



An Open-Source Platform
for Scalable FPGA-
Accelerated Hardware
Simulation in the Cloud

<https://fires.im>



@firesimproject

Sagar Karandikar, David Biancolin, Howard Mao, Alon Amid, Nathan Pemberton, Albert Magyar,
Albert Ou, Qijing Huang, Randy Katz, Borivoje Nikolić, Jonathan Bachrach, Krste Asanović



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

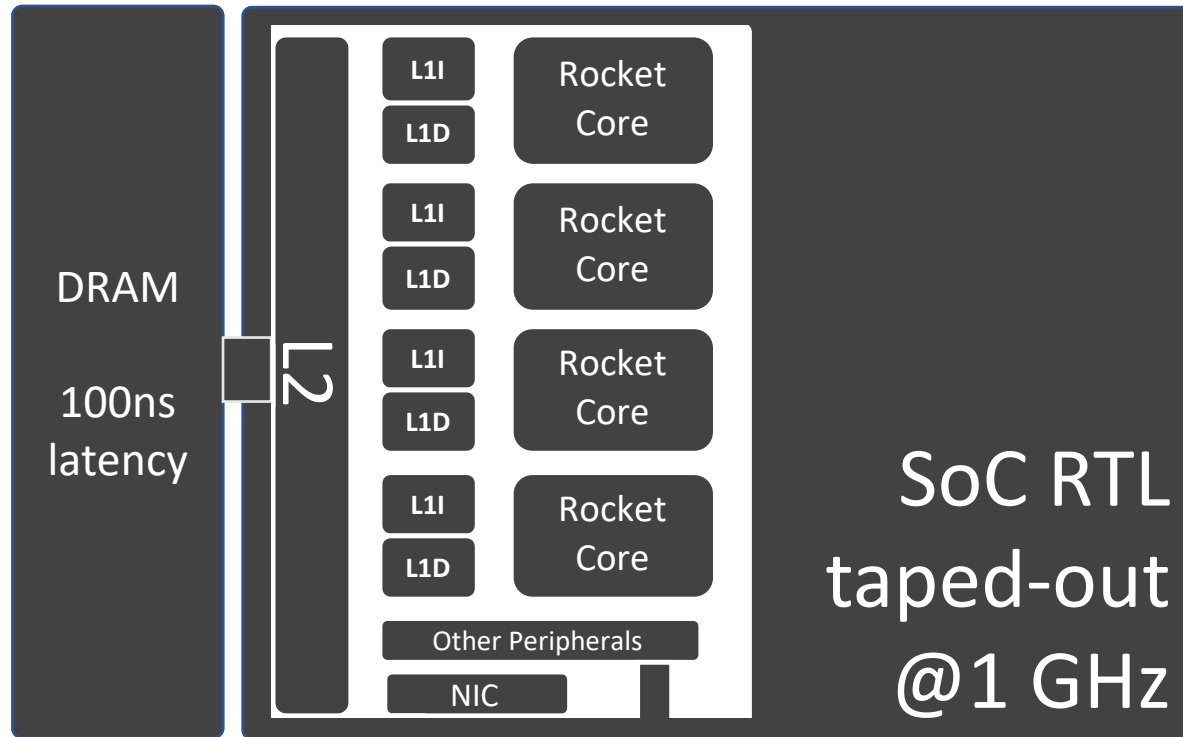
1. High-level Simulation
2. Write RTL + Software, plug into your favorite ecosystem (e.g. Chipyard)
3. Co-design in software RTL sim (e.g. Verilator, VCS, etc.)
 - Run microbenchmarks
4. **Co-design in FPGA-accelerated simulation**
 - **Boot an OS and run the complete software stack, obtain realistic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





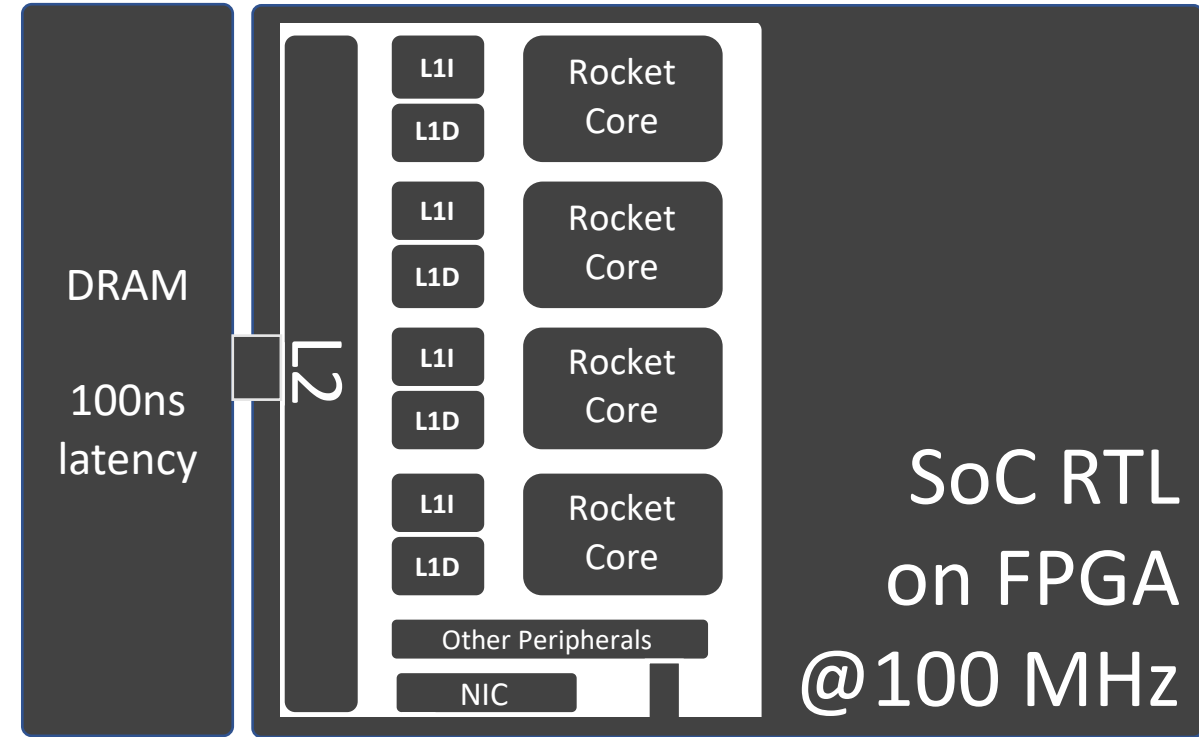
What about FPGA prototyping?

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 implications:

- 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



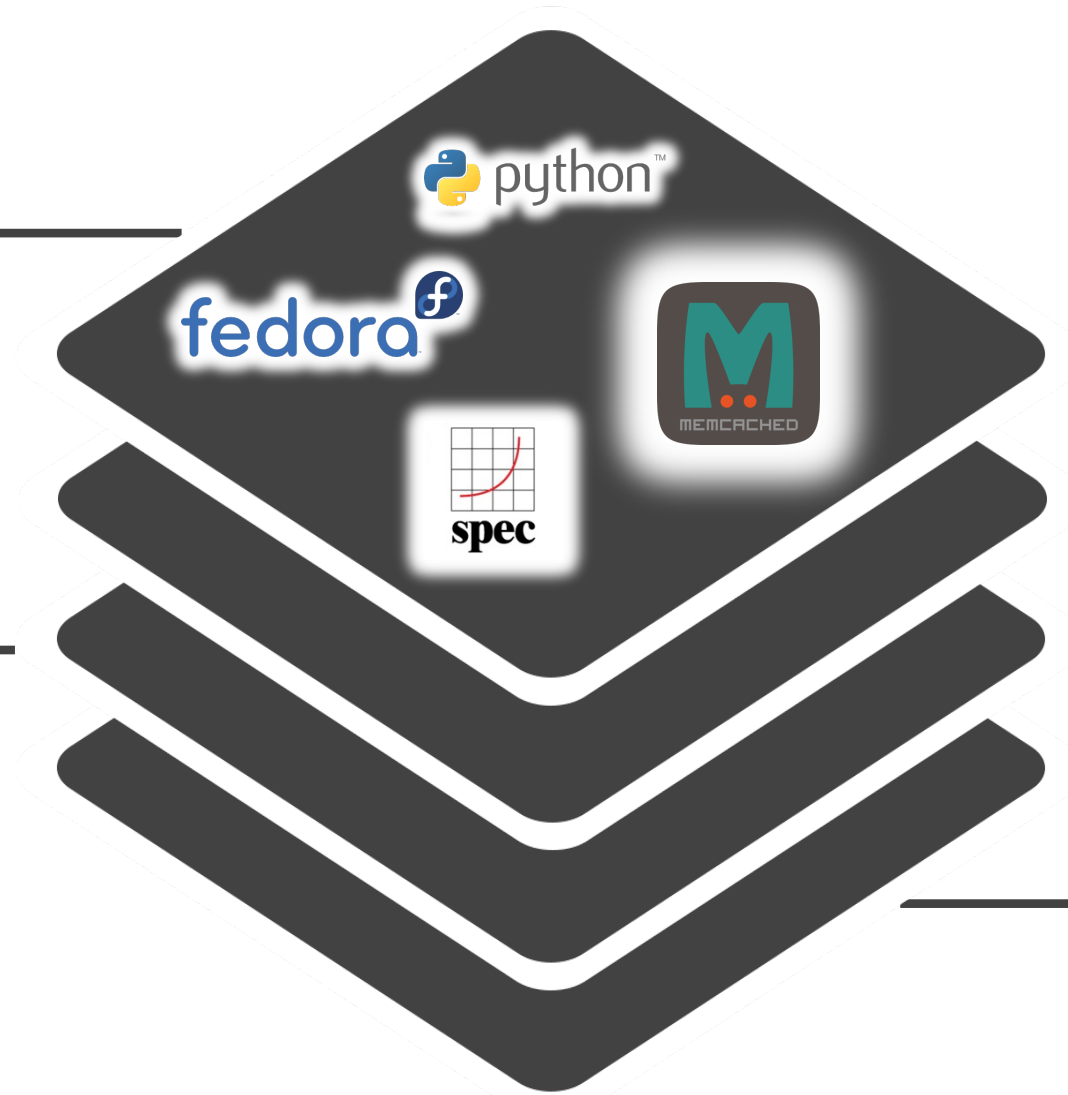
How? Useful Trends Throughout the Stack

Open ISA



CHISEL

High-Productivity
Hardware Design
Language & IR



Open, Silicon-Proven
SoC Implementations



FPGAs in the 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, either via Chisel or Verilog via Yosys*)
 - 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)



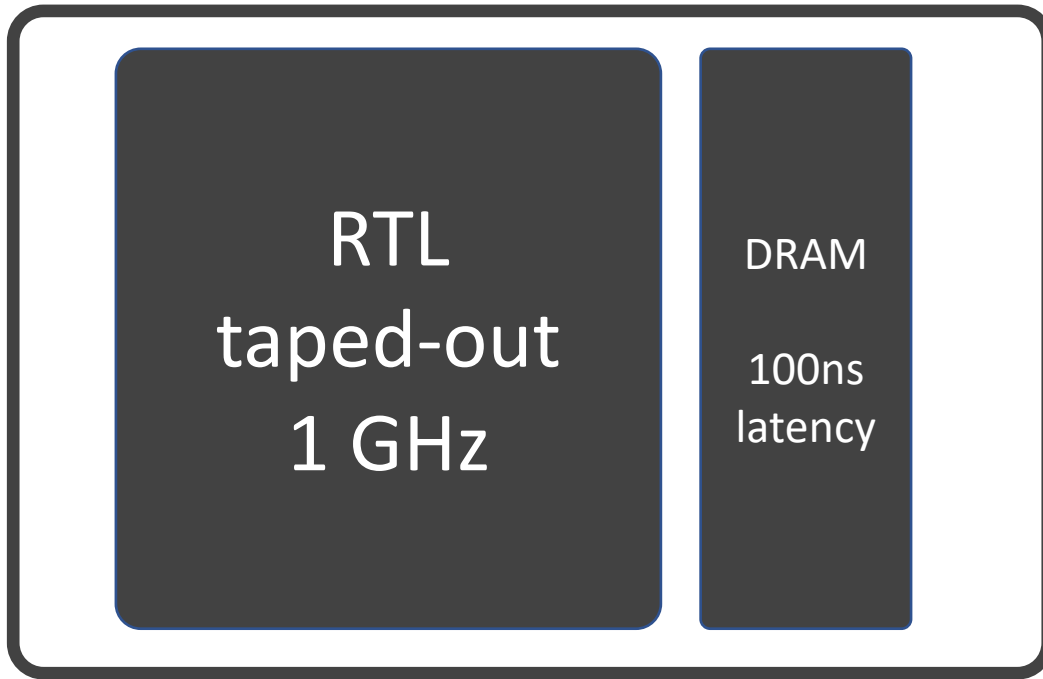
Three Distinguishing Features of FireSim

- 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>)



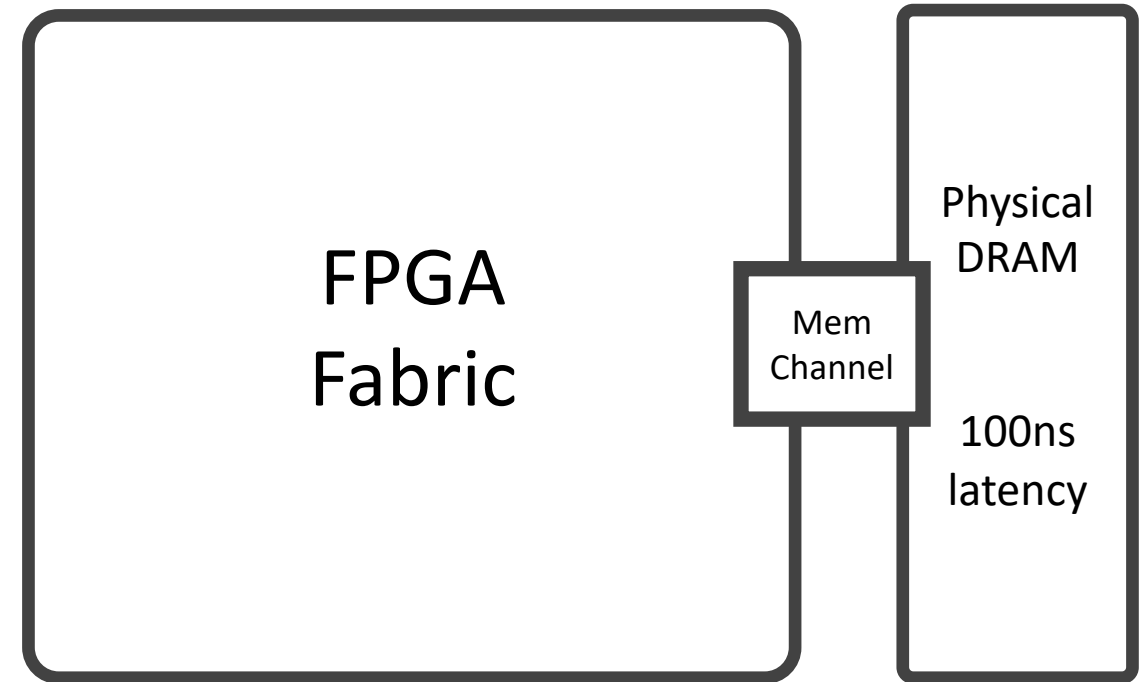
Separating Target and Host

Target: the machine under simulation



Closed simulation world.

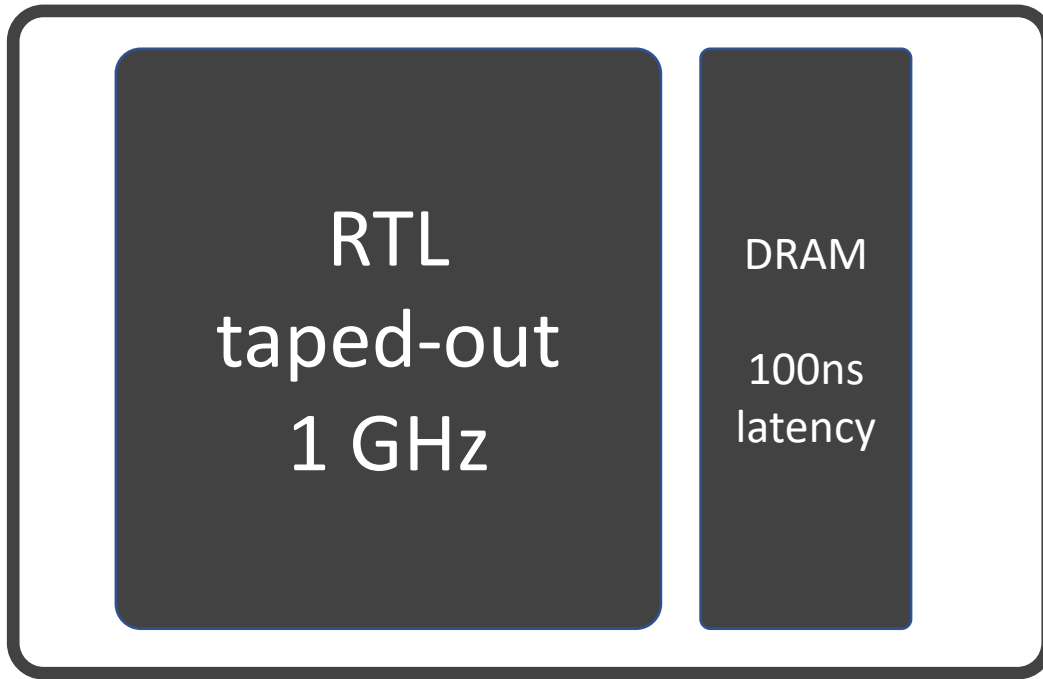
Host: the machine executing (*hosting*) the simulation





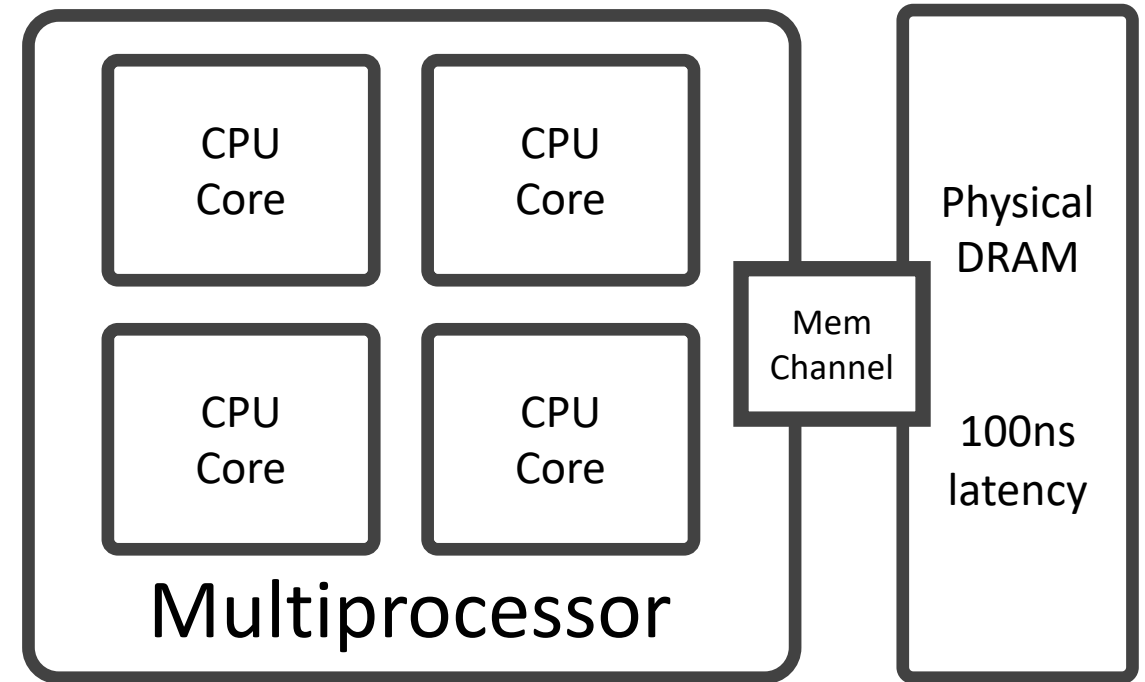
Separating Target and Host

Target: the machine under simulation



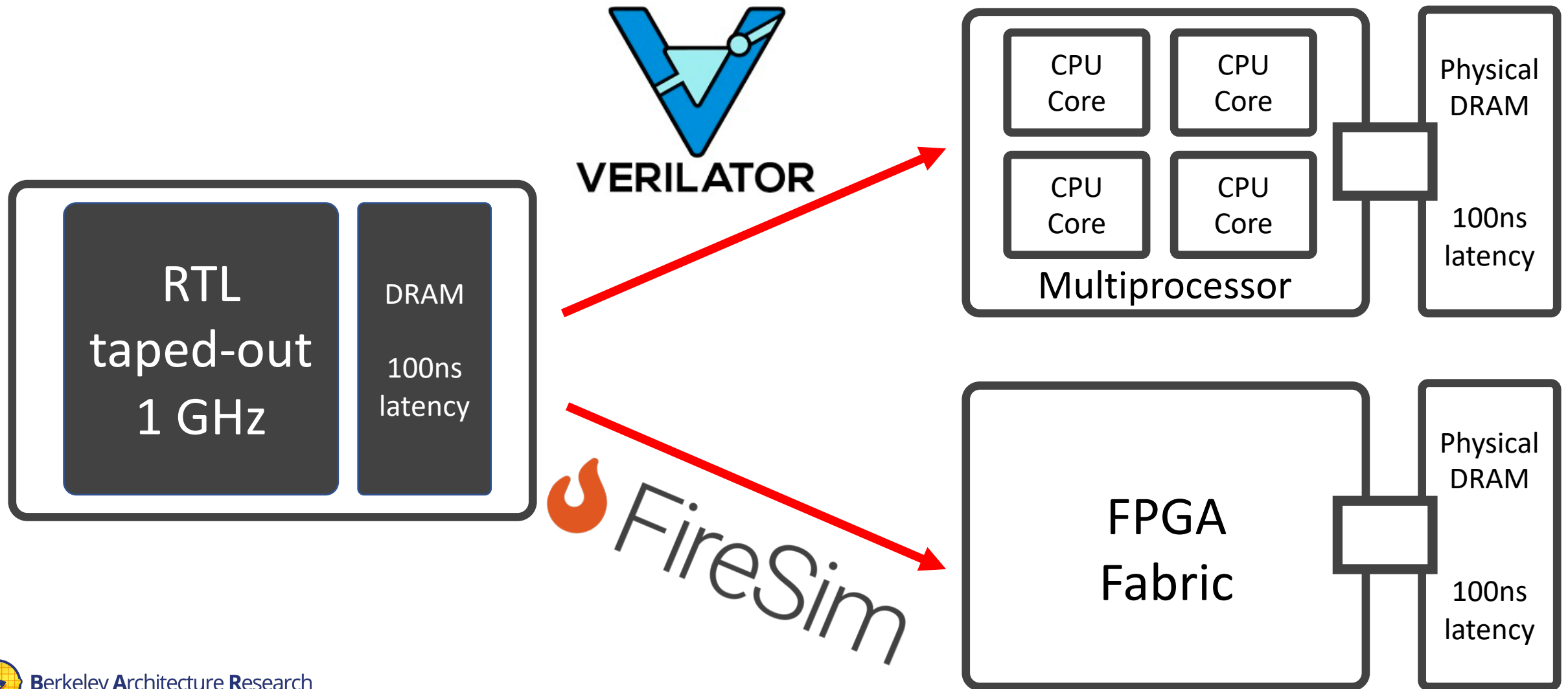
Closed simulation world.

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





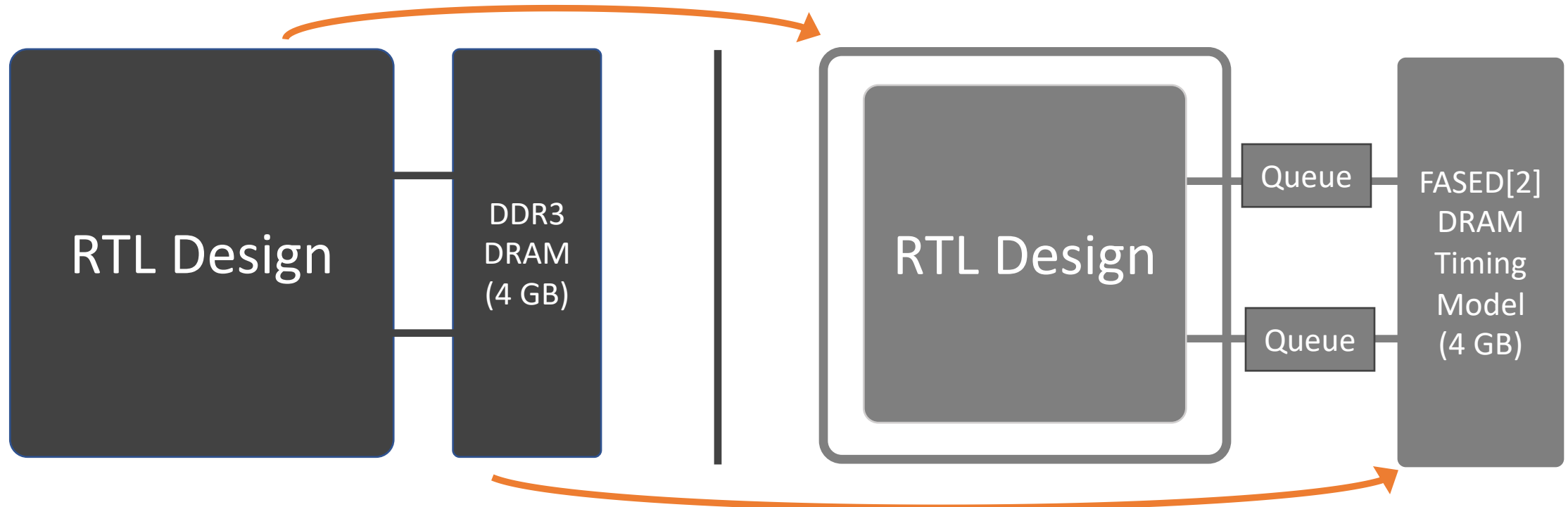
FireSim Generates FPGA-Hosted Simulators





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

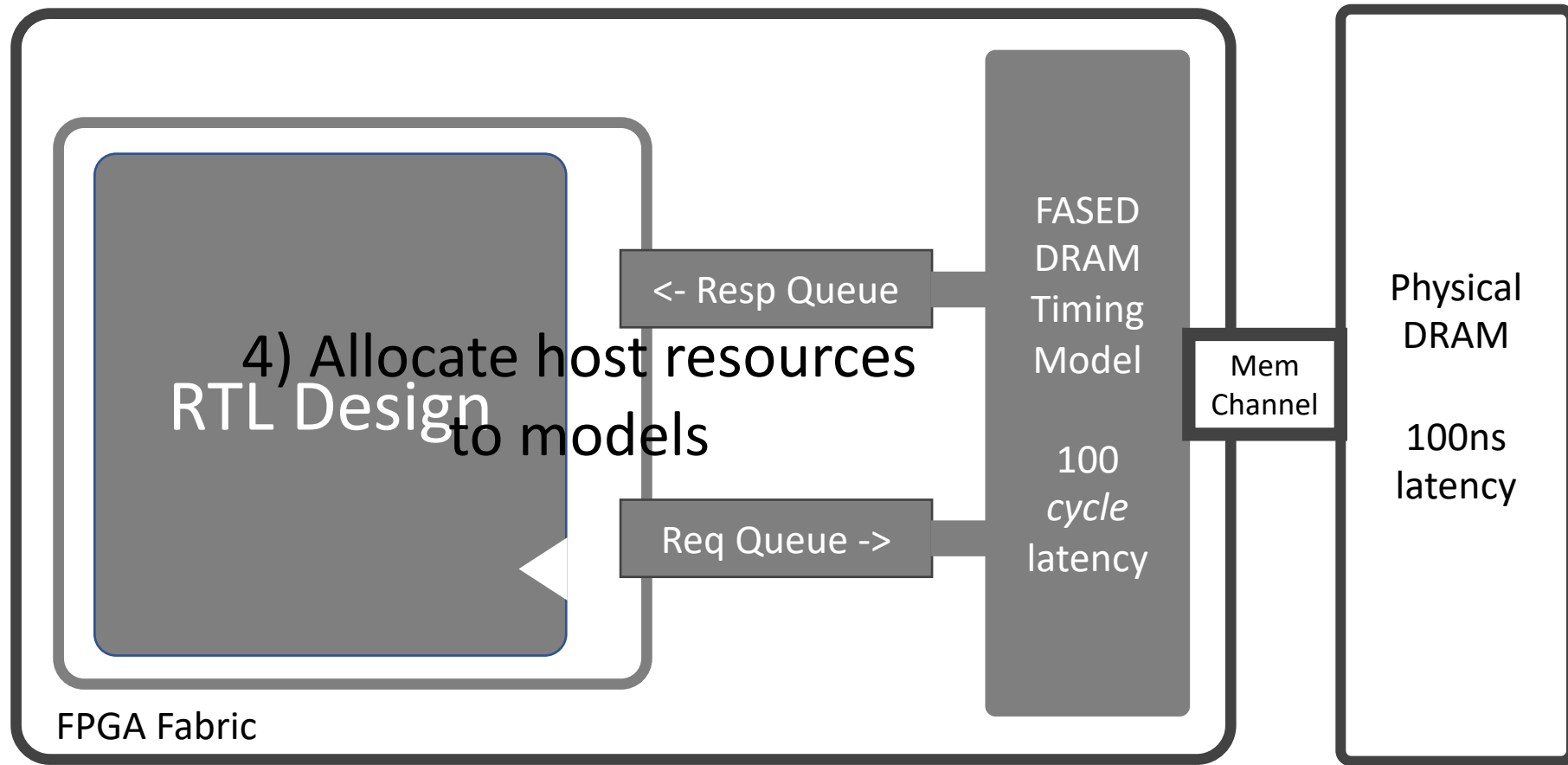
[1] *Theory of Latency Insensitive Design*, Carloni et al, also see: RAMP

[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?



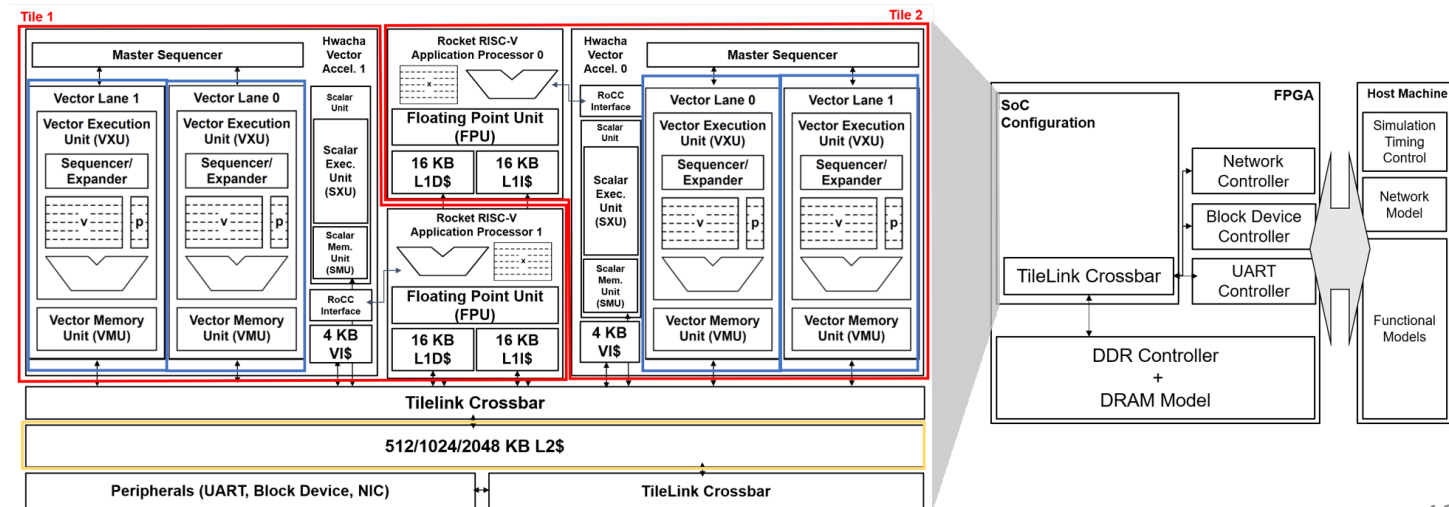
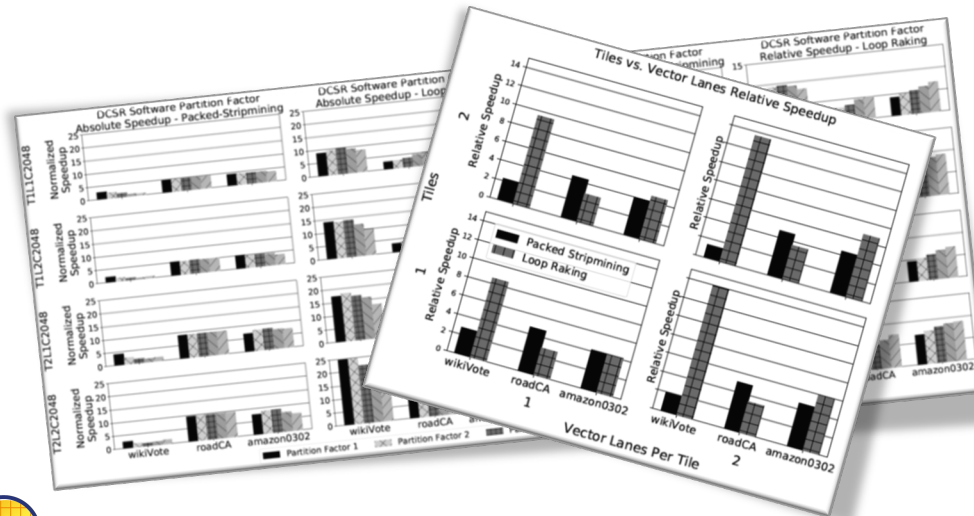
Target Designs available in FireSim

- Chipyard (<https://github.com/ucb-bar/chipyard>)
 - Combines large collection of open-source IP to enable constructing complex RISC-V SoCs
 - RISC-V Cores: Rocket (in-order), Berkeley Out-of-Order Machine (BOOM), Ariane (Verilog in-order core)
 - Uncore components: L1, L2 caches, TileLink Xbar interconnect, TileLink Ring Interconnect
 - Accelerators: Hwacha vector co-processor, Gemmini Systolic Array (ML)
 - Peripherals: Disk, UART, Ethernet NIC
- NVIDIA Deep Learning Accelerator (NVDLA)
 - <https://github.com/nvdla/firesim-nvdl>
- Vivado HLS-Generated Accelerators
- PicoRV32 (via Yosys)
- Bring your own:
 - Chisel (really, FIRRTL) designs get all features
 - Verilog can be ingested through Yosys or blackbox FAME-1 clock gating



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

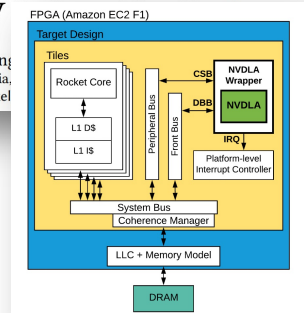
- Security:
 - BOOM Spectre replication
 - A. Gonzalez, et. al., *Replicating and Mitigating Spectre Attacks on an Open Source RISC-V Microarchitecture*, CARRV '19
 - Keystone Enclave performance evaluation
 - D. Lee, et. al., *Keystone*, EuroSys '20
- Accelerator evaluation
 - Chisel-based accelerators:
 - ML (H. Genc, et. al., *Gemmini*, Arxiv)
 - Garbage collection (M. Maas, et. al., *A Hardware Accelerator for Tracing Garbage Collection*, ISCA '18)
 - 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)
- Novel scale-out systems
 - nanoPU NIC-CPU co-design (S. Ibanez, et. al., *nanoPU*, OSDI '21)



Integrating NVIDIA Deep Learning Accelerator (NVDLA) with RISC-V

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
 - 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)

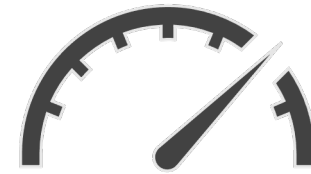
BROOM
An open-source out-of-order processor resilient low-voltage operation in RISC-V
Christopher Cello, Pi-Feng Chi, Krste Asanović, David Patterson, and Bo Xie
Hot Chips 2018

ASPIRE
Directed tests and a random Verilator/VCS/FPGA simulator
VCS for post-gl/par simulation
Speculative OOO pipelines
- Need tests that build up a lot of state
- Need tests that cover a lot of state
- Assertions are king.

BOOM-v2 Assertion Results

Benchmark	Assertion	Cycle(B)	Simulation Time (Min)
483.xalancbmk.test	Invalid write back in ROB	1.9	3.4
464.h264ref.test	Pipeline hung	3.2	3.8
471.omnetpp.test	Pipeline hung	3.3	3.9
445.gobmk.test	Invalid write back in ROB	14.9	9.0
471.omnetpp.ref	Pipeline hung	62.6	22.2
401.bzip2.ref	Wrong JAL target	473.7	164.6

Cost: 2 x 50 cents / hour
Total cost: \$2 (compilation) + 2 x \$1.56 (simulation) = \$5.12



FirePerf



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

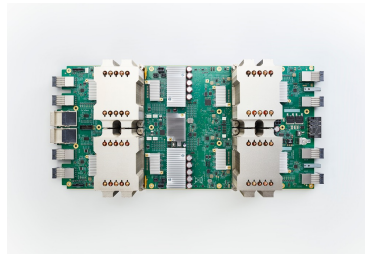
[1] S. Karandikar et. al., “FireSim: FPGA-Accelerated Cycle-Exact Scale-Out System Simulation in the Public Cloud.” *ISCA 2018*

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



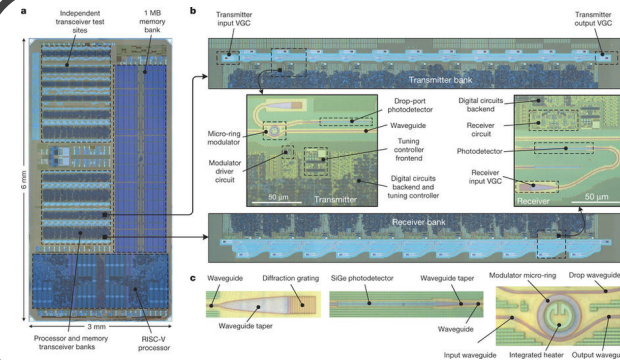


The new datacenter hardware environment



The end of Moore's Law

Custom Silicon in the Cloud

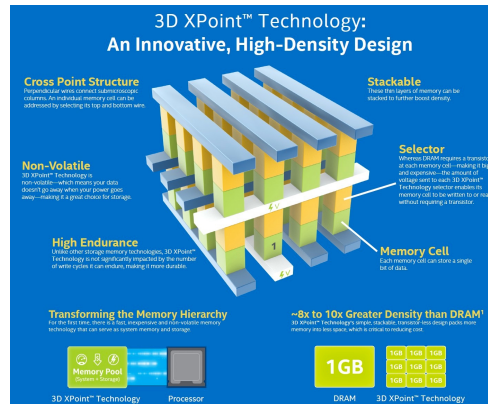


Faster networks

e.g. Silicon Photonics

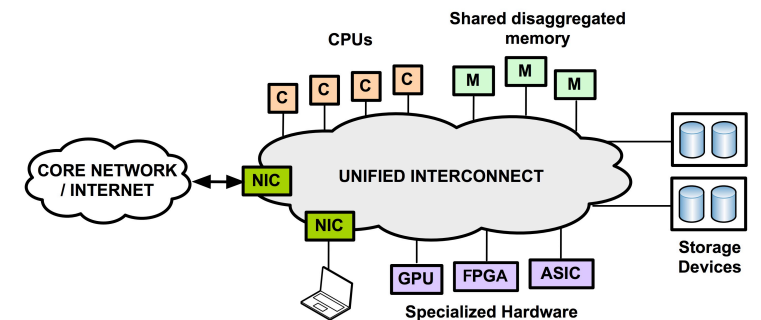
Deeper memory/storage hierarchies

e.g. 3DXPoint, HBM



New datacenter architectures

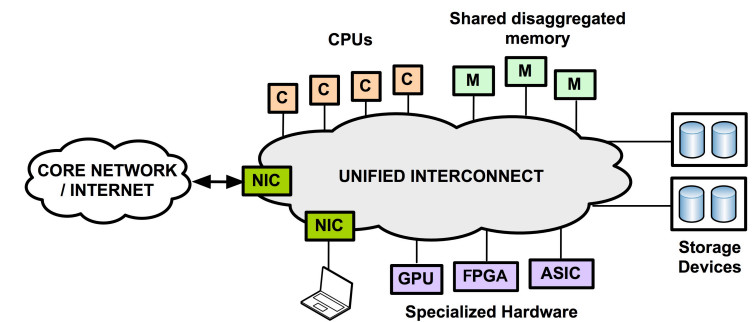
e.g. disaggregation



[1]

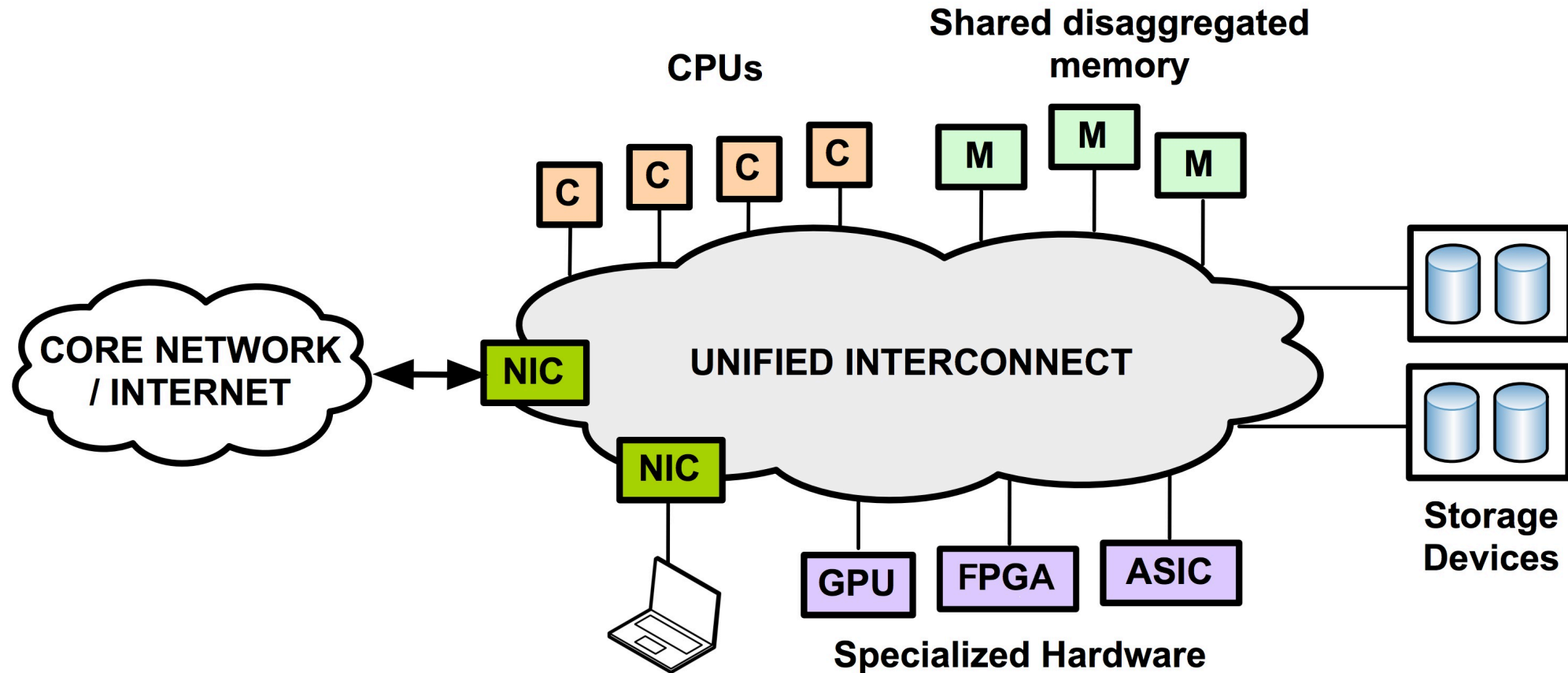


The new datacenter hardware environment





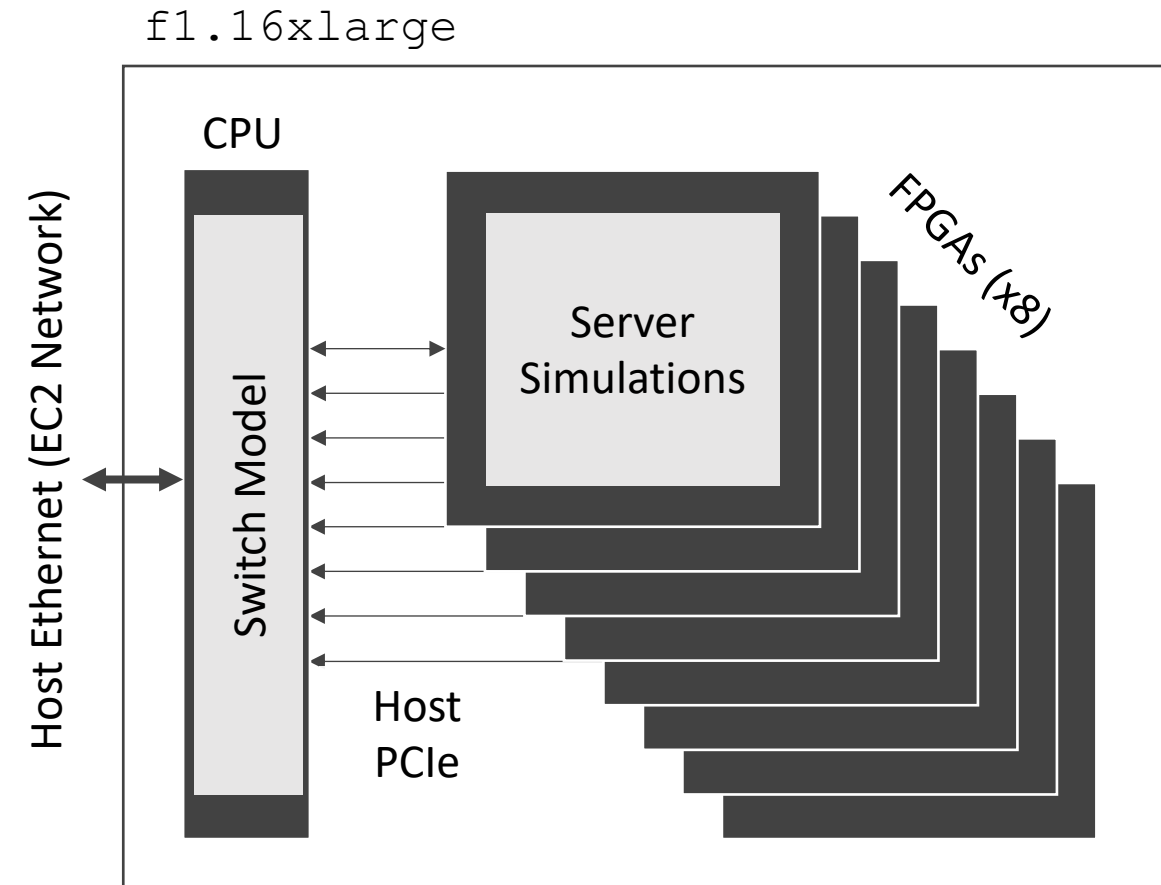
Disaggregated Datacenters



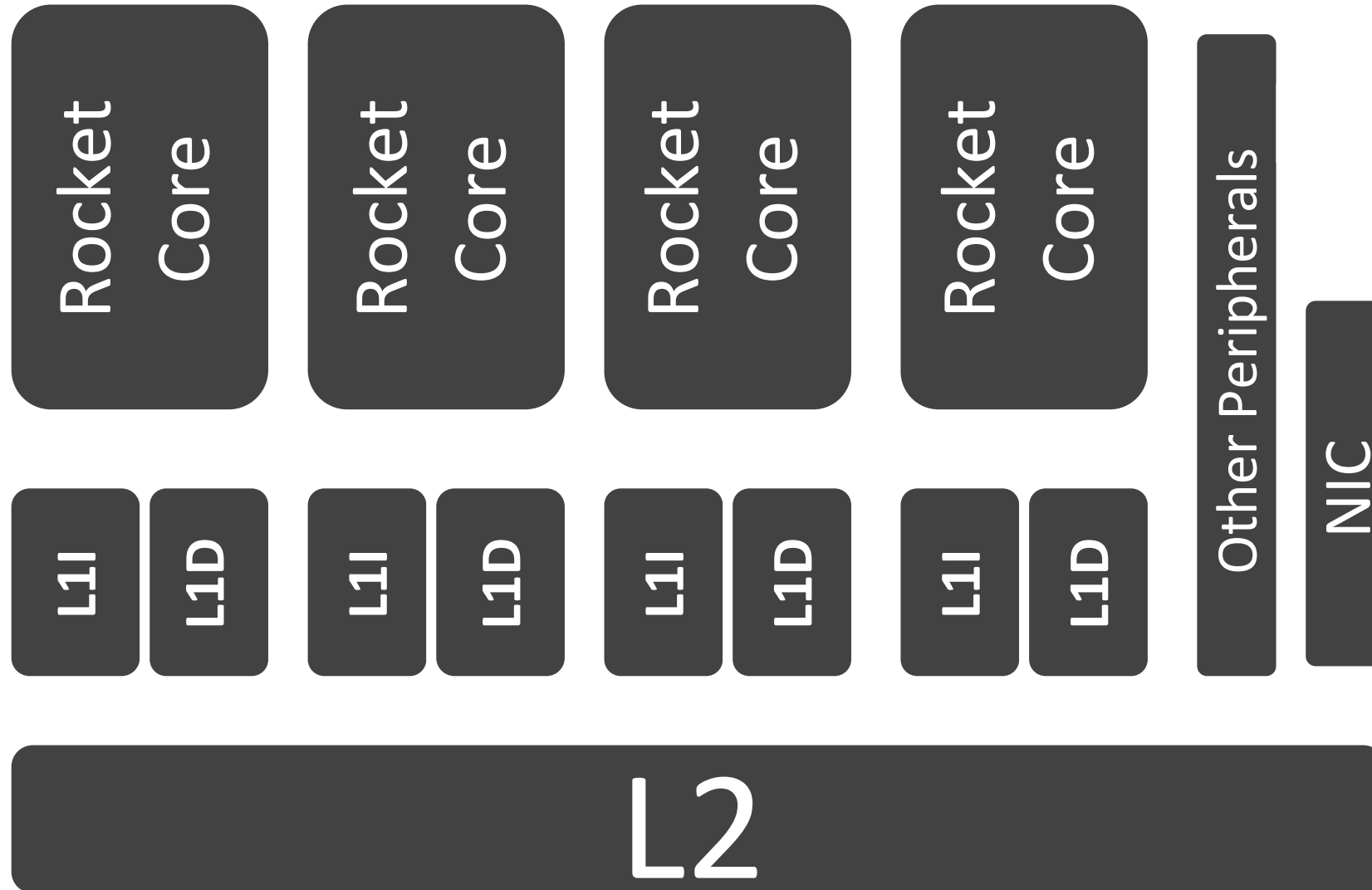


Mapping a datacenter simulation

- 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.

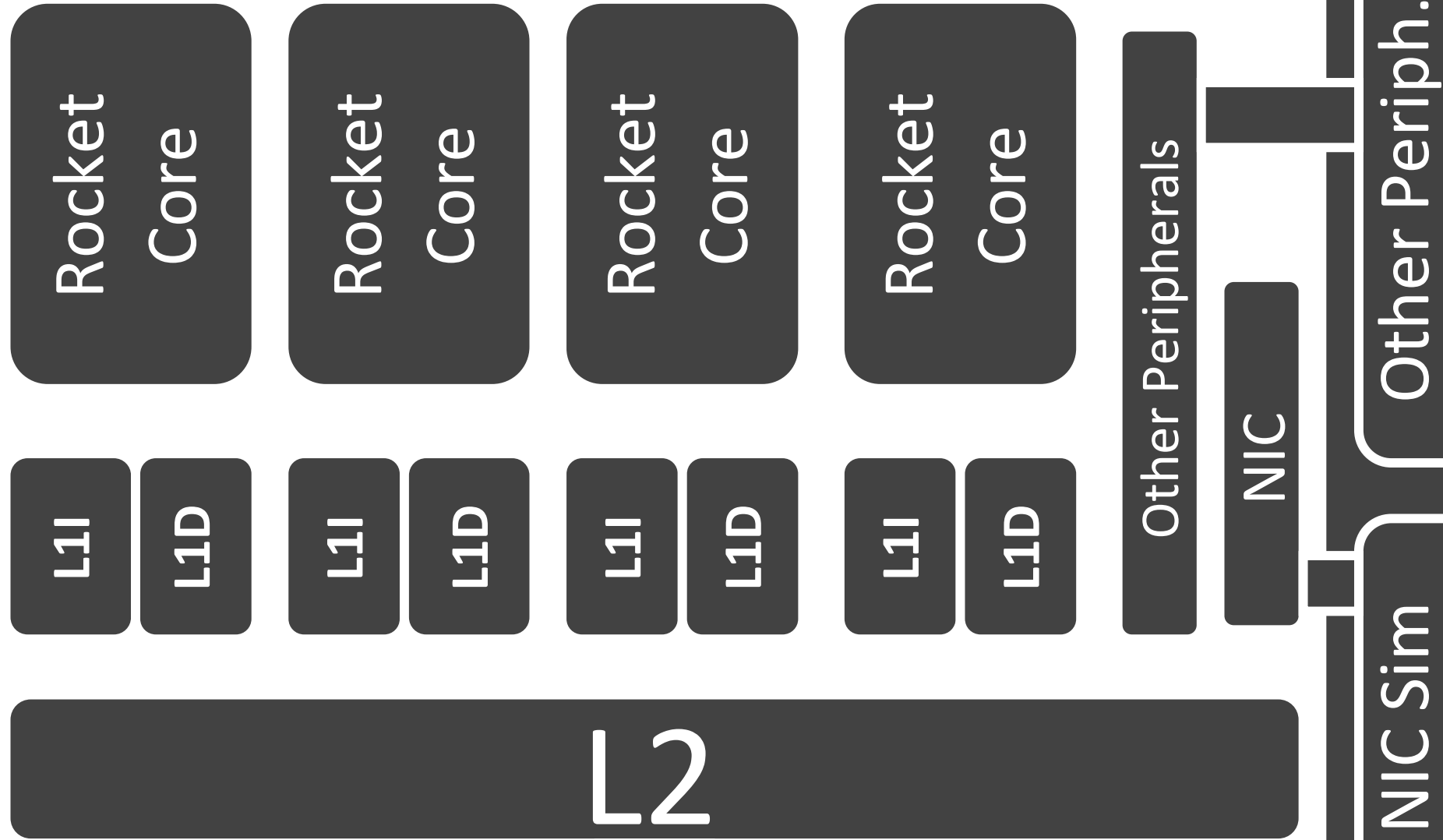
- < ¼ of an FPGA

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.

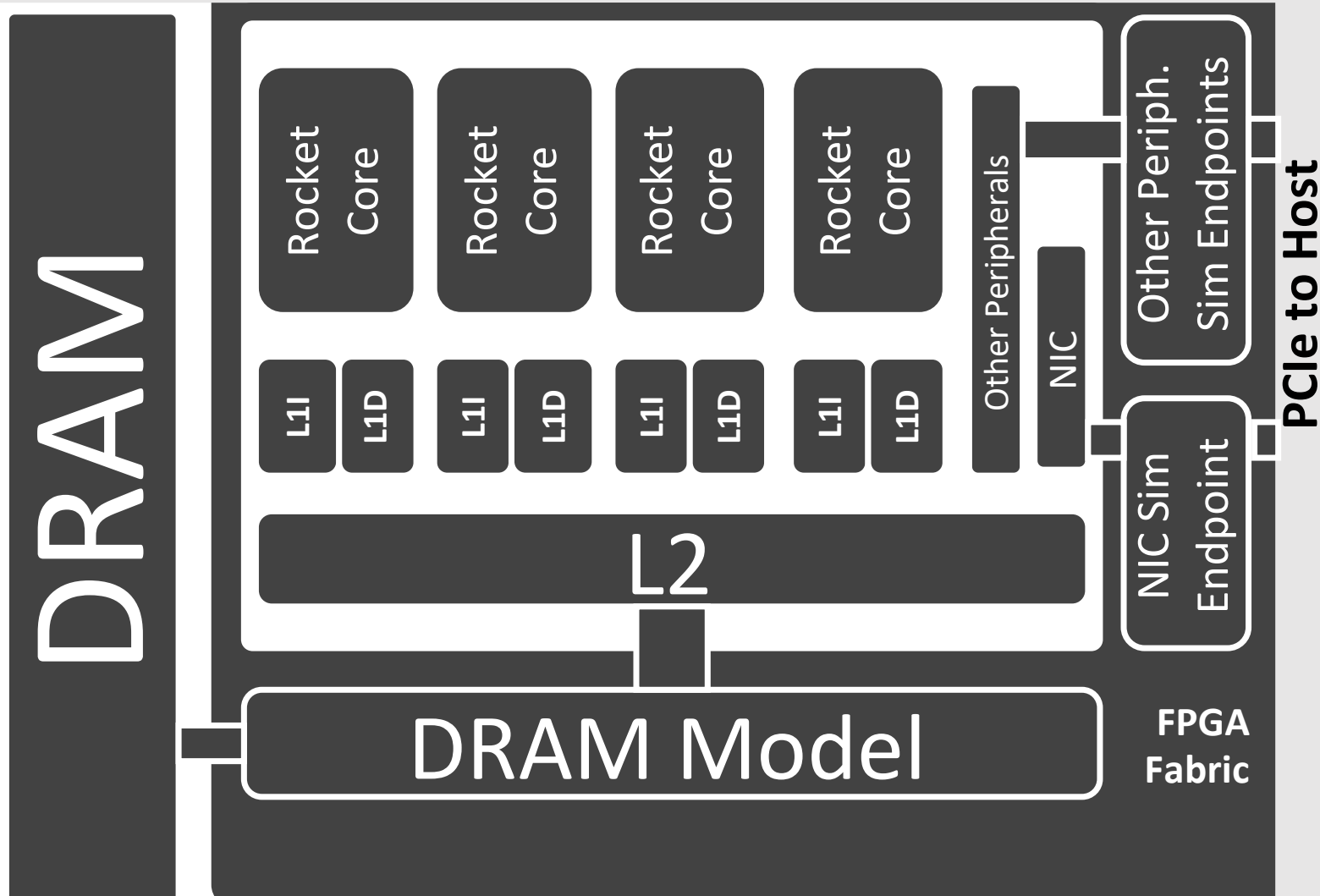
- < ¼ of an FPGA

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.

- < ¼ of an FPGA
- ¼ Mem Chans

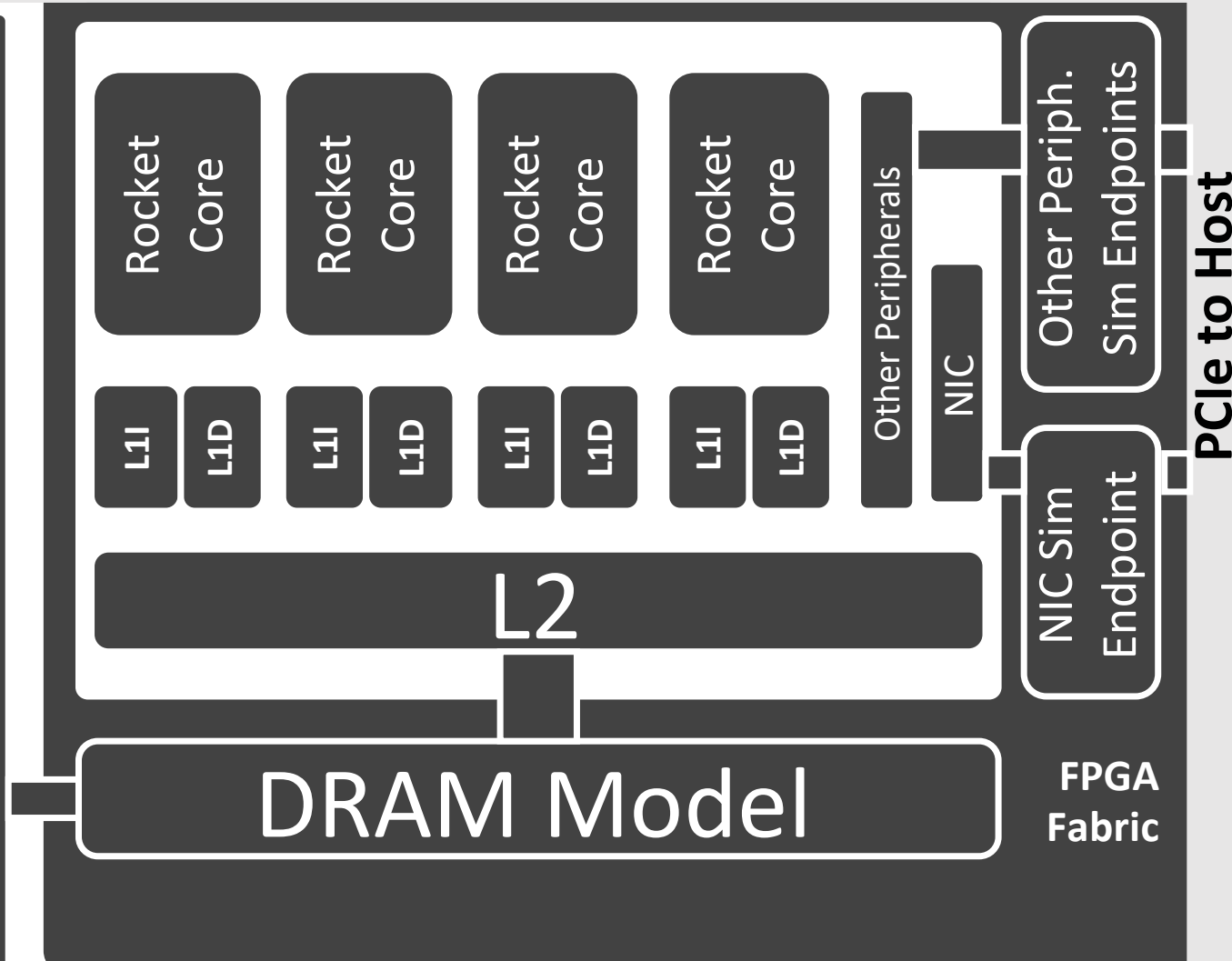
Sim Rate

- ~150 MHz
- ~40 MHz (netw)



Step 2: FPGA Simulation of one server blade

DRAM



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.

- < ¼ of an FPGA
- ¼ Mem Chans

Sim Rate

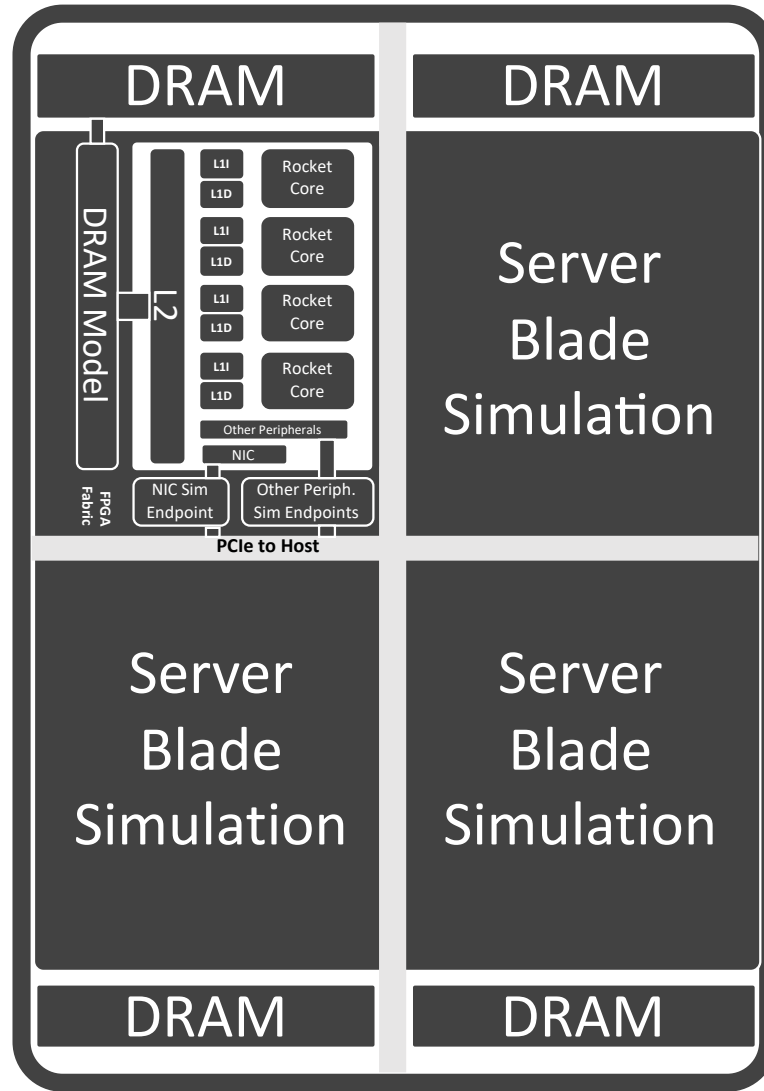
- ~150 MHz
- ~40 MHz (netw)



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

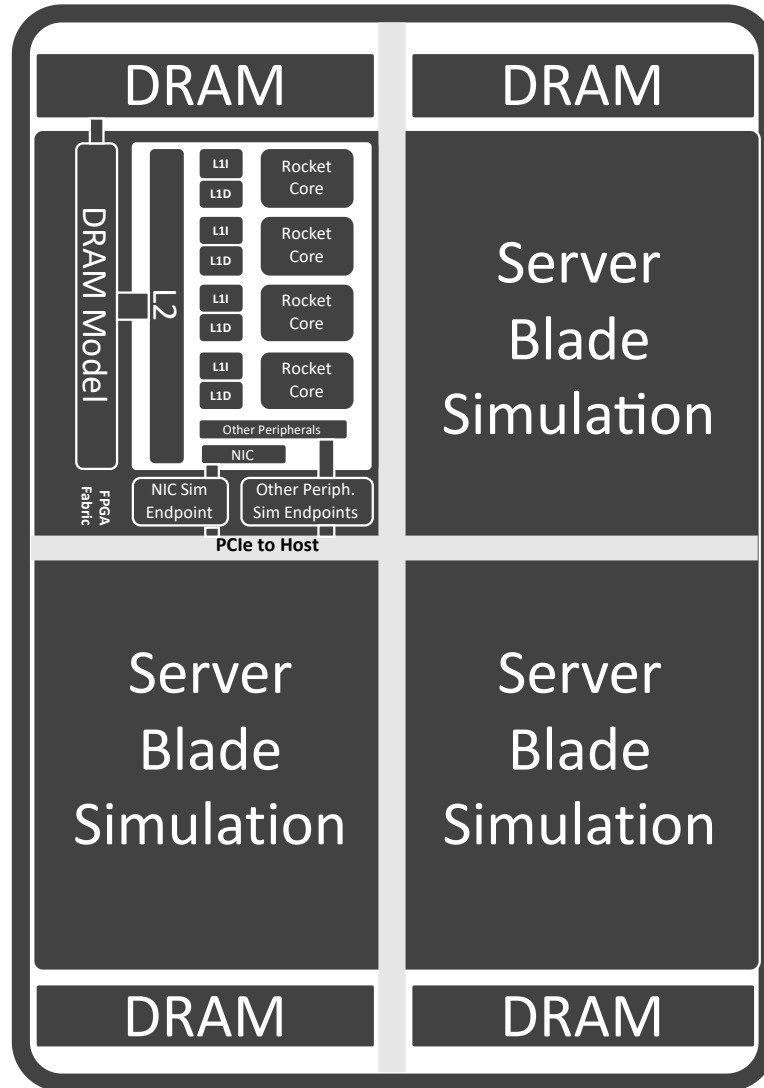
Sim Rate

- ~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

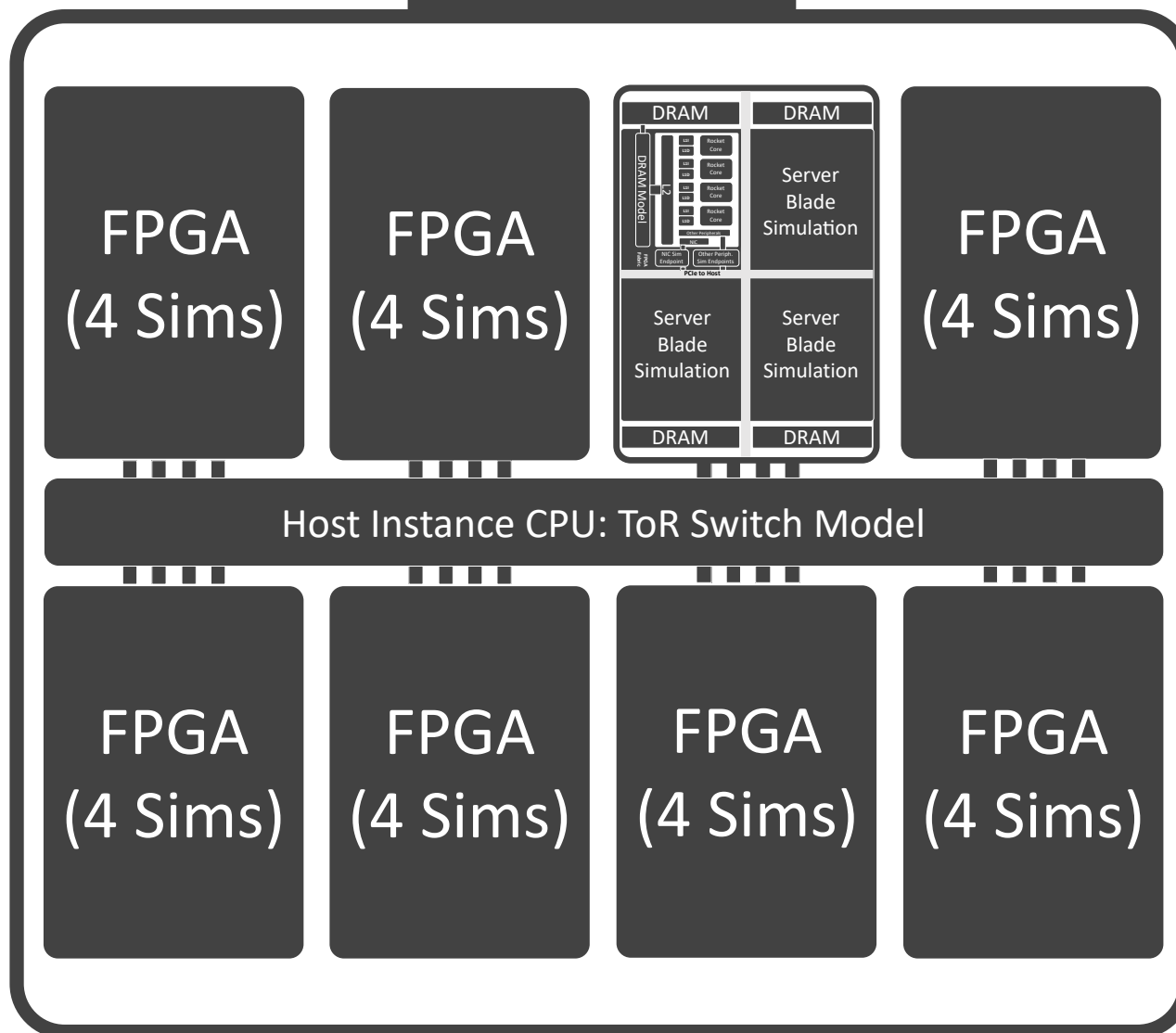
- ~14.3 MHz (netw)



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

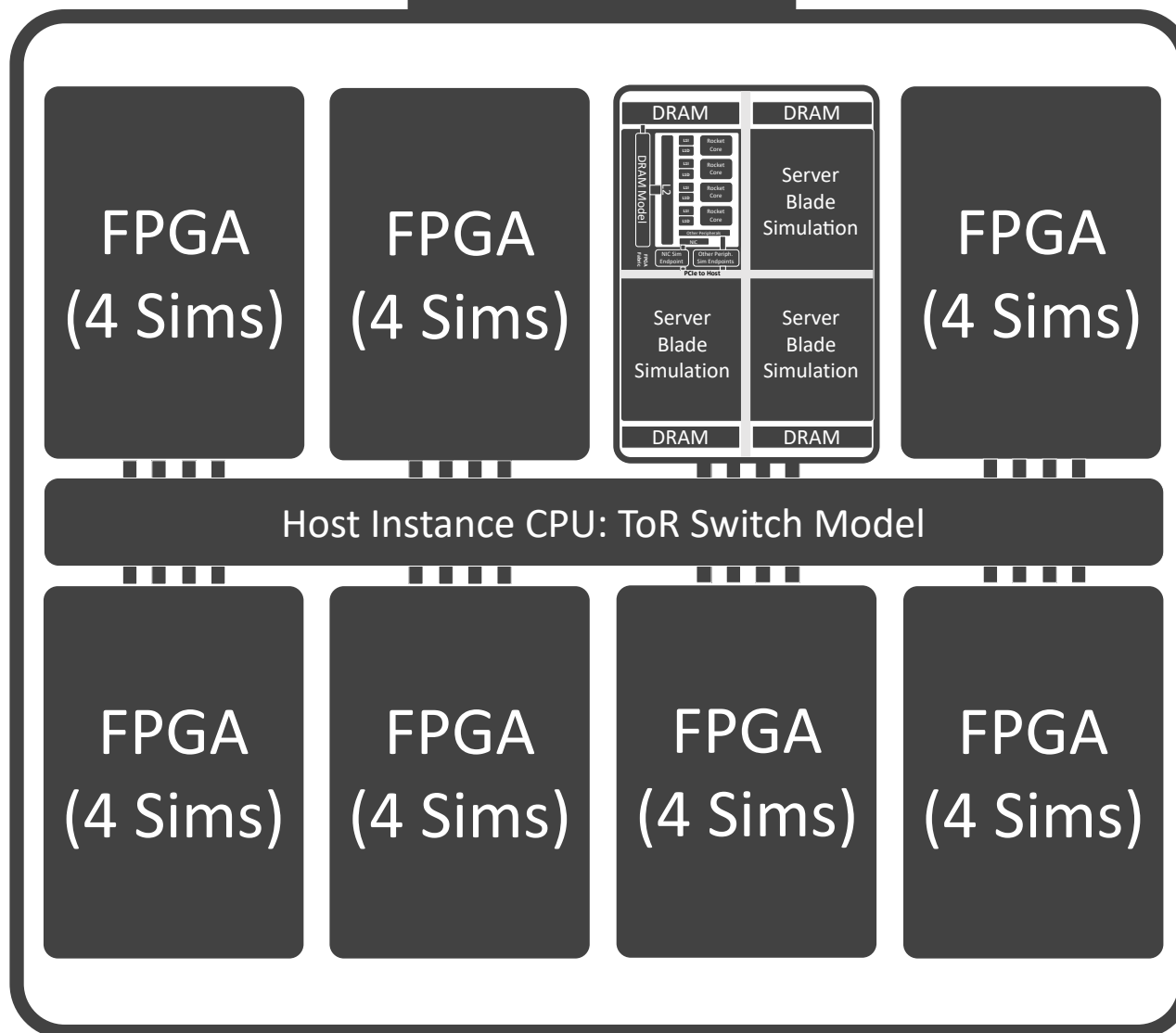
- ~10.7 MHz (netw)



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.

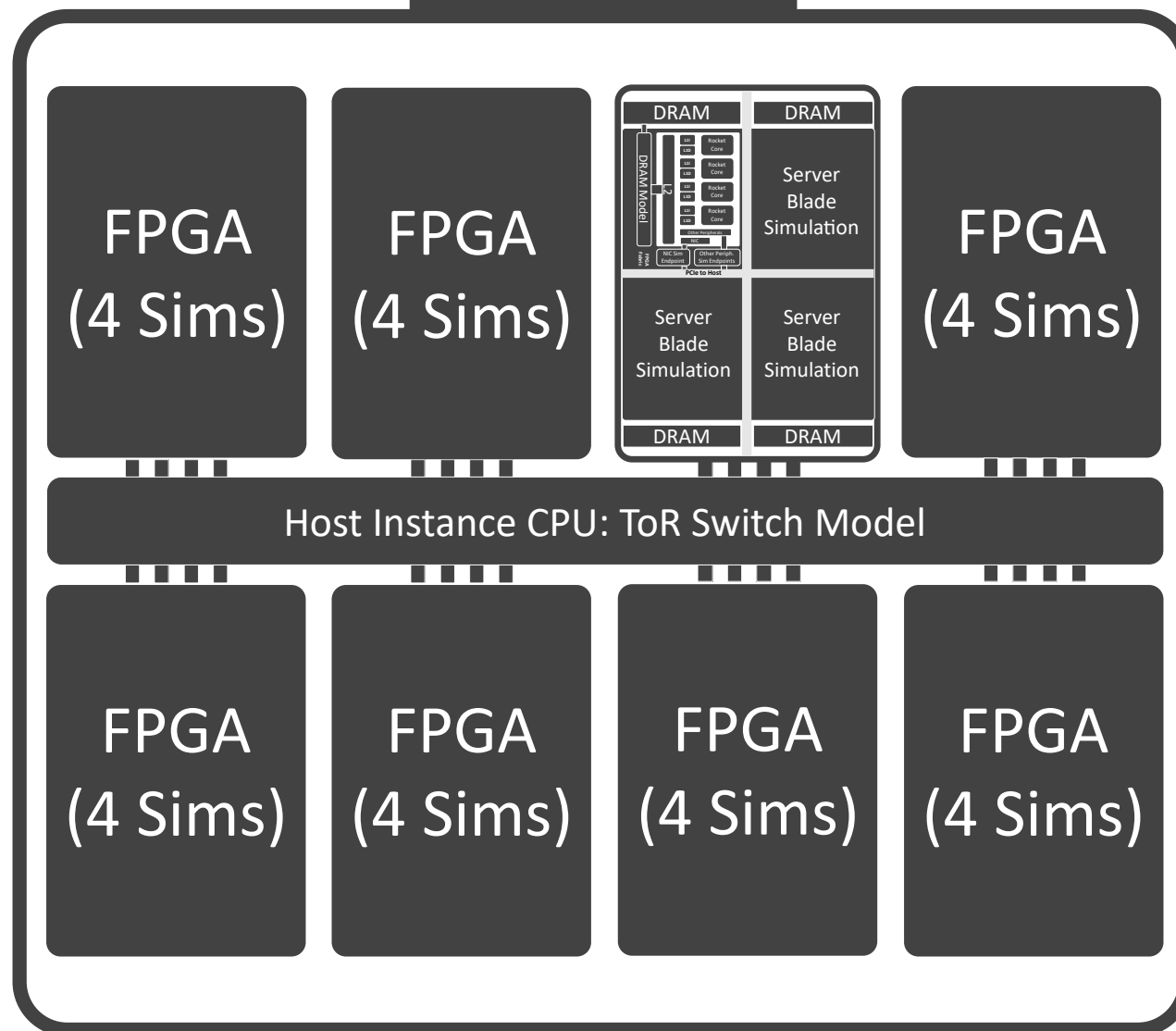
- 8 FPGAs =
- 1x f1.16xlarge

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

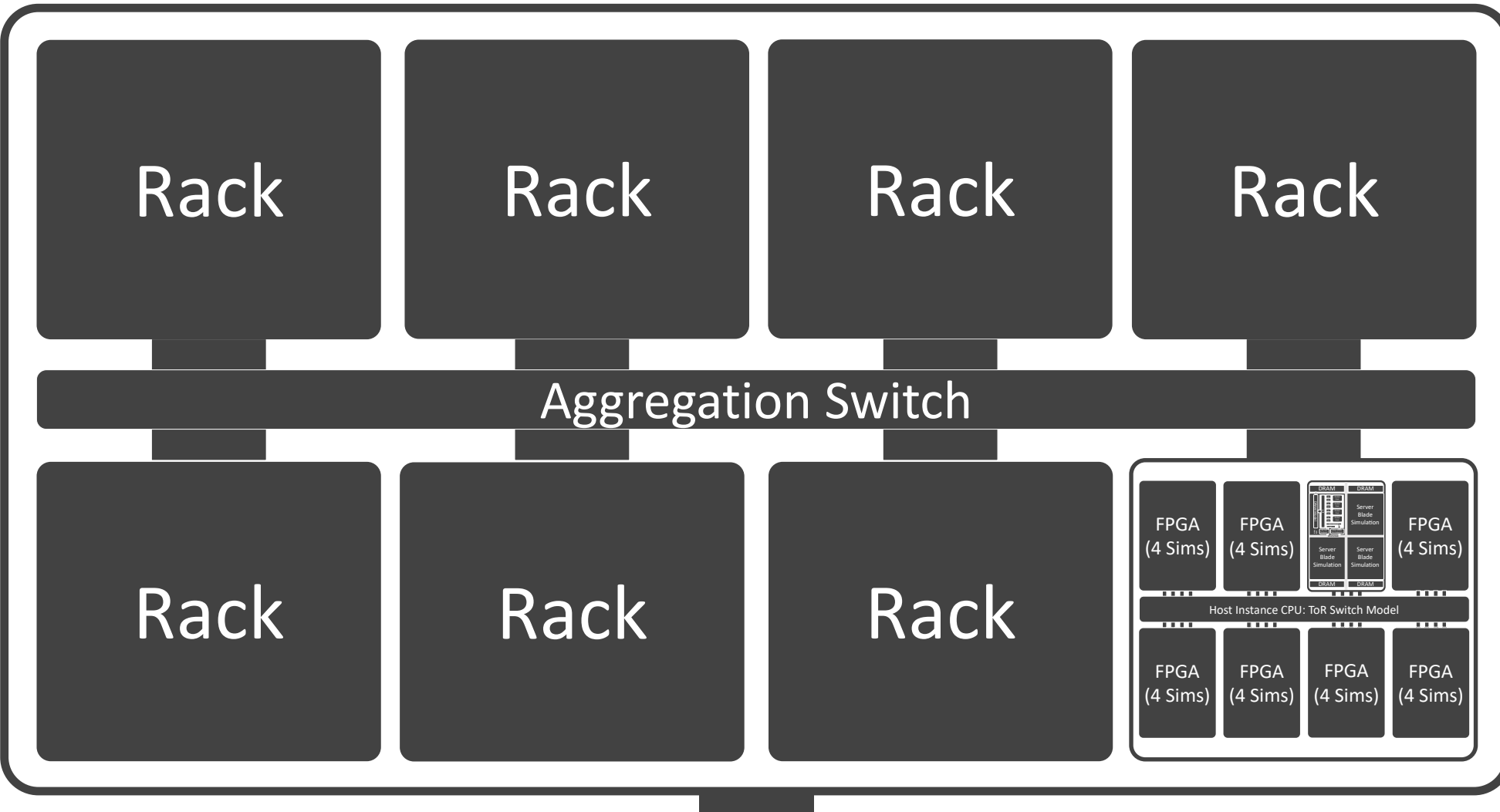
Resource Util.

- 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

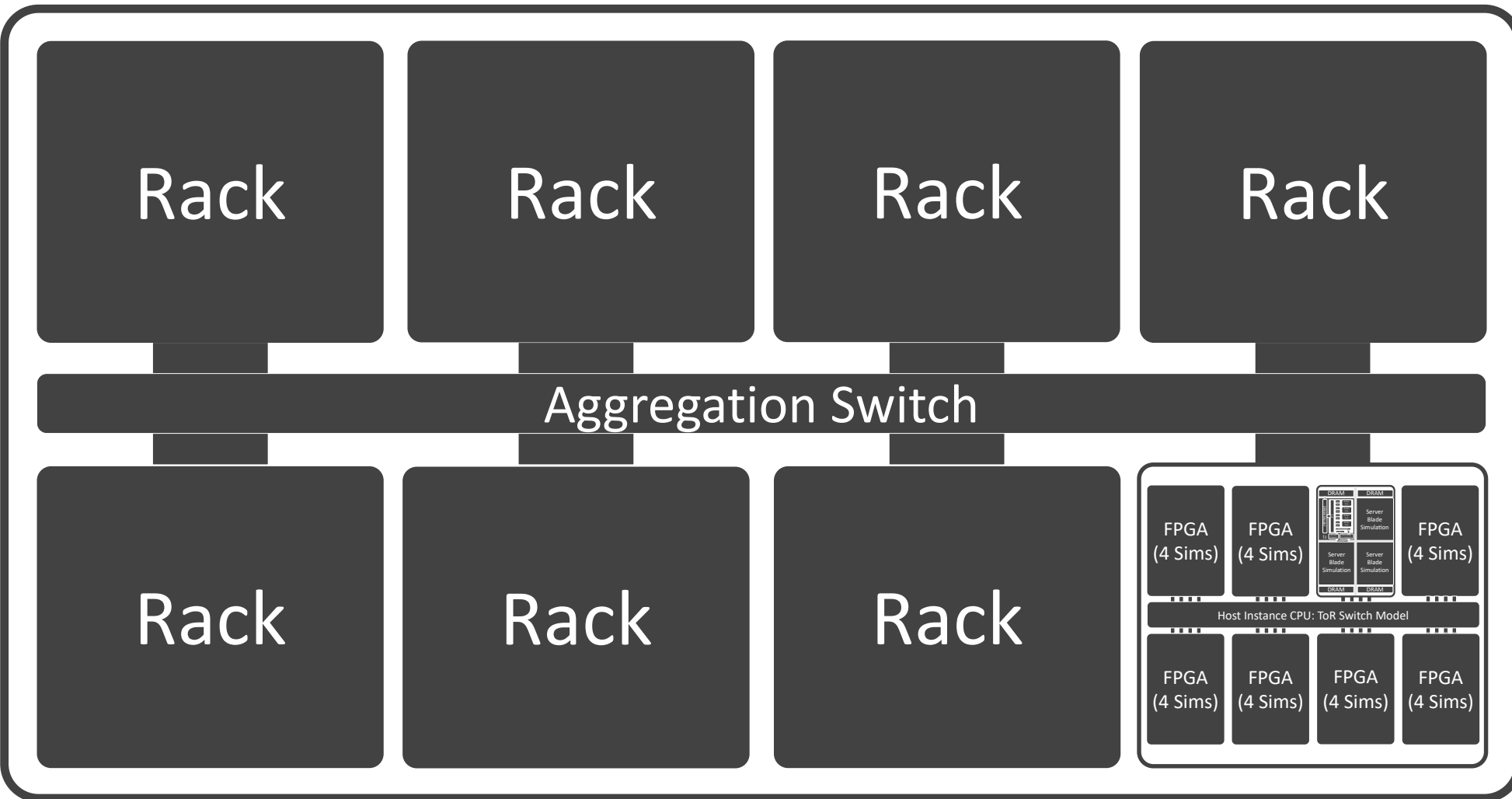
Resource Util.

- 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.

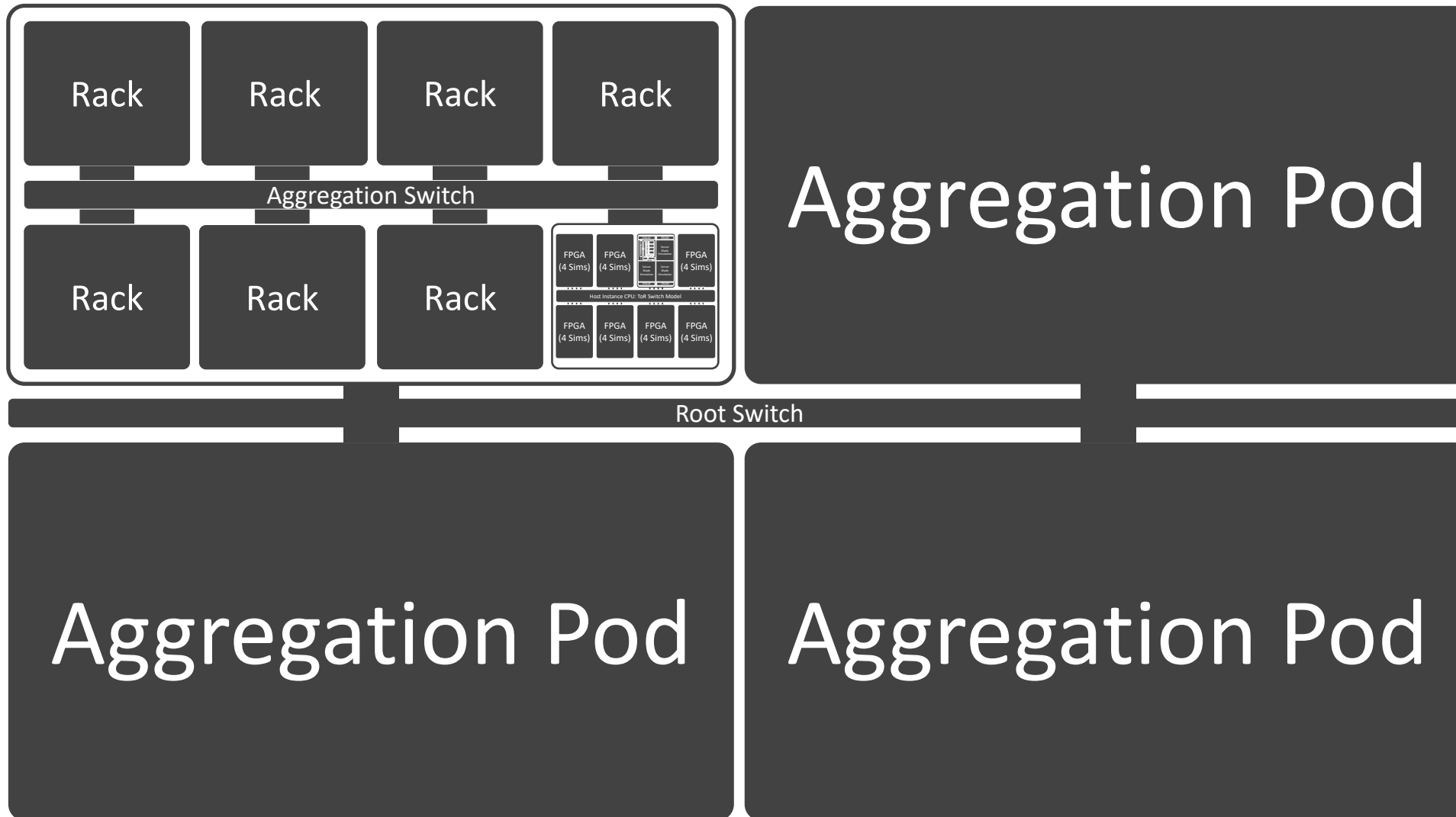
- 64 FPGAs =
- 8x f1.16xlarge
- 1x m4.16xlarge

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
- 16 TB DDR3
- 16 ToRs, 4 Aggr, 1
- 10 Gb/s, 2us
- Source Util.
- 250 FPGAs =
- 32x f1.16xlarge
- 5x m4.16xlarge
- Sim Rate
- ~6.6 MHz (netw)



Join the FireSim Community!

- Companies publicly announced using FireSim
 - Esperanto Maxis ET
 - Intersil IntenCore
 - SiFive validation paper @ VLSI'20
- Chipyard integration
- Projects with public FireSim support
 - Rocket Chip, BOOM
 - Hwacha Vector Accelerator
 - Keystone Secure Enclave
 - NVIDIA Deep Learning Accelerator (NVDLA)
 - <https://github.com/nvdla/firesim-nvdla>
 - <https://devblogs.nvidia.com/nvdla/>
 - BOOM Spectre replication/mitigation
 - More in-progress! PR yours!
- Many academic users
 - 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)
 - See FireSim website for more!
- Education
 - Berkeley CS152/252
 - CCC/RV-Summit tutorials
 - MICRO 2019 full-day tutorial
- More than 100 mailing list members
- More than 250 unique cloners per week

*FireSim ISCA'18 paper selected as an **IEEE Micro Top Pick of 2018 Arch. Confs***

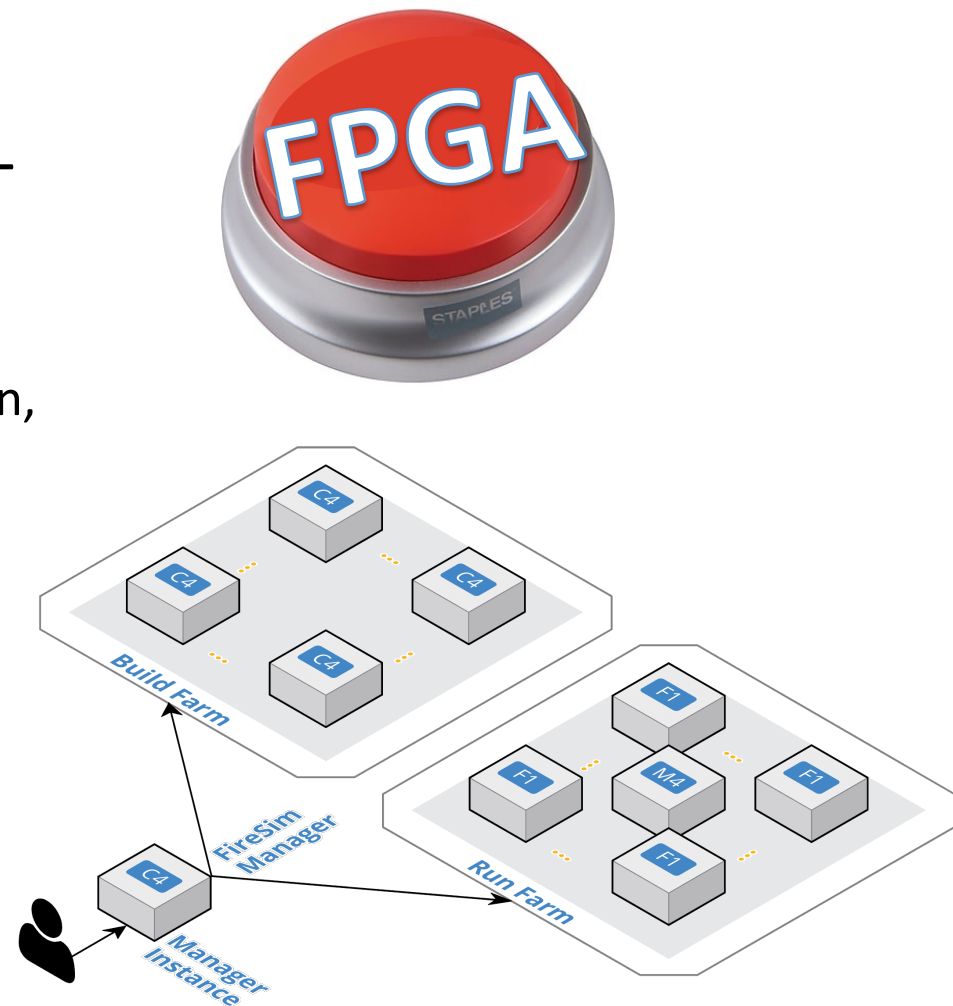
*and as the **CACM Research Highlights Nominee from ISCA'18***



Wrapping-up: 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



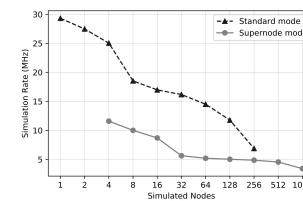
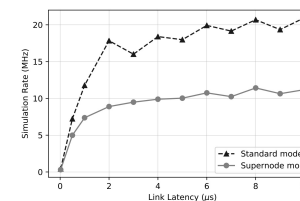
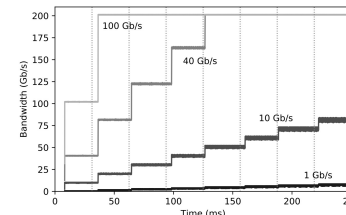
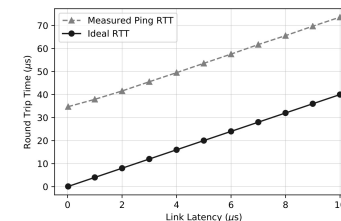
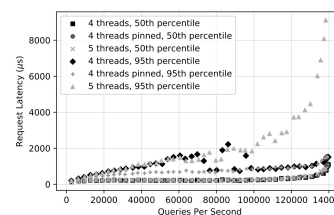
Wrapping-up: Productive Open-Source FPGA Simulation



- 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

- 130+ pages of documentation:
<https://docs.firesim.im>
- AWS provides grants for researchers:
<https://aws.amazon.com/grants/>

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





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, and by DARPA, Award Number HR0011-12-2-0016. Research was also partially funded by ADEPT Lab industrial sponsors and affiliates Intel, Apple, Futurewei, Google, and Seagate, 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.