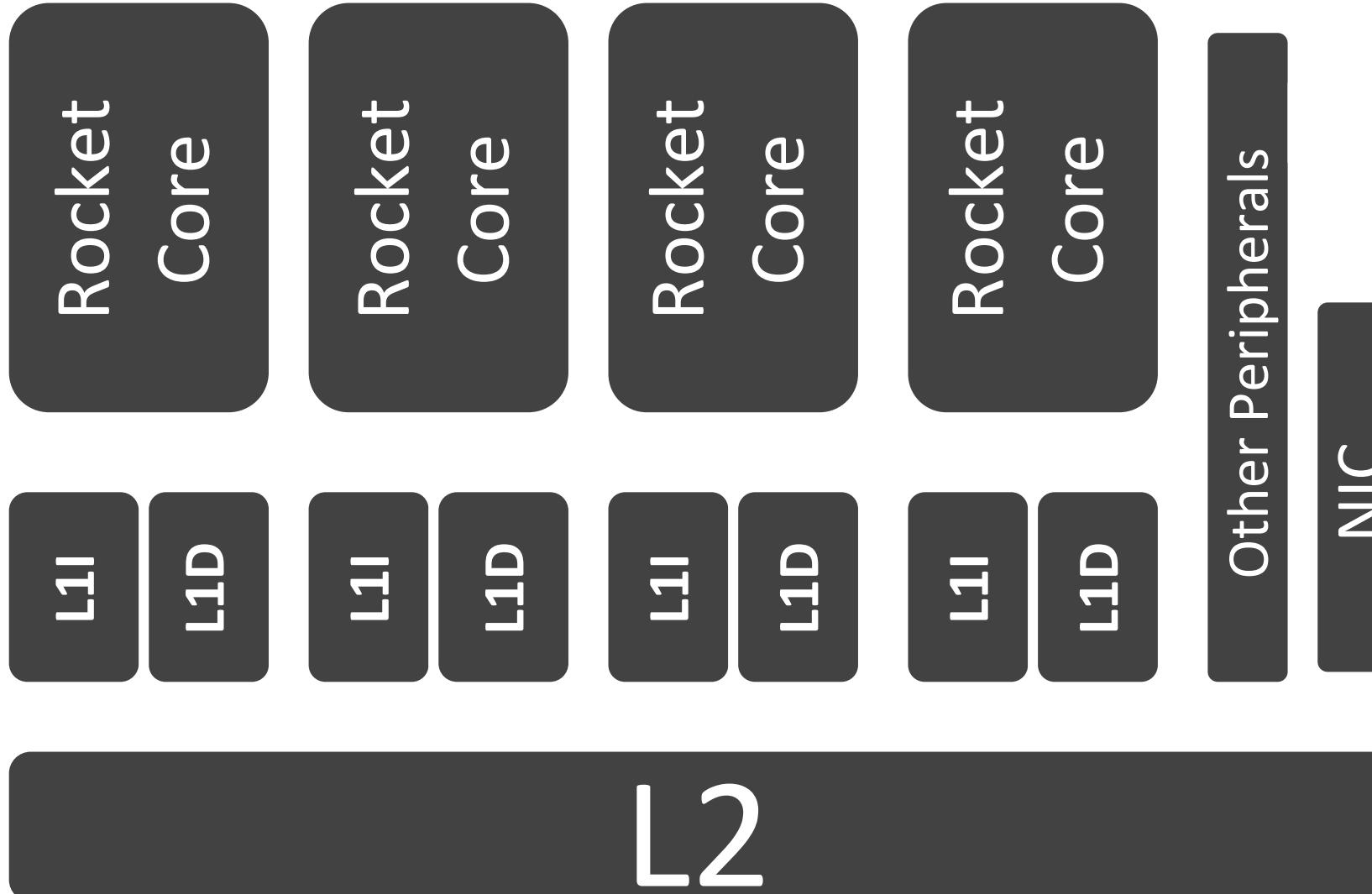
Mapping a datacenter simulation

- DC simulation requires:
 - Model hardware at scale, cycle-accurately
 - Run real software
- RTL and abstract SW model co-simulation
- Server Simulations
 - Good fit for the FPGA
 - We have tapeout-proven RTL: FAME-1 transform w/Golden-Gate
- Network simulation
 - Little parallelism in switch models (e.g. a thread per port)
 - Need to coordinate all the distributed server simulations
 - So use CPUs + host network





Step 1: Server SoC in RTL



Modeled System

- 4x RISC-V Rocket Cores @ 3.2 GHz
- 16K I/D L1\$
- 256K Shared L2\$
- 200 Gb/s Eth. NIC

Resource Util.

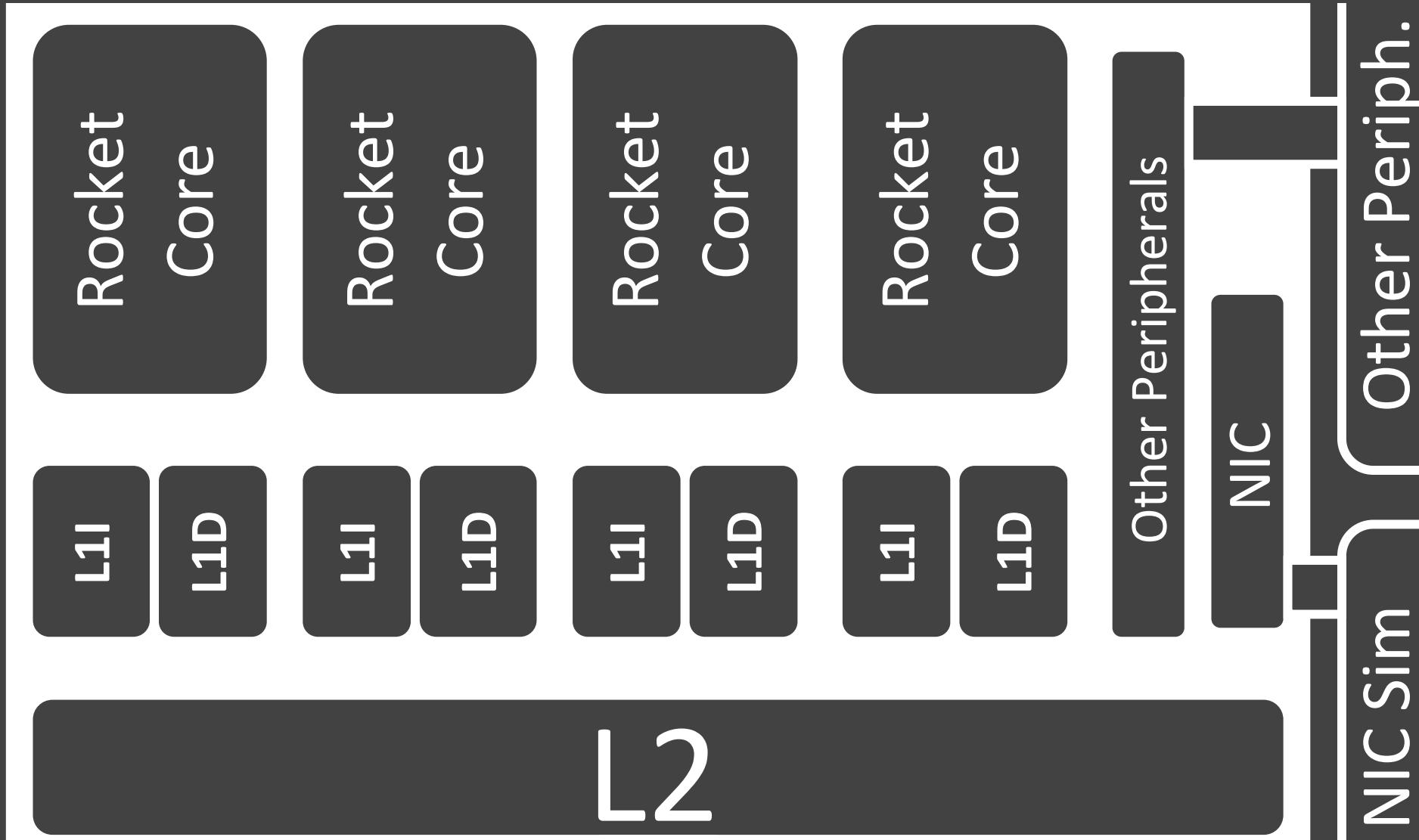
- < ¼ of an FPGA

Sim Rate

- N/A



Step 1: Server SoC in RTL



Modeled System

- 4x RISC-V Rocket Cores @ 3.2 GHz
- 16K I/D L1\$
- 256K Shared L2\$
- 200 Gb/s Eth. NIC

Resource Util.

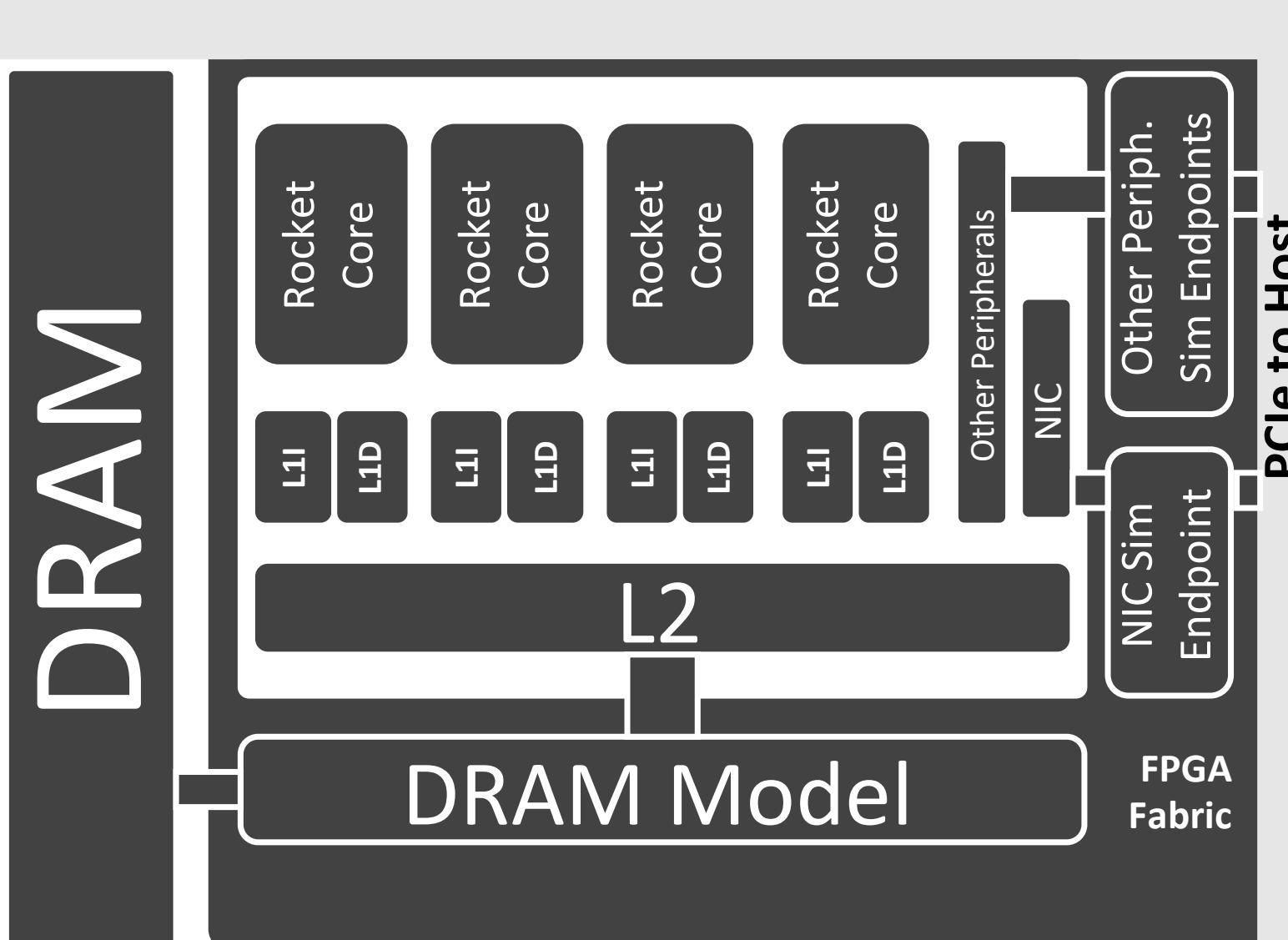
- < $\frac{1}{4}$ of an FPGA

Sim Rate

- N/A



Step 2: FPGA Simulation of one server blade



Modeled System

- 4x RISC-V Rocket Cores @ 3.2 GHz
- 16K I/D L1\$
- 256K Shared L2\$
- 200 Gb/s Eth. NIC
- 16 GB DDR3

Resource Util.

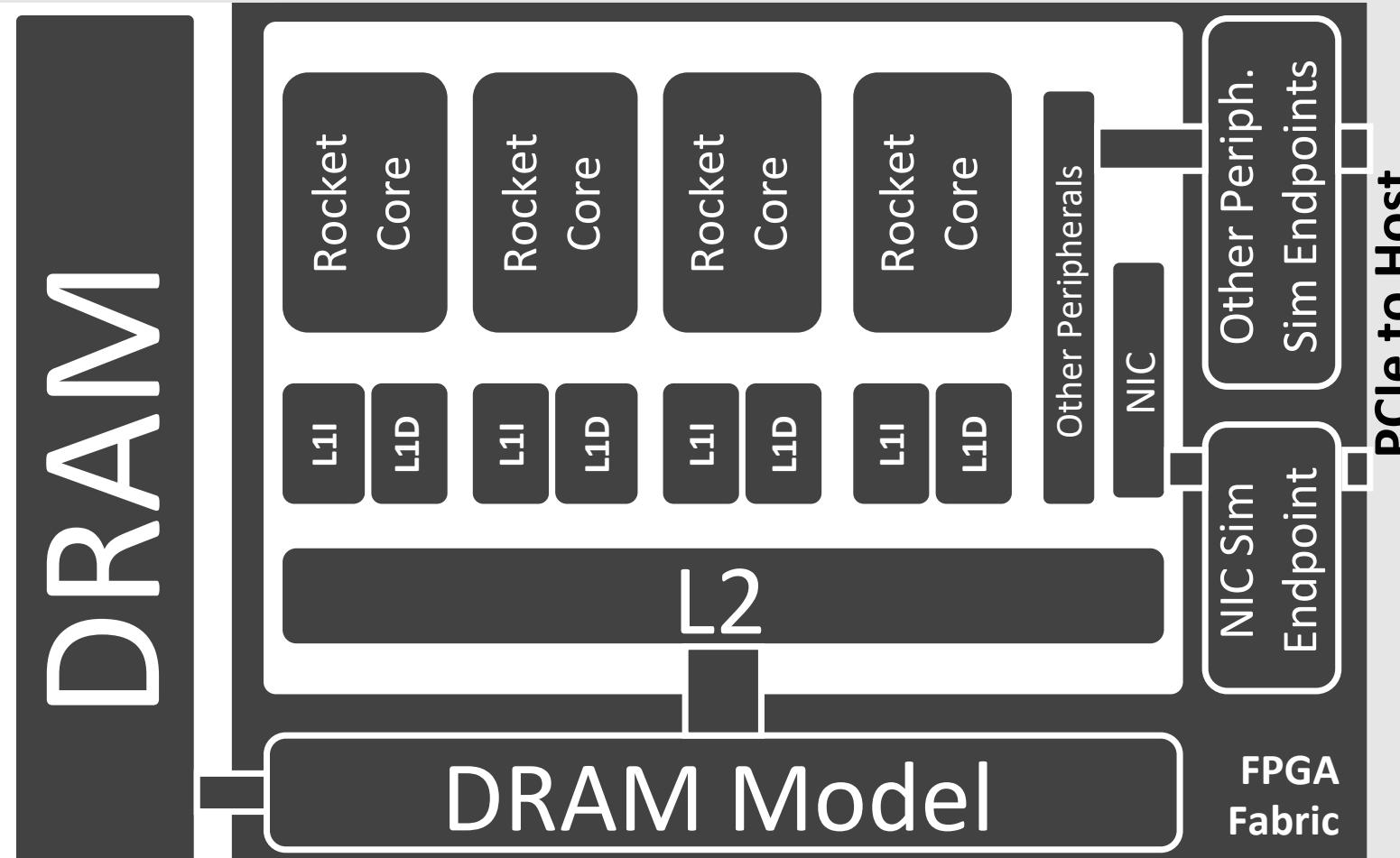
- < $\frac{1}{4}$ of an FPGA
- $\frac{1}{4}$ Mem Chans

Sim Rate

- ~150 MHz
- ~40 MHz (netw)



Step 2: FPGA Simulation of one server blade



Modeled System

- 4x RISC-V Rocket Cores @ 3.2 GHz
- 16K I/D L1\$
- 256K Shared L2\$
- 200 Gb/s Eth. NIC

Resource Util.

- < $\frac{1}{4}$ of an FPGA
- $\frac{1}{4}$ Mem Chans

Sim Rate

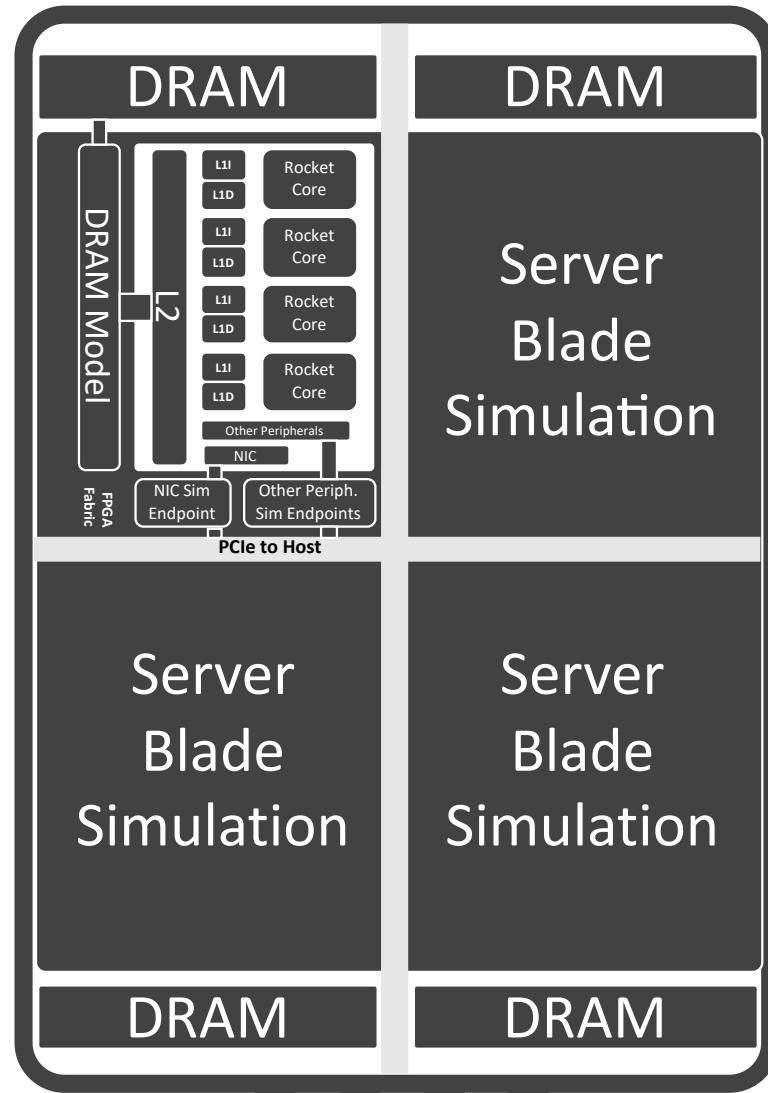
- ~150 MHz
- ~40 MHz (netw)



Step 3: FPGA Simulation of 4 server blades

Cost:
\$0.49 per hour
(spot)

\$1.65 per hour
(on-demand)



Modeled System

- 4 Server Blades
- 16 Cores
- 64 GB DDR3

Resource Util.

- < 1 FPGA
- 4/4 Mem Chans

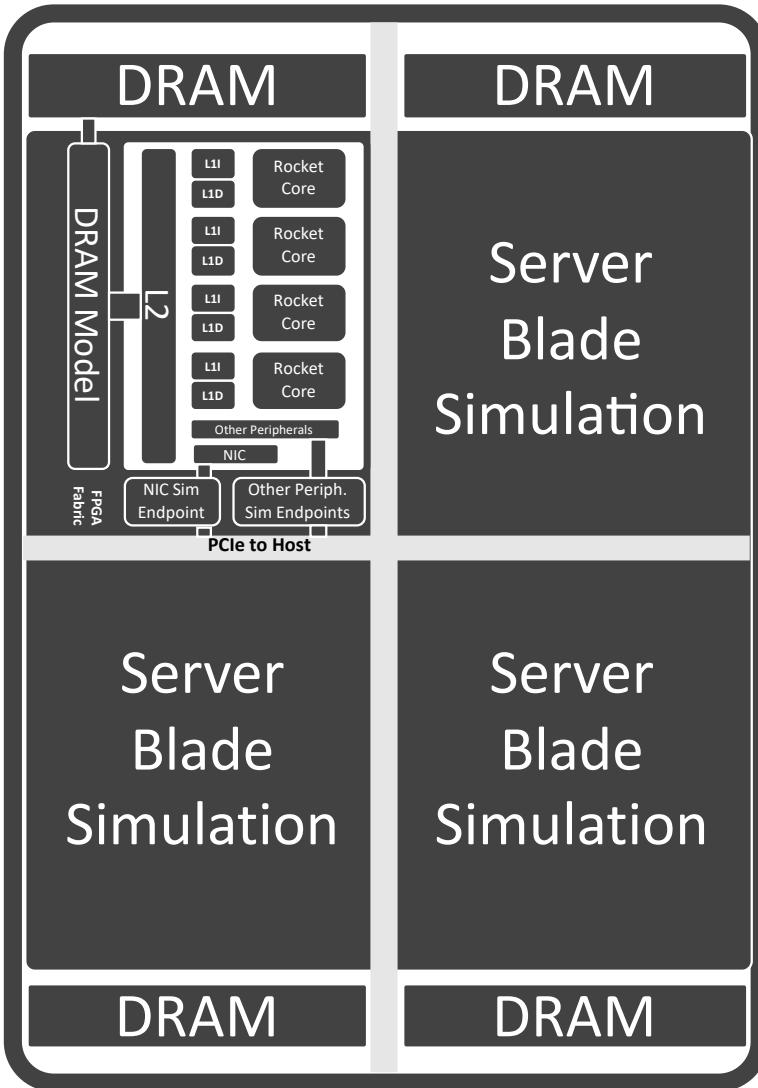
Sim Rate

- ~14.3 MHz
(netw)



Step 3: FPGA Simulation of 4 server blades

FPGA
4 Sims)



FPGA
(4 Sims)

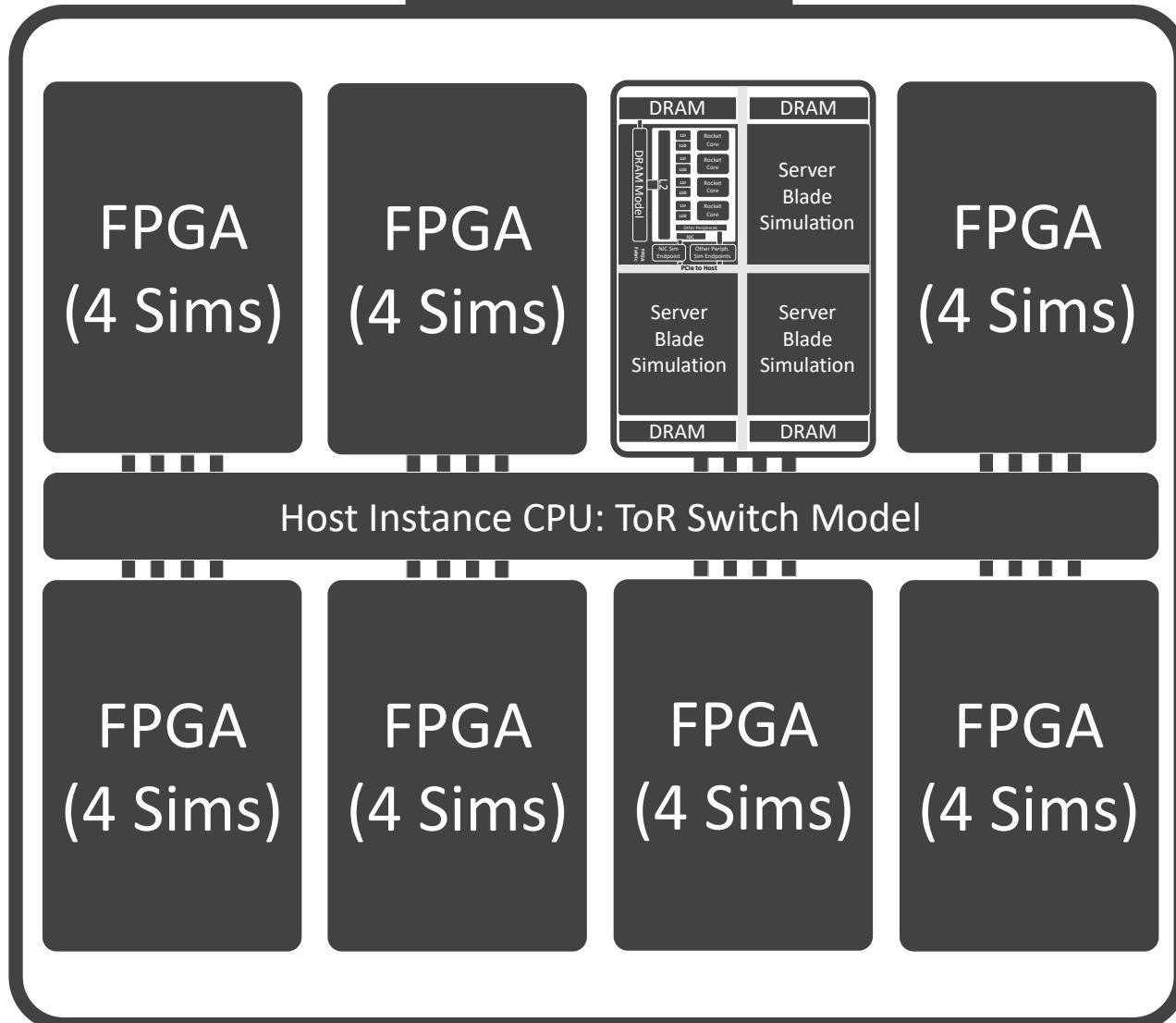
- Modeled System**
- 4 Server Blades
 - 16 Cores
 - 64 GB DDR3
- Resource Util.**
- < 1 FPGA
 - 4/4 Mem Chans
- Sim Rate**
- ~14.3 MHz (netw)



Step 4: Simulating a 32 node rack

Cost:
\$2.60 per
hour (spot)

\$13.20 per
hour (on-
demand)



Modeled System

- 32 Server Blades
- 128 Cores
- 512 GB DDR3
- 32 Port ToR Switch
- 200 Gb/s, 2us links

Resource Util.

- 8 FPGAs =
- 1x f1.16xlarge

Sim Rate

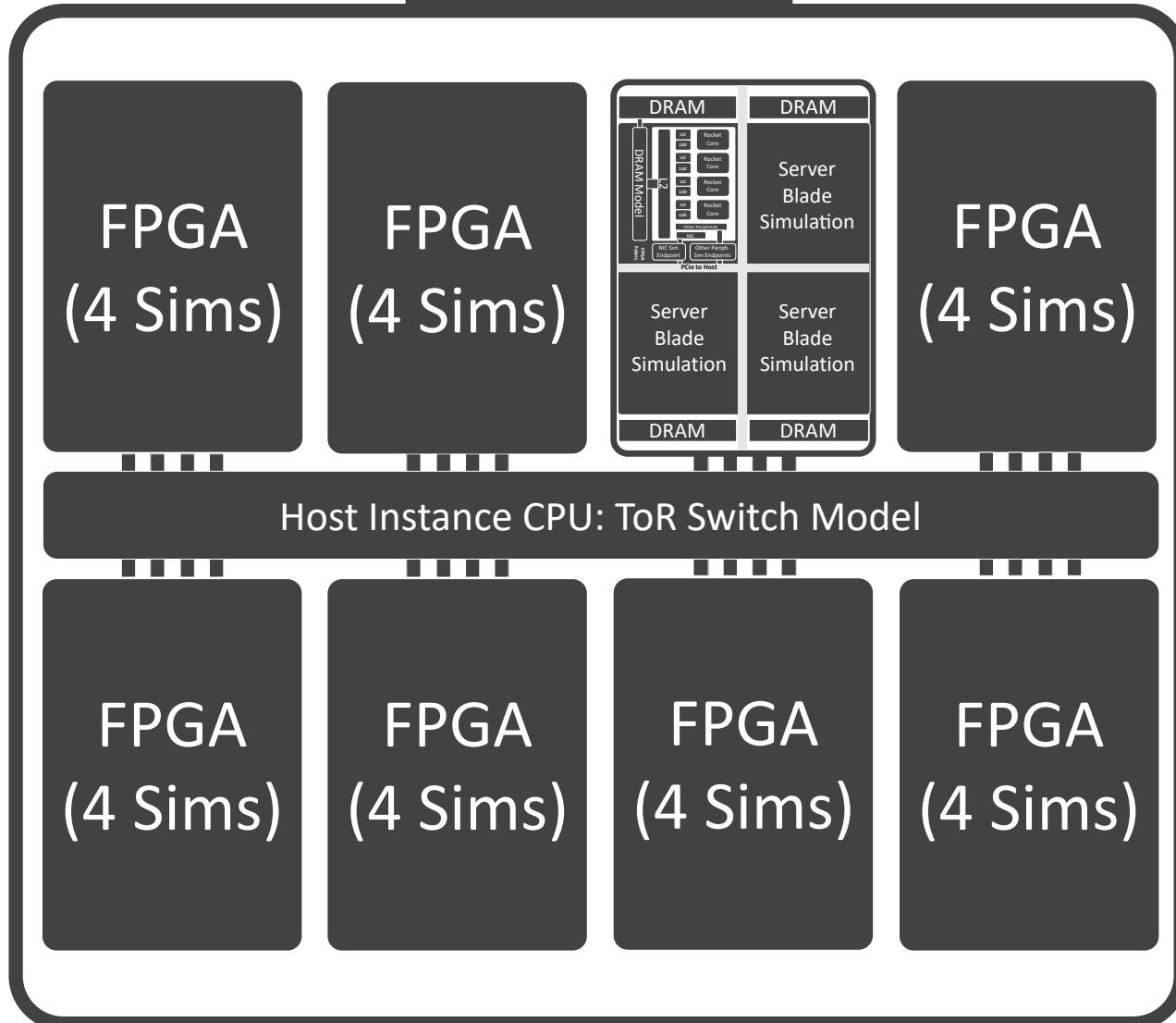
- ~10.7 MHz (netw)



Step 4: Simulating a 32 node rack

Cost:
\$2.60 per
hour (spot)

\$13.20 per
hour (on-
demand)



Modeled System

- 32 Server Blades
- 128 Cores
- 512 GB DDR3
- 32 Port ToR Switch
- 200 Gb/s, 2us links

Resource Util.

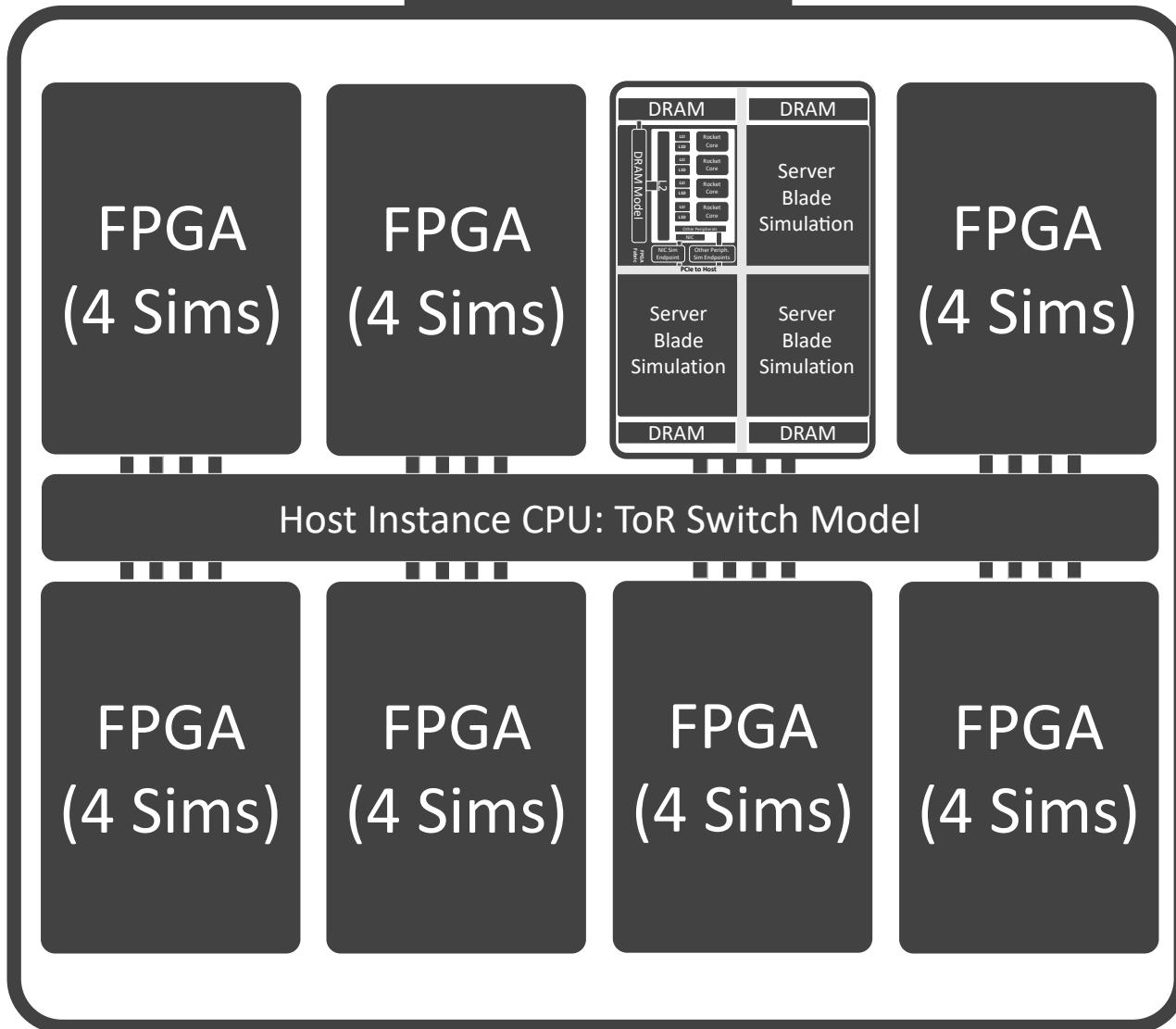
- 8 FPGAs =
- 1x f1.16xlarge

Sim Rate

- ~10.7 MHz (netw)



Step 4: Simulating a 32 node rack



Modeled System

- 32 Server Blades
- 128 Cores
- 512 GB DDR3
- 32 Port ToR Switch
- 200 Gb/s, 2us links

Resource Util.

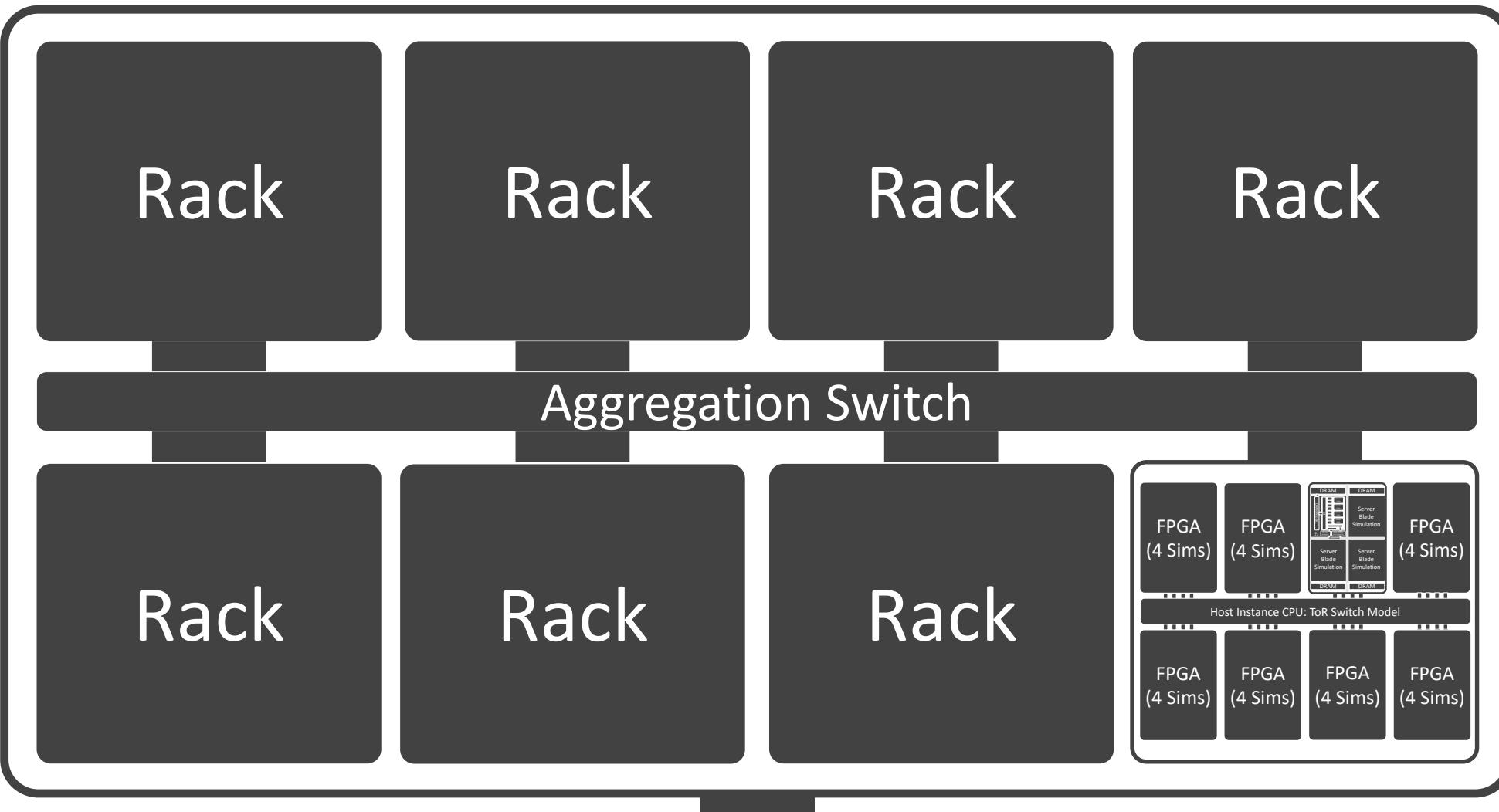
- 8 FPGAs =
- 1x f1.16xlarge

Sim Rate

- ~10.7 MHz (netw)



Step 5: Simulating a 256 node “aggregation pod”



Modeled System

- 256 Server Blades
- 1024 Cores
- 4 TB DDR3
- 8 ToRs, 1 Aggr
- 200 Gb/s, 2us links

Resource Util.

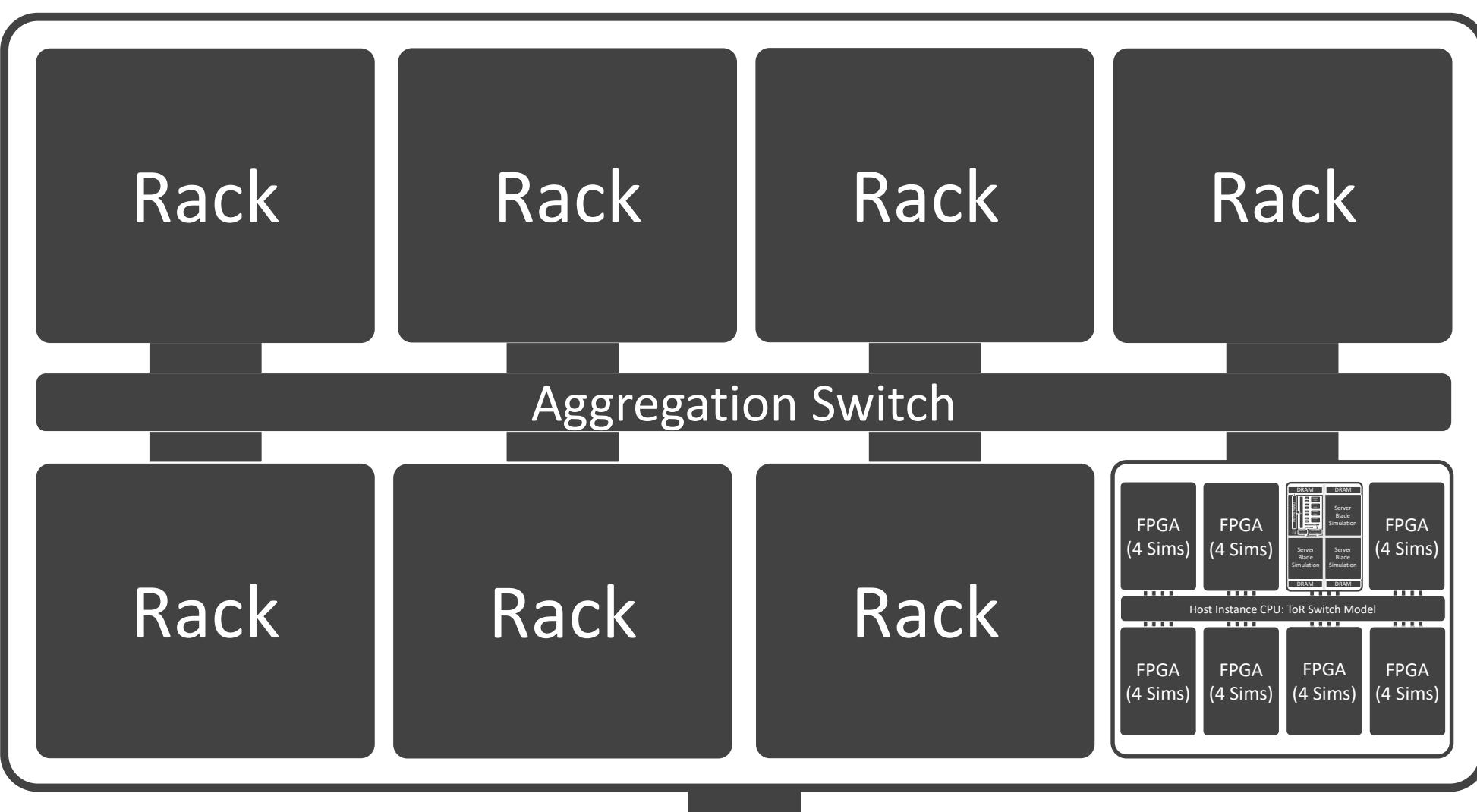
- 64 FPGAs =
- 8x f1.16xlarge
- 1x m4.16xlarge

Sim Rate

- ~9 MHz (netw)



Step 5: Simulating a 256 node “aggregation pod”



Modeled System

- 256 Server Blades
- 1024 Cores
- 4 TB DDR3
- 8 ToRs, 1 Aggr
- 200 Gb/s, 2us links

Resource Util.

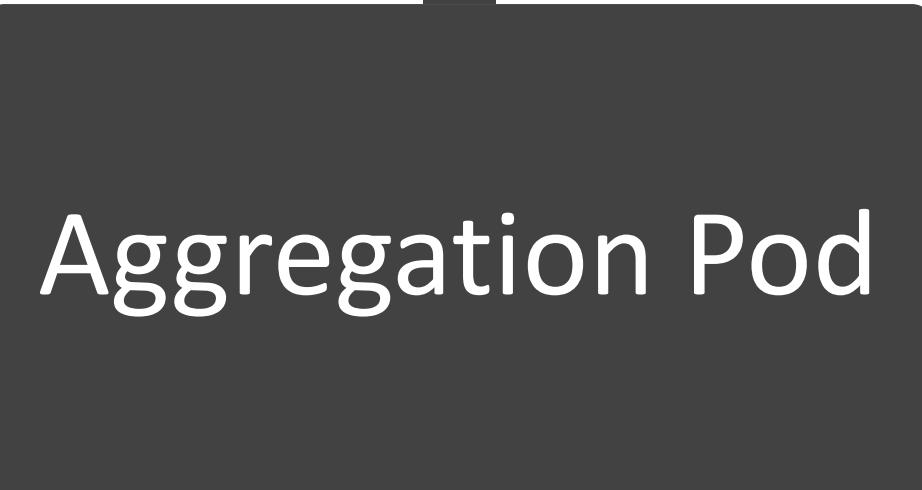
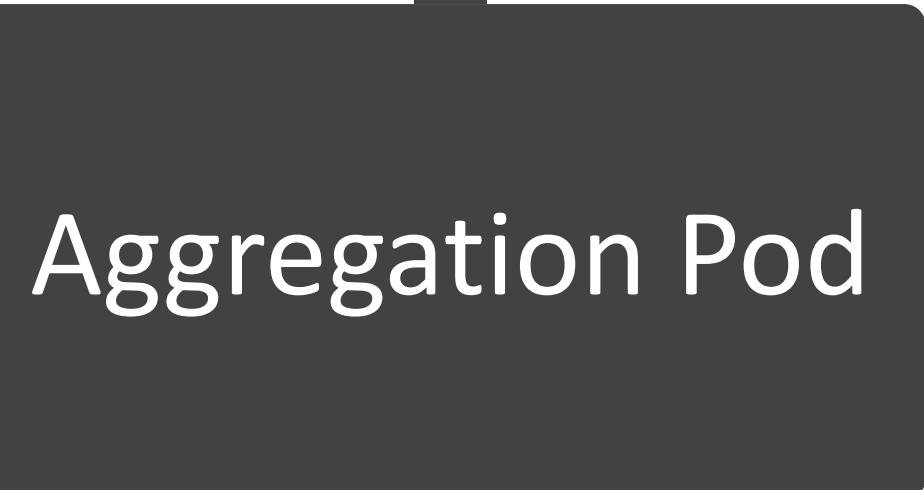
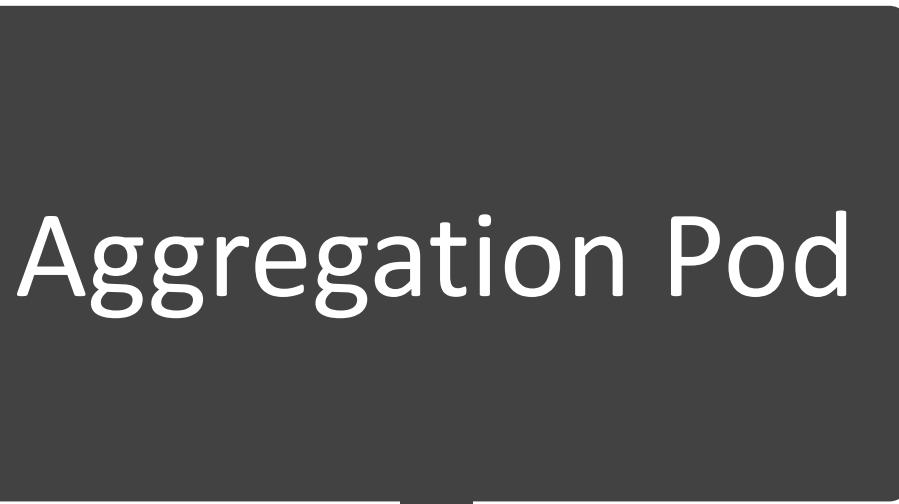
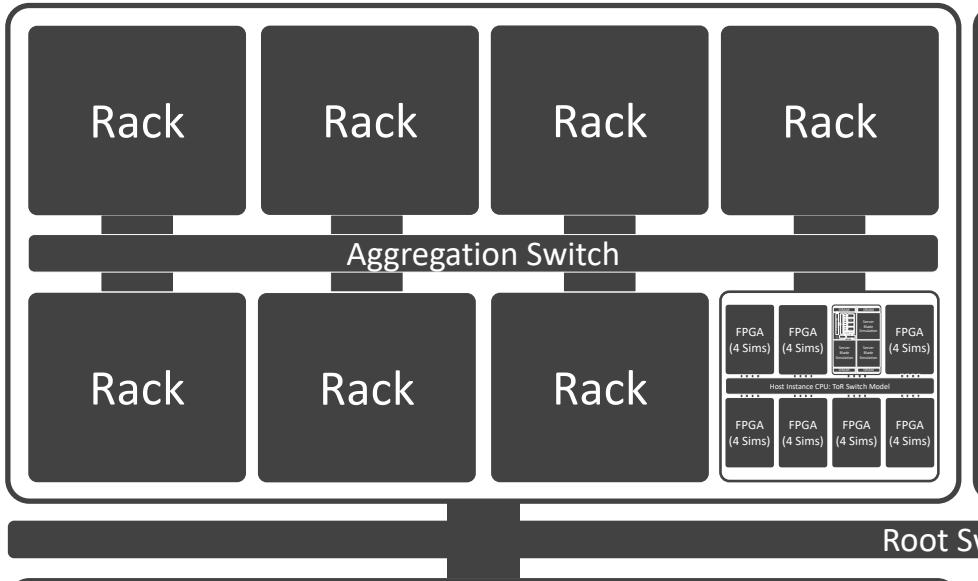
- 64 FPGAs =
- 8x f1.16xlarge
- 1x m4.16xlarge

Sim Rate

- ~9 MHz (netw)



Step 6: Simulating a 1024 node datacenter



Modeled System

- 1024 Servers
- 4096 Cores
- 16 TB DDR3
- 32 ToRs, 4 Aggr, 1 Root
- 200 Gb/s, 2us links

Resource Util.

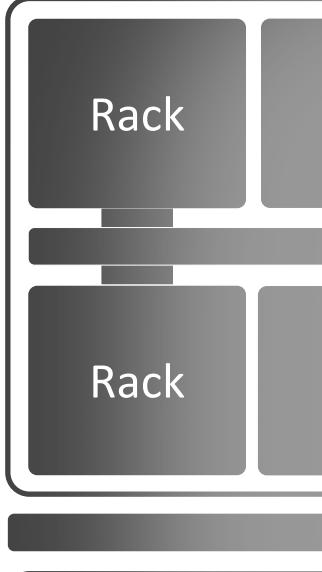
- 256 FPGAs =
- 32x f1.16xlarge
- 5x m4.16xlarge

Sim Rate

- ~6.6 MHz (netw)



Step 6: Simulating a 1024 node datacenter



Harnesses *millions of dollars* of FPGAs
to simulate *1024 nodes cycle-exactly*
with a cycle-accurate *network simulation*
and *global synchronization*
at a cost-to-user of only *100s of dollars/hour*

Aggregation Pod

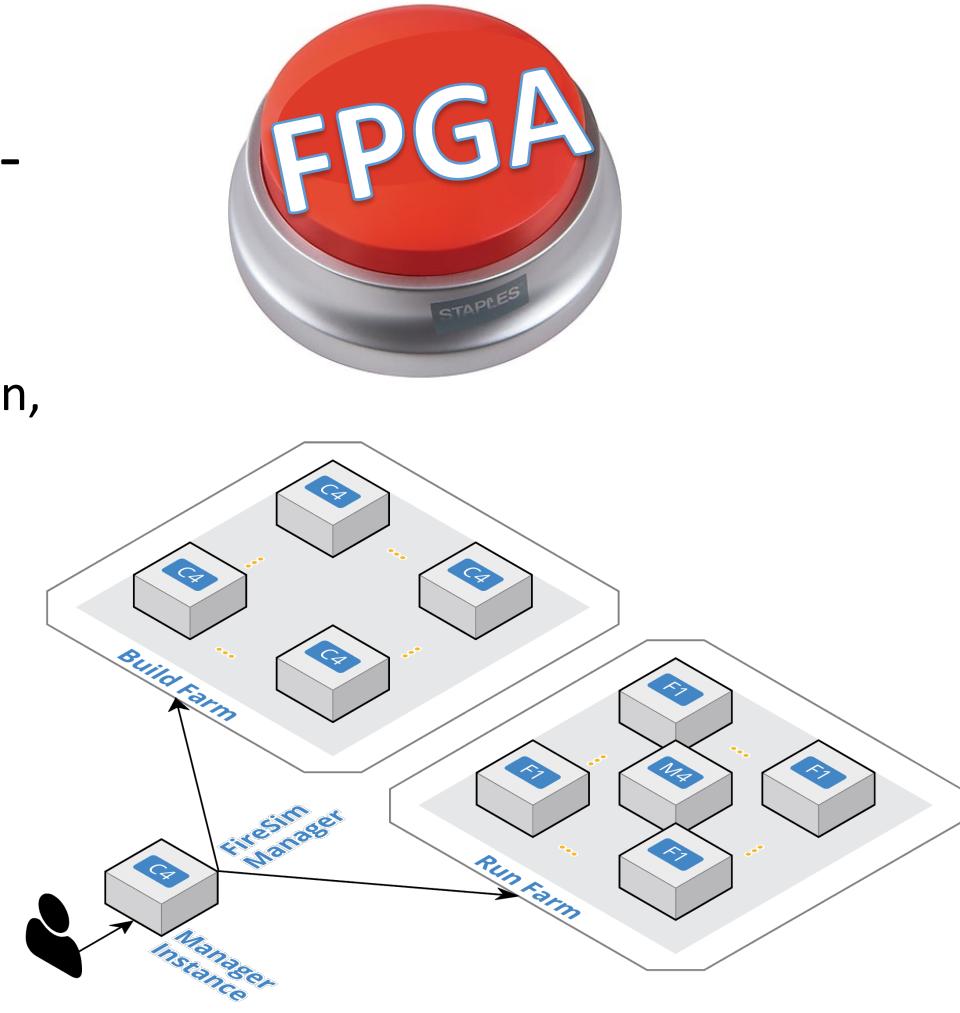
Aggregation Pod

Modeled System
- 1024 Servers
6 Cores
1TB DDR3
ToRs, 4 Aggr, 1
Gb/s, 2us
source Util.
250 FPGAs =
- 32x f1.16xlarge
- 5x m4.16xlarge
Sim Rate
- ~6.6 MHz (netw)



Productive Open-Source FPGA Simulation

- github.com/firesim/firesim, BSD Licensed
- An “easy” button for fast, FPGA-accelerated full-system simulation
 - Plug in your own RTL designs, your own HW/SW models
 - One-click: Parallel FPGA builds, Simulation run/result collection, building target software
 - Scales to a variety of use cases:
 - Networked (performance depends on scale)
 - Non-networked (150+ MHz), limited by your budget
- **firesim command line program**
 - Like docker or vagrant, but for FPGA sims
 - User doesn’t need to care about distributed magic happening behind the scenes

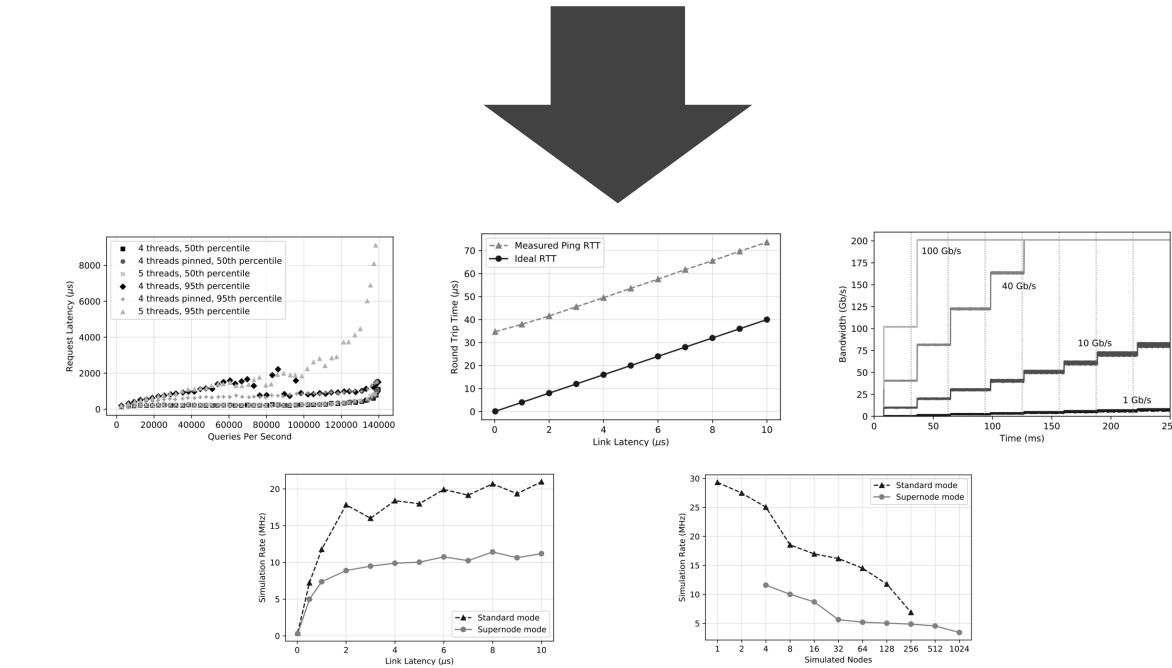




Productive Open-Source FPGA Simulation

- Scripts can call `firesim` to fully automate distributed FPGA sim
 - **Reproducibility:** included scripts to reproduce ISCA 2018 results
 - e.g. scripts to automatically run SPECInt2017 **reference inputs** in ≈ 1 day
 - Many others included
 - Several user papers have gone through artifact evaluation using FireSim (*nanoPU*, *FirePerf*, *Protobuf accel.*, etc.)
- 200+ pages of documentation: <https://docs.firesim.org>
- AWS provides grants for researchers:
<https://aws.amazon.com/grants/>
- Xilinx University Program provides FPGA donations for university researchers:
<https://www.xilinx.com/support/university.html>

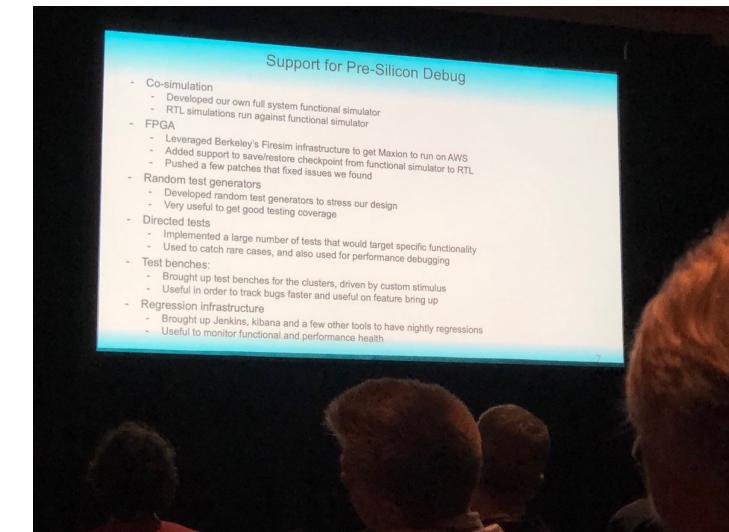
```
$ cd fsim/deploy/workloads  
$ ./run-all.sh
```



Join the FireSim Community!: Open-source users and industrial users



- More than 190 mailing list members and 800 unique cloners per-week
- Projects with public FireSim support
 - Chipyard
 - Rocket Chip
 - BOOM
 - Hwacha Vector Accelerator
 - Keystone Secure Enclave
 - Gemmini
 - NVIDIA Deep Learning Accelerator (NVDLA)
 - NVIDIA Blog post:
<https://devblogs.nvidia.com/nvdl/>
 - BOOM Spectre replication/mitigation
 - Protobuf Accelerator
 - Too many to list here!
- Companies publicly announced using FireSim
 - Esperanto Maxion ET
 - Intensivate IntenCore
 - SiFive validation paper @ VLSI'20
 - Galois and Lockheed Martin (DARPA SSITH/FETT)



Esperanto announcement at RISC-V Summit 2018





FireSim in DARPA FETT

- DARPA SSITH: Building hardware defenses to address common software vulnerabilities
- DARPA FETT: How good are the defenses built in SSITH?
 - Multiple designs hosted for attack in FireSim [1]
- “Morpheus II: A RISC-V Security Extension for Protecting Vulnerable Software and Hardware”
 - Developed by UT Austin, U Mich., Agita Labs
 - Hosted on FireSim for FETT [2]
 - Over 500 attackers tried to break Morpheus II defenses, working for large bug bounties. None succeeded [3]



[1] K. Hopfer. Leveraging Amazon EC2 F1 Instances for Development and Red Teaming in DARPA’s First-Ever Bug Bounty Program. AWS APN Blog. May 2021.

[2] A. Harris, et. al., “Morpheus II: A RISC-V Security Extension for Protecting Vulnerable Software and Hardware”. In proceedings of the 2021 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), December 2021.

[3] T. Austin., et. al., “Morpheus II: A RISC-V Security Extension for Protecting Vulnerable Software and Hardware”. In HotChips 33, August 2021.



Join the FireSim Community!:

Academic Users and Awards



- **ISCA '18:** Maas et. al. HW-GC Accelerator (**Berkeley**)
- **MICRO '18:** Zhang et. al. “Composable Building Blocks to Open up Processor Design” (**MIT**)
- **RTAS '20:** Farshchi et. al. BRU (**Kansas**)
- **EuroSys '20:** Lee et. al. Keystone (**Berkeley**)
- **OSDI '21:** Ibanez et. al. nanoPU (**Stanford**)
- **CCS '21:** Ding et. al. “Hardware Support to Improve Fuzzing Performance and Precision” (**Georgia Tech**)
- Too many to list here: see FireSim website for more!
 - <https://fires.im/publications/#userpapers>
- Awards: FireSim ISCA '18 paper:
 - IEEE Micro Top Pick
 - CACM Research Highlights Nominee from ISCA '18
- Awards: FireSim users:
 - ISCA '18 Maas et. al.:
 - IEEE Micro Top Pick
 - MICRO '18 Zhang et. al.:
 - IEEE Micro Top Pick
 - MICRO '21 Gottschall et. al.:
 - MICRO-54 Best paper runner-up
 - MICRO '21 Karandikar et. al.:
 - MICRO-54 Distinguished Artifact winner
 - IEEE Micro Top Pick Honorable Mention
 - DAC '21 Genc et. al.:
 - DAC 2021 Best Paper winner



Join the FireSim Community!:

Academic Users and Awards



- ISCA '18: Maas et. al. HW-GC Accelerator (Berkeley)
- MICRO '18: Blocks to Instructions
- RTAS '20: Performance Analysis
- EuroSys '20: Performance Analysis
- OSDI '21: Performance Analysis
- CCS '21: Improve Performance (Georgia Tech)
- Too many to list here. See FireSim website for more!
 - <https://firesim.org/publications/#userpapers>
- Awards: FireSim ISCA '18 paper:
 - IFFF Micro Top Pick
- IEEE Micro Top Pick Honorable Mention
- DAC '21 Genc et. al.:
 - DAC 2021 Best Paper winner

FireSim has been used in published work from authors at over 20 academic and industrial institutions*

**actually used, not only cited*





Two new FireSim features!

- Local FPGA support!
 - Re-architecting of FireSim Manager, Drivers, and Shells to support adding new FPGA and host platforms
 - Now supports Xilinx Alveo XRT-enabled FPGAs (e.g. U250)
- Distributed Meta-simulations
 - Early-stage accelerator development requires running many parallel verilator/vcs sims, but these are traditionally accessed via Make-system in Chipyard/FireSim
 - FireSim manager now supports distributing metasims using the same machinery as distributing FPGA simulations, on both EC2 and local machines
 - Same user-interface for workload/job specification/mapping, constructing heterogeneous systems, running sims, and collecting outputs (now including waveforms)

Abe will talk about these in detail @ 4:40pm!





Questions?



Berkeley Architecture Research

Learn More:

Web: <https://fires.im>

Docs: <https://docs.fires.im>

GitHub: <https://github.com/firesim/firesim>

Mailing List:

<https://groups.google.com/forum/#!forum/firesim>

 [@firesimproject](https://twitter.com/firesimproject)

Email: sagark@eecs.berkeley.edu

The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0000849, by DARPA, Award Number HR0011-12-2-0016, and by NSF CCRI ENS Chipyard Award #2016662. Research was also partially funded by SLICE/ADEPT Lab industrial sponsors and affiliates Amazon, Apple, Google, Intel, Qualcomm, and Western Digital, and RISE Lab sponsor Amazon Web Services. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.