FireSim

Scalable FPGA-accelerated Cycle-Accurate Hardware Simulation in the Cloud

Introduction

https://fires.im

@firesimproject

Chisel Community Conference 2018 Intensive

Speakers: **Sagar Karandikar**, David Biancolin, Alon Amid

Berkeley Architecture Research

What is this “Intensive” about?

- We’re going to keep things ***very*** practical today, treat this as a tutorial
- For more of the research background, see our papers/previous talks
- We’re going to cram in a lot of info, but feel free to interrupt us with questions
  - We’ll put up the full slide deck online anyway

Today’s Agenda

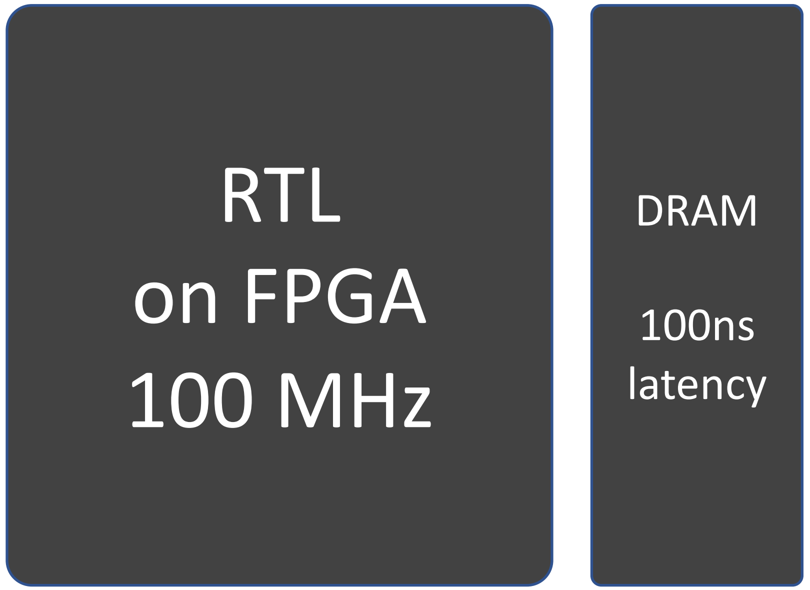
- Intro, what is FireSim capable of?
  - (Sagar, 10 mins)
- Building/Deploying FireSim Simulations
  - (Sagar, 15 mins)
- Using FireSim to simulate your own custom HW
  - (David, 15 mins)
- Testing and Debugging your design in FireSim
  - (Alon, 15 mins)
- Community/Contributing/Conclusion
  - (David, 10 mins)

What can FireSim do?

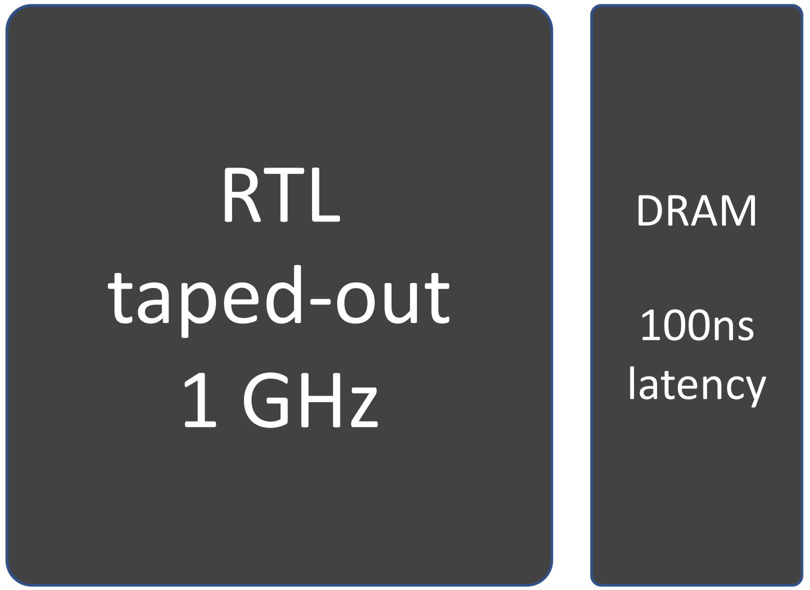
- Model hardware at scale, cycle-accurately:
  - CPUs down to microarchitecture (automatically transformed from Chisel RTL)
  - Fast networks, switches (SW models)
  - Novel accelerators (also transformed from Chisel)
- Run real software:
  - Real OS, networking stack (Linux)
  - Real frameworks/applications (not microbenchmarks)
- Be productive/usable:
  - Run on a commodity platform (Amazon EC2 F1)
  - Want to encourage collaboration between systems, architecture: *real* HW/SW co-design research

Why do we want an FPGA-based *simulator*?

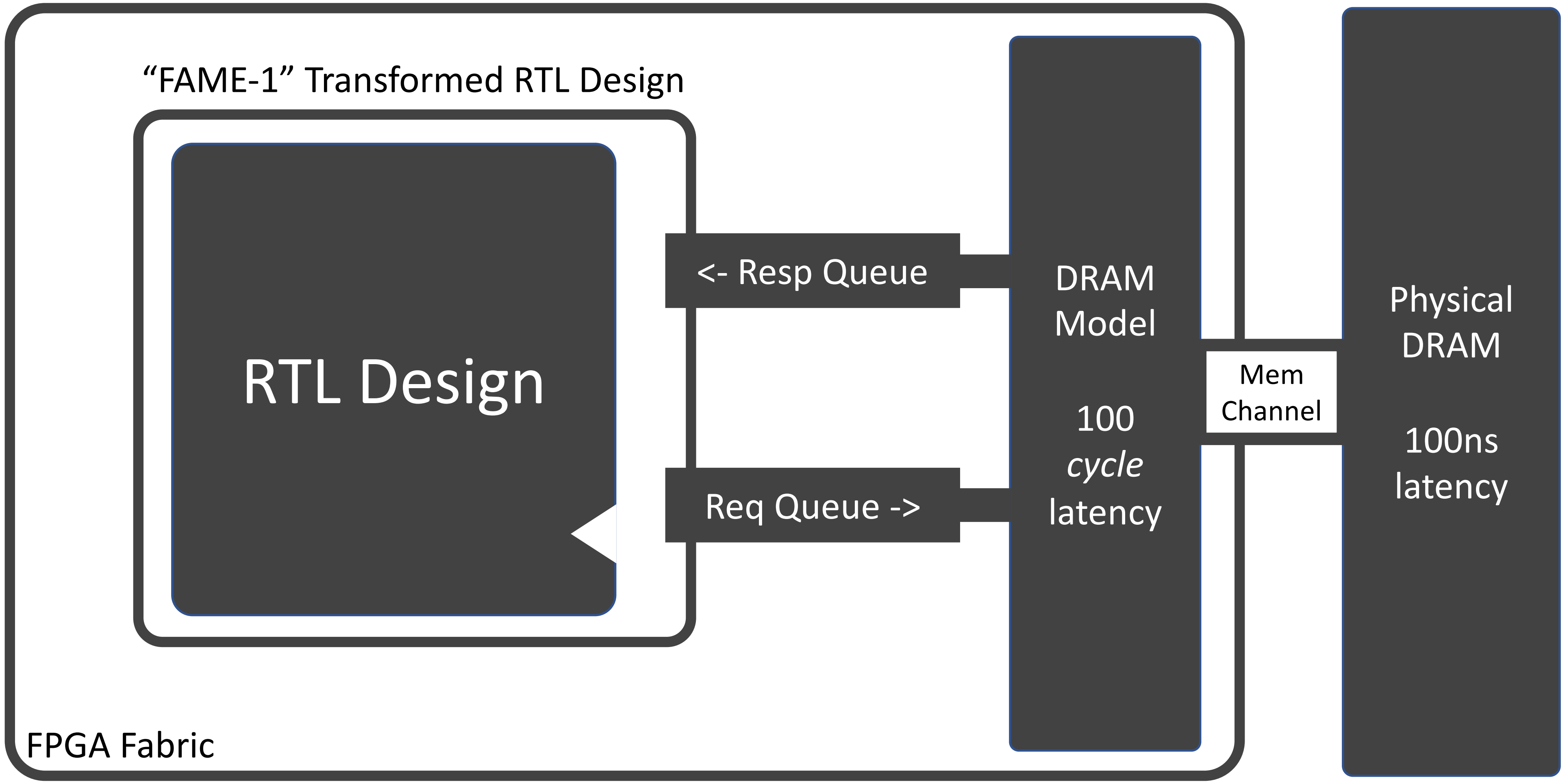
FPGA Emulation/Prototyping
SoC sees 10 cycle DRAM latency



Taped-out Design
SoC sees 100 cycle DRAM latency



How do we fix this? FAME-1 Xform in FIRRTL w/MIDAS



Comparing existing HW “simulation” systems

- Taping-out excels at:
  - Modeling reality: “single source of truth”
  - Scalability
- Hardware-accelerated simulators excel at:
  - Simulation rate
  - Ability to run real workloads (as fn. of sim rate)
- Software-based simulators excel at:
  - Ease-of-use
  - Ease-of-rebuild (time-to-first-cycle)
  - Commodity host platform
  - Cost
  - Introspection

Prior work on HW-accelerated simulators: DIABLO

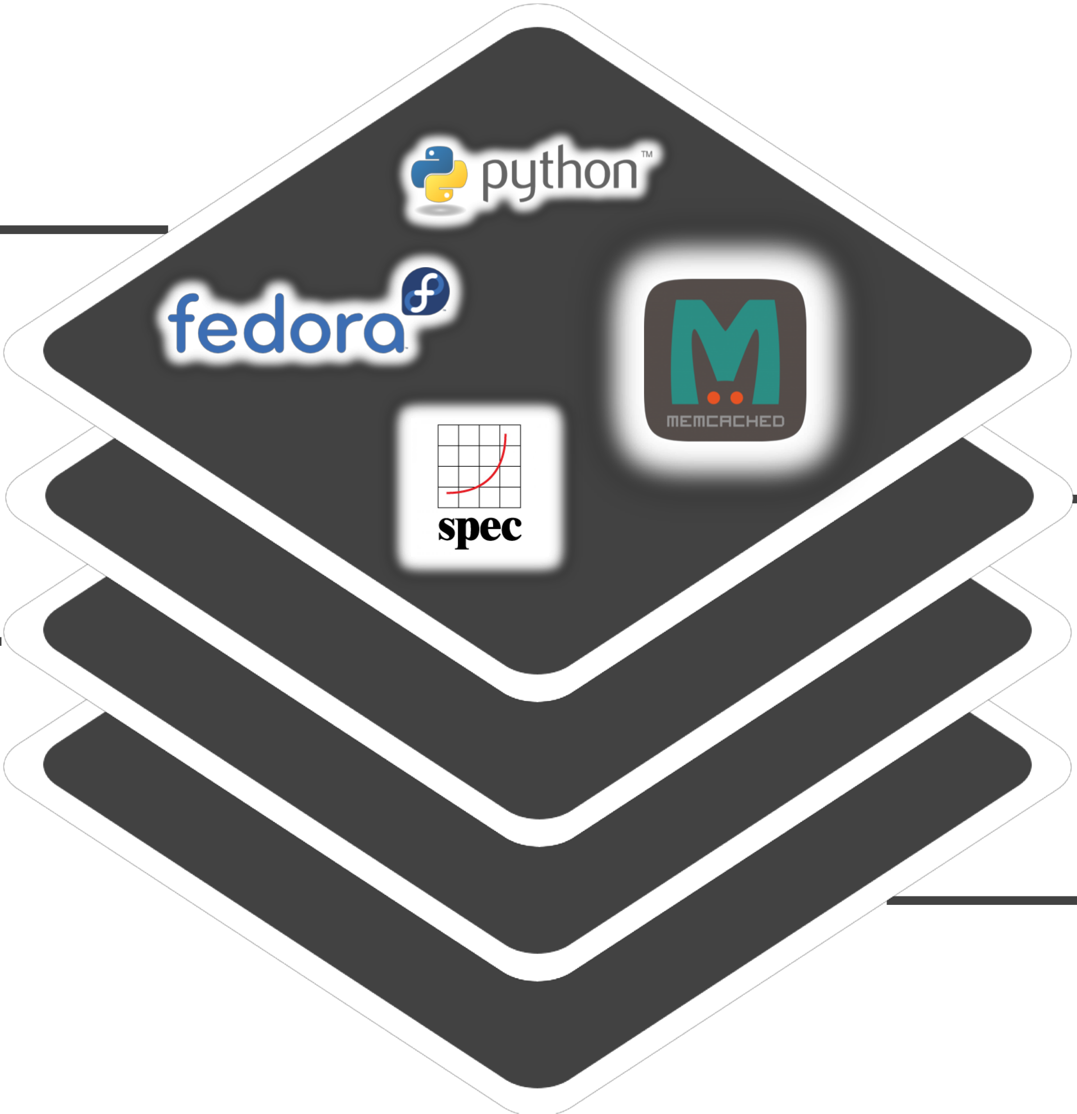
- DIABLO, *ASPLOS'15* [4]:
  - Simulated 3072 servers, 96 ToRs at ~2.7 MHz
  - Booted Linux, ran apps like Memcached
  - An example of the many projects from RAMP
- Need to hand-write abstract RTL models
  - Harder than writing "tapeout-ready" RTL
  - Need to validate against real HW
- Tied to an expensive custom host-platform
  - $100k+ host platform, custom built

DIABLO Prototype

Useful trends throughout the architect's stack

Open ISA

High-Productivity Hardware Design Language w/IR



Open, Silicon-Proven SoC Implementations

FPGAs in the Cloud

Amazon EC2 F1 Instances

Run Customizable FPGAs in the AWS Cloud

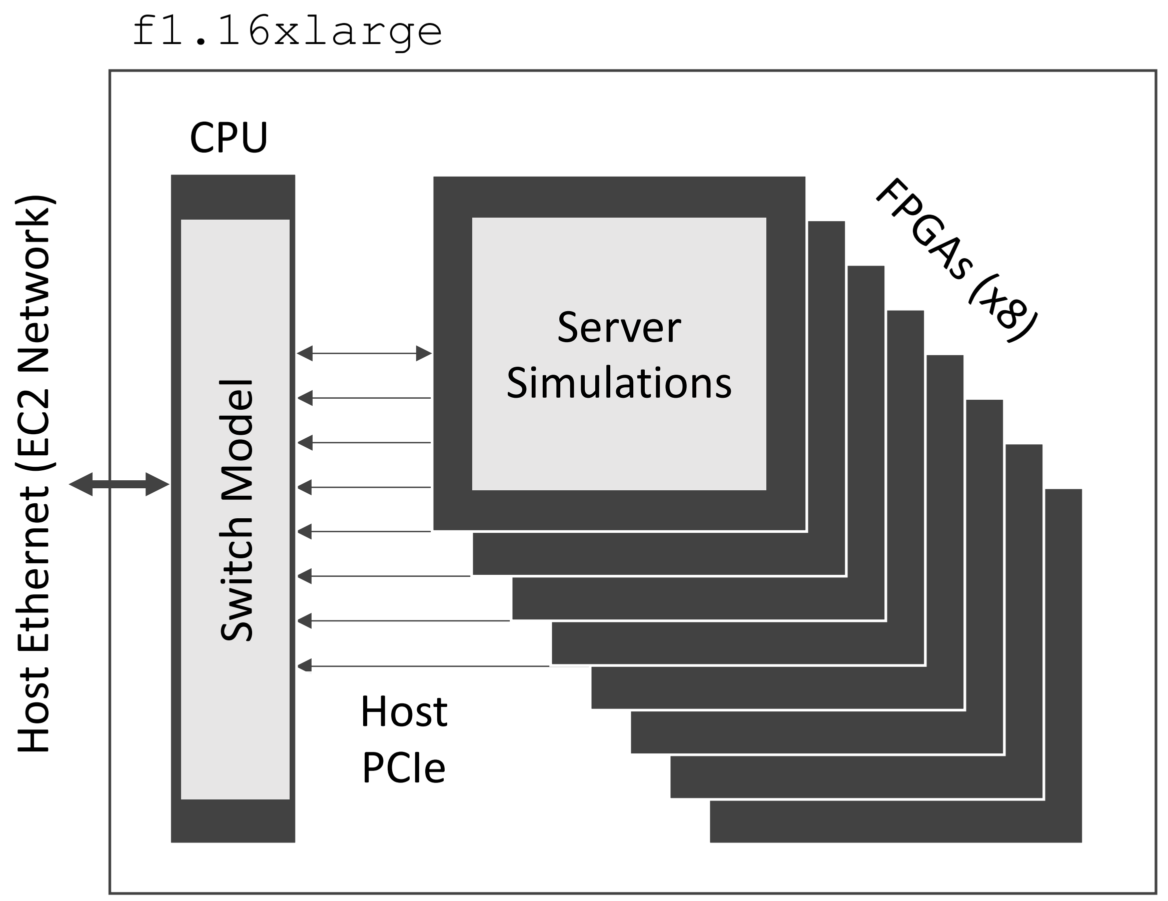
FireSim at a high-level

Server Simulations

- Or your own custom hardware
- Good fit for the FPGA
- We have tapeout-proven RTL: automatically FAME-1 transform

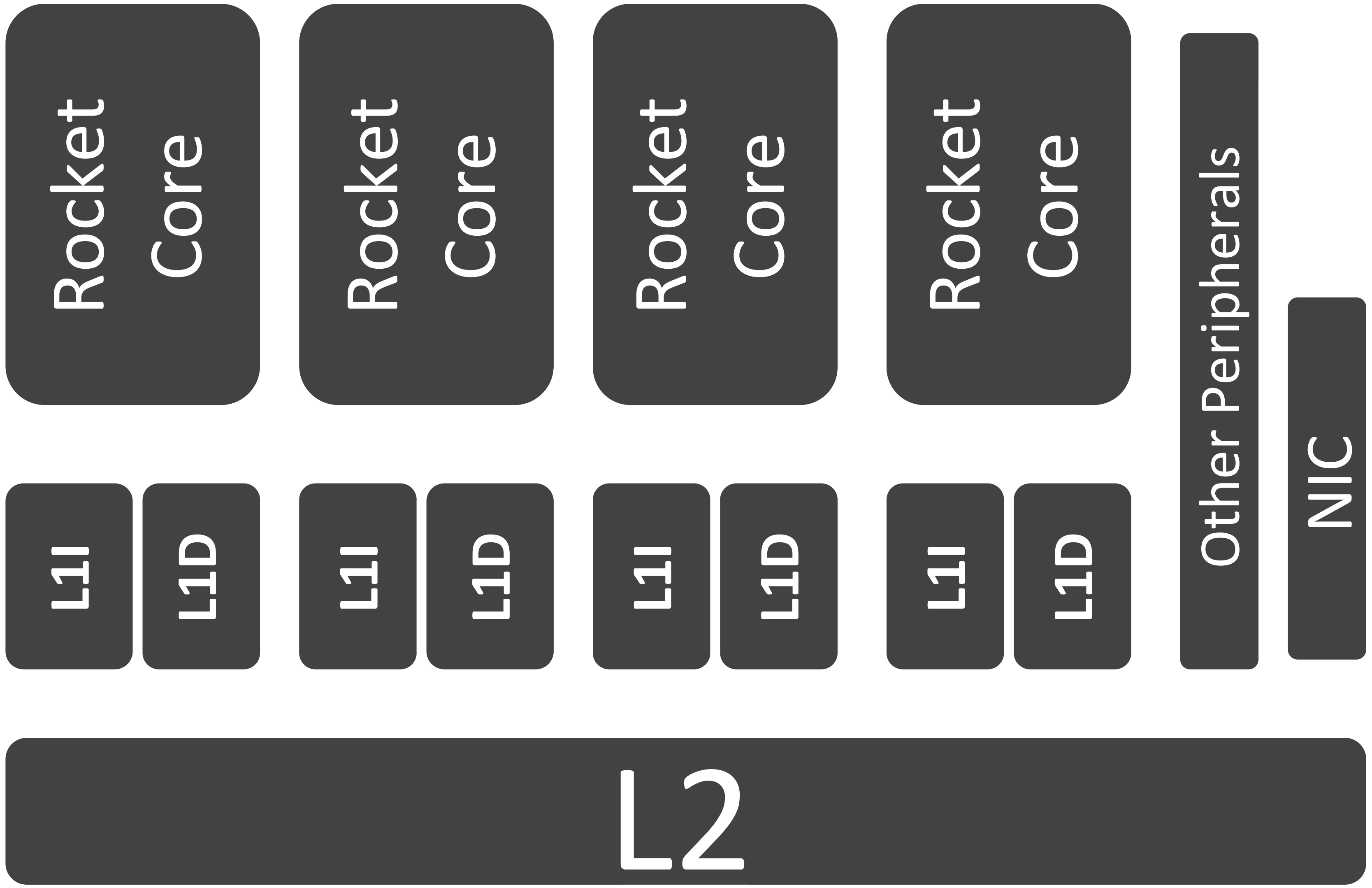
Network simulation

- Or your own custom SW models
- Little parallelism in switch models (e.g. a thread per port)
- Need to coordinate all of our distributed server simulations
- So use CPUs + host network



Now, let’s build a datacenter-scale FireSim simulation!

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

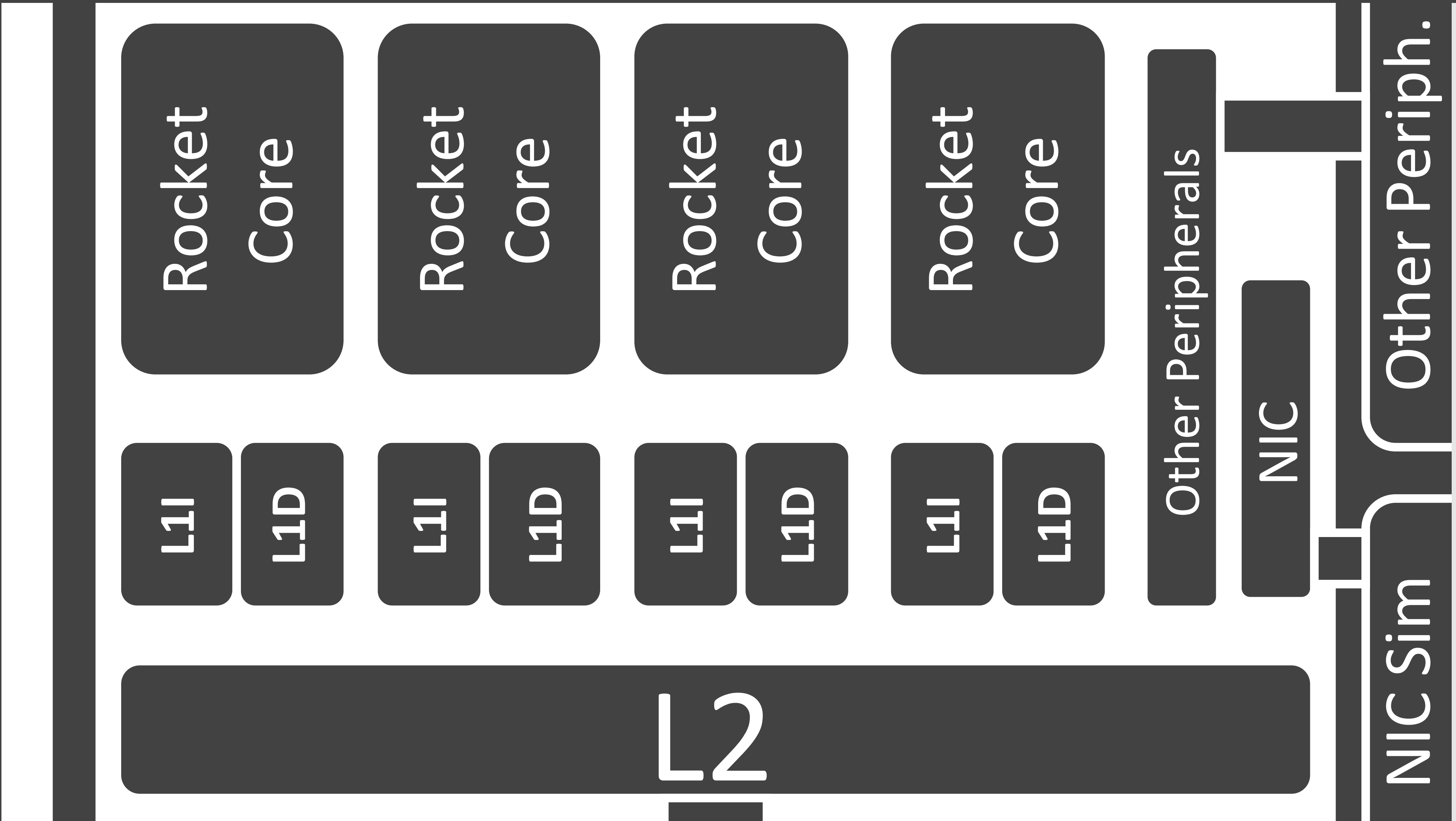
Resource Util.

- < 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

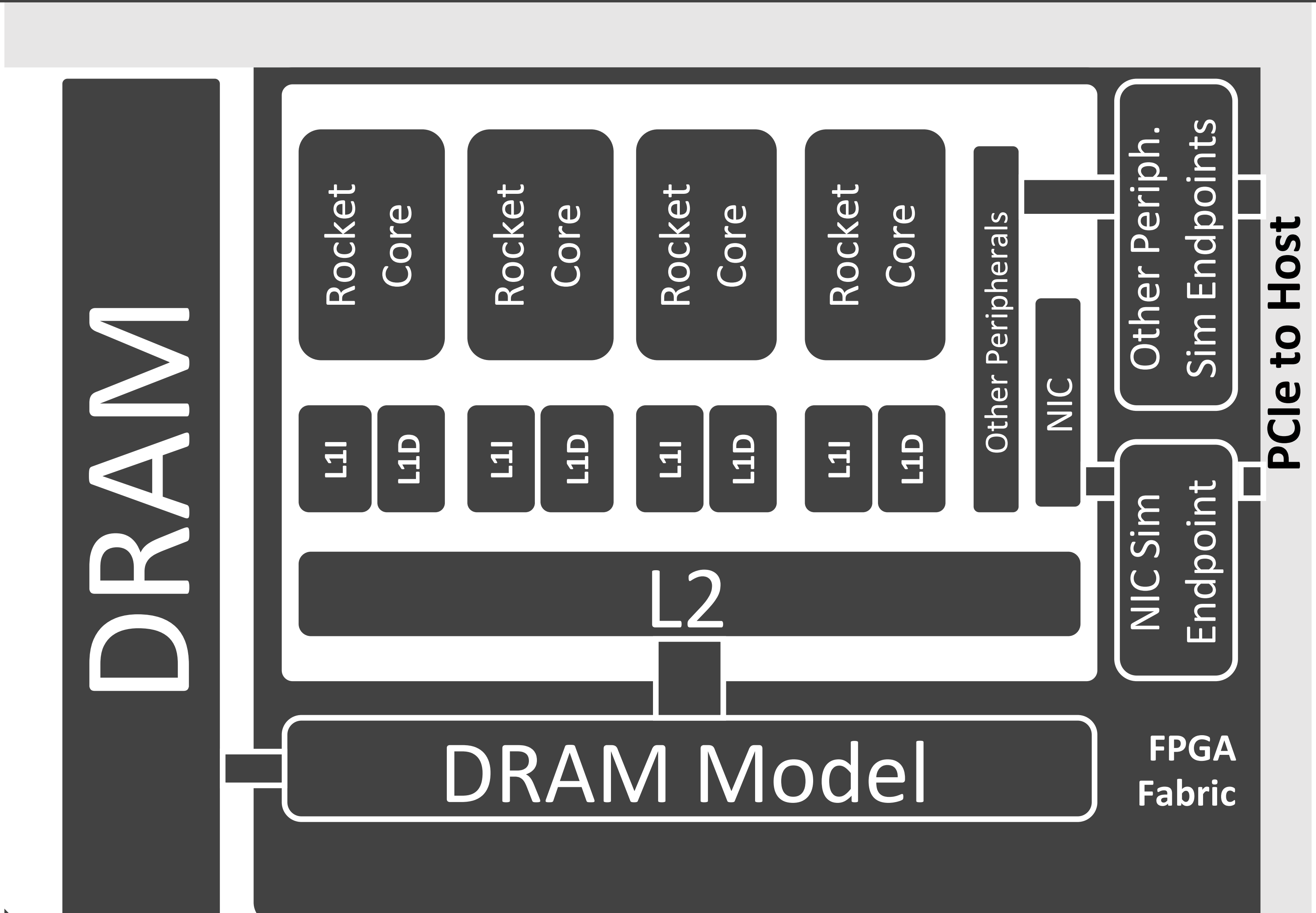
Resource Util.

- < 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

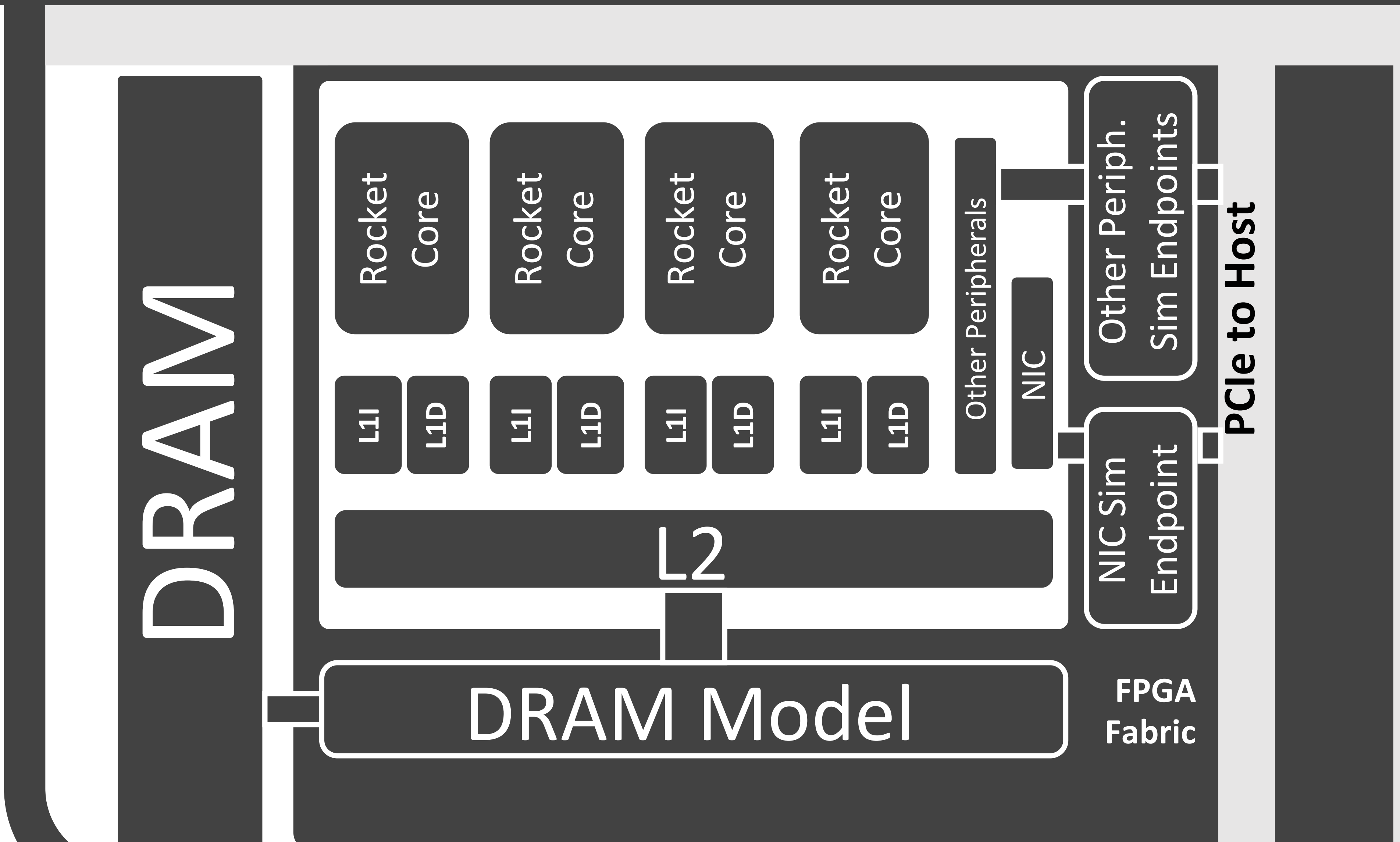
Resource Util.

- < ¼ of an FPGA
- ¼ Mem Chans

Sim Rate

- ~150 MHz
- ~40 MHz (netw)

Step 2: FPGA Simulation of one server blade



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

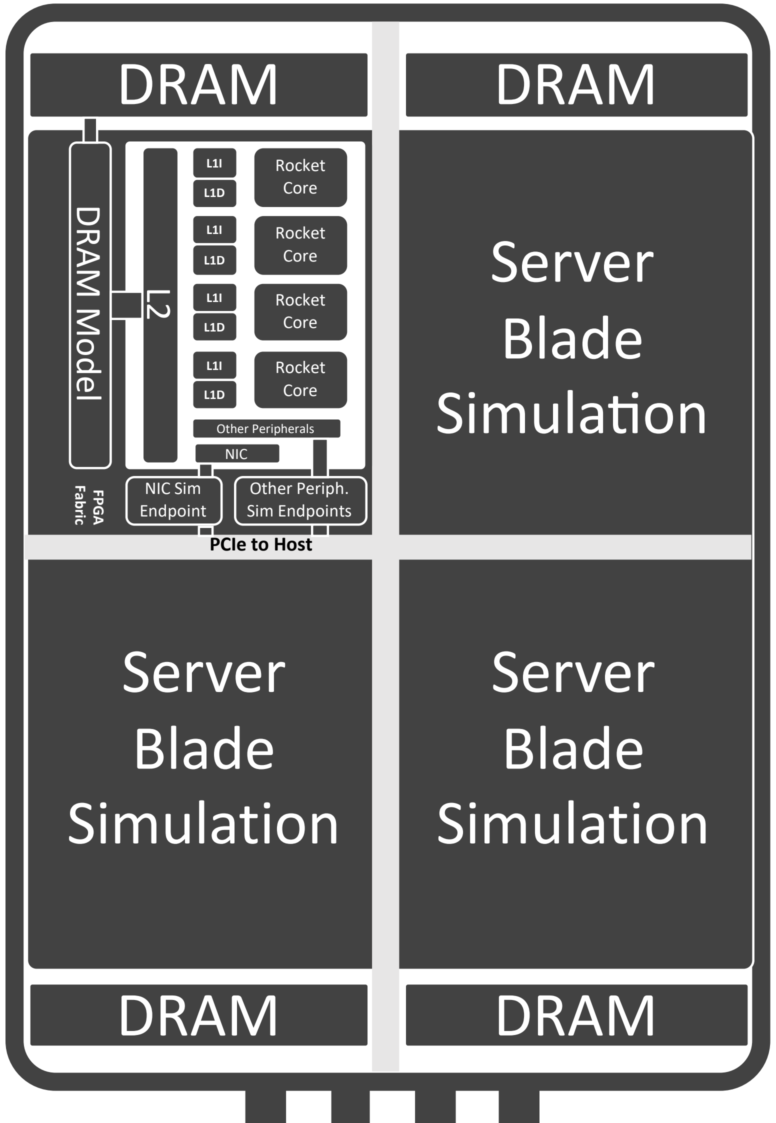
Sim Rate

- ~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)

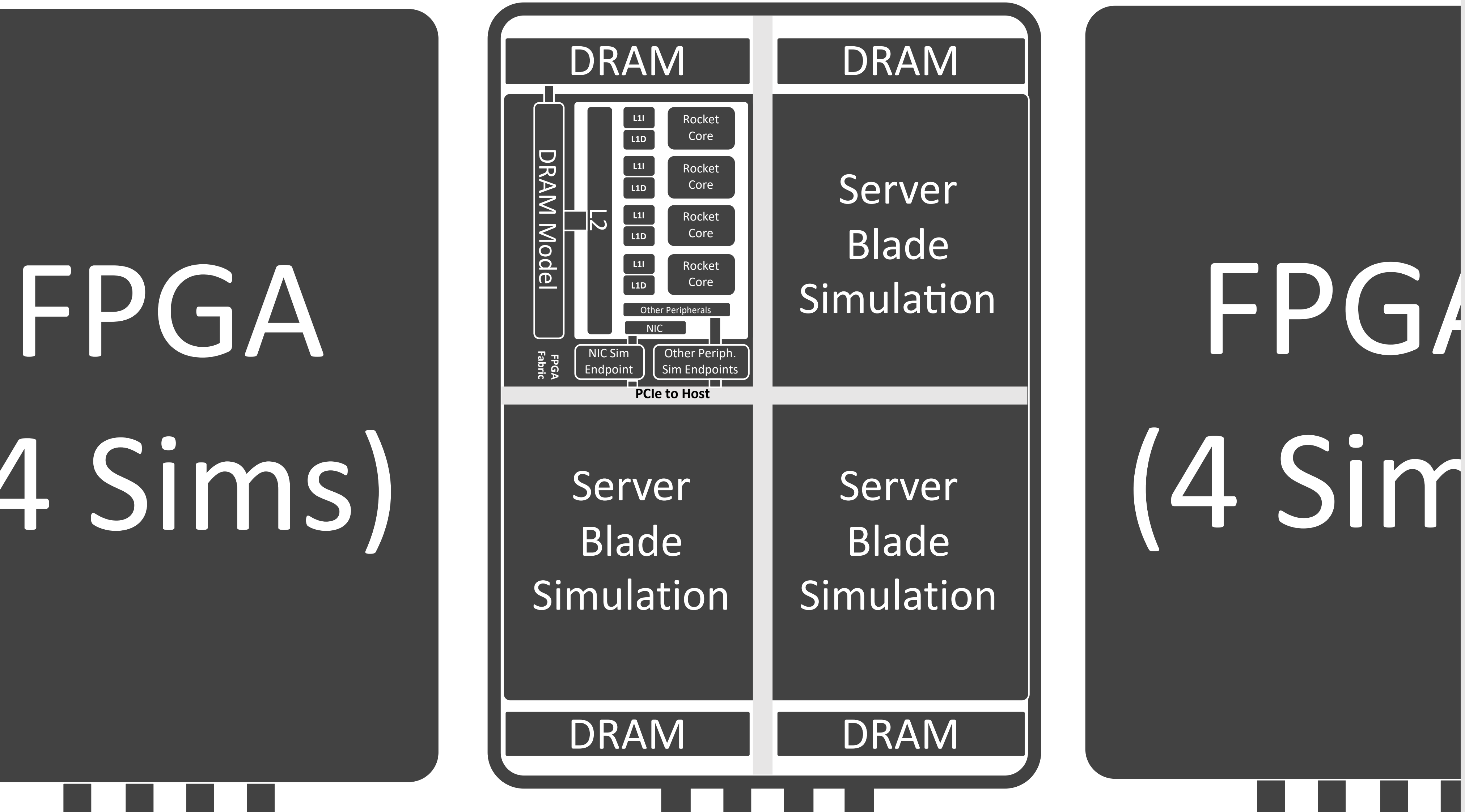


Modeled System
- 4 Server Blades
- 16 Cores
- 64 GB DDR3

Resource Util.
- < 1 FPGA
- 4/4 Mem Chans

Sim Rate
- ~14.3 MHz (netw)

Step 3: FPGA Simulation of 4 server blades



Modeled System
- 4 Server Blades
- 16 Cores
- 64 GB DDR3

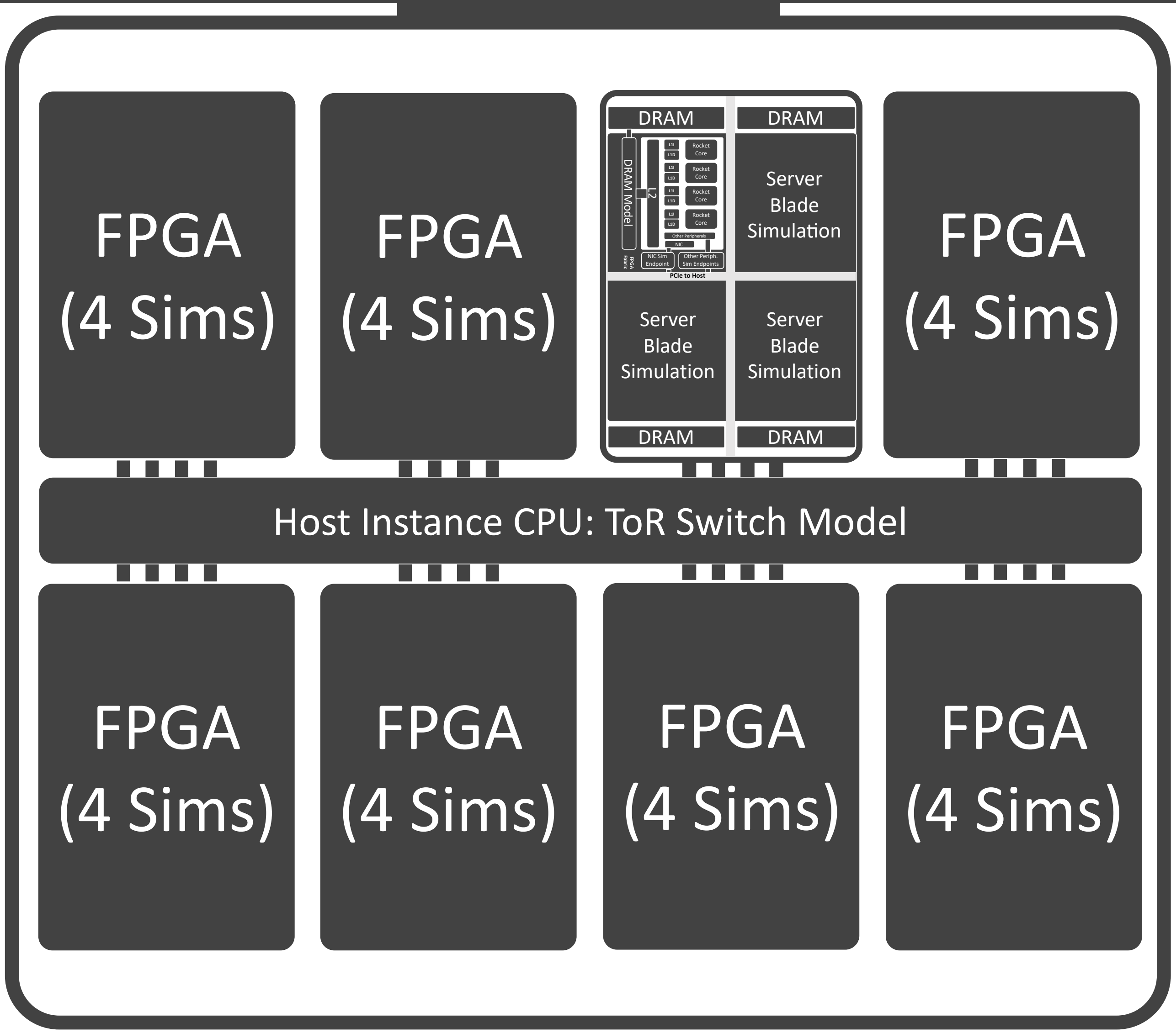
Resource Util.
- < 1 FPGA
- 4/4 Mem Chans

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

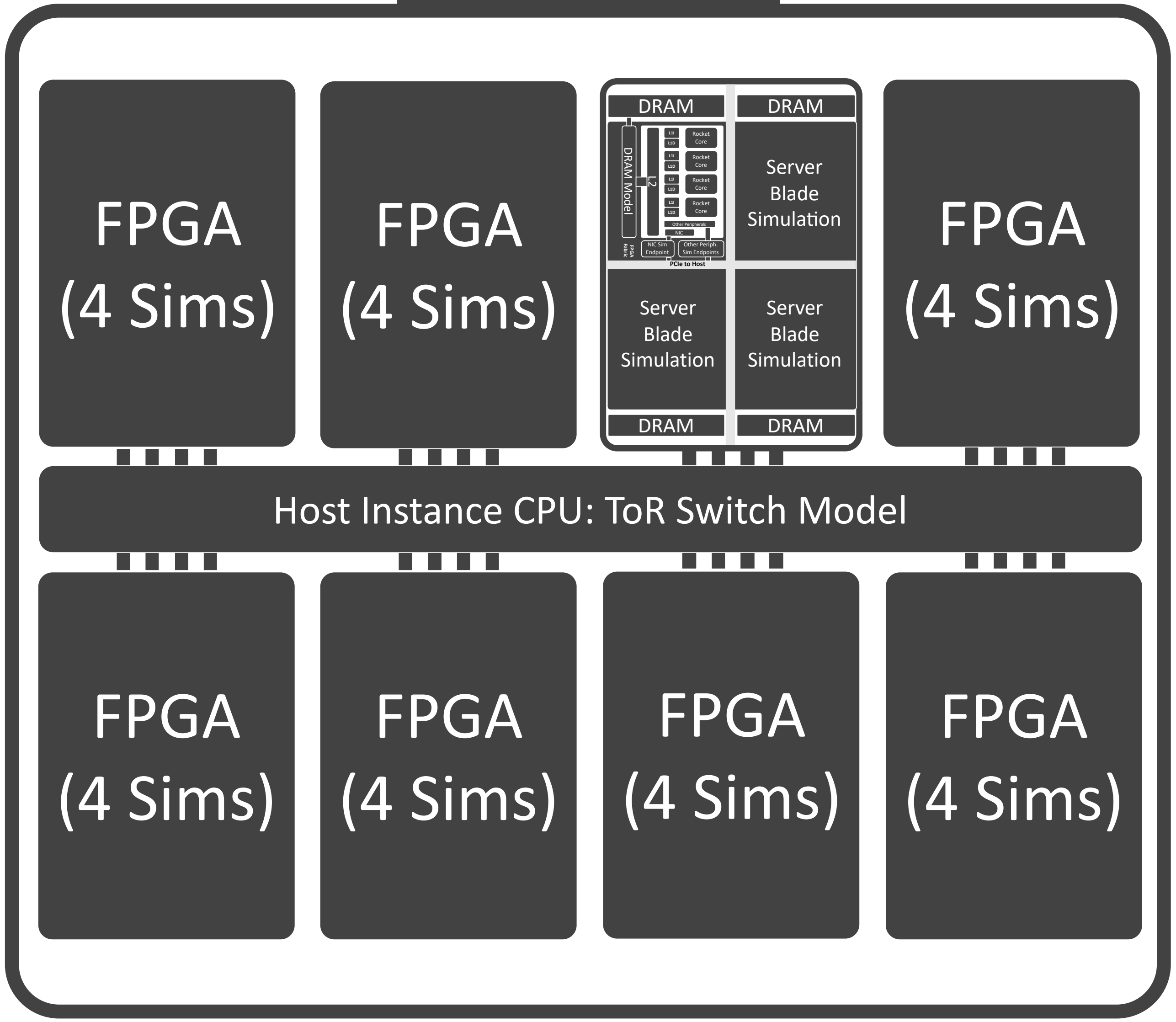
Sim Rate

- ~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

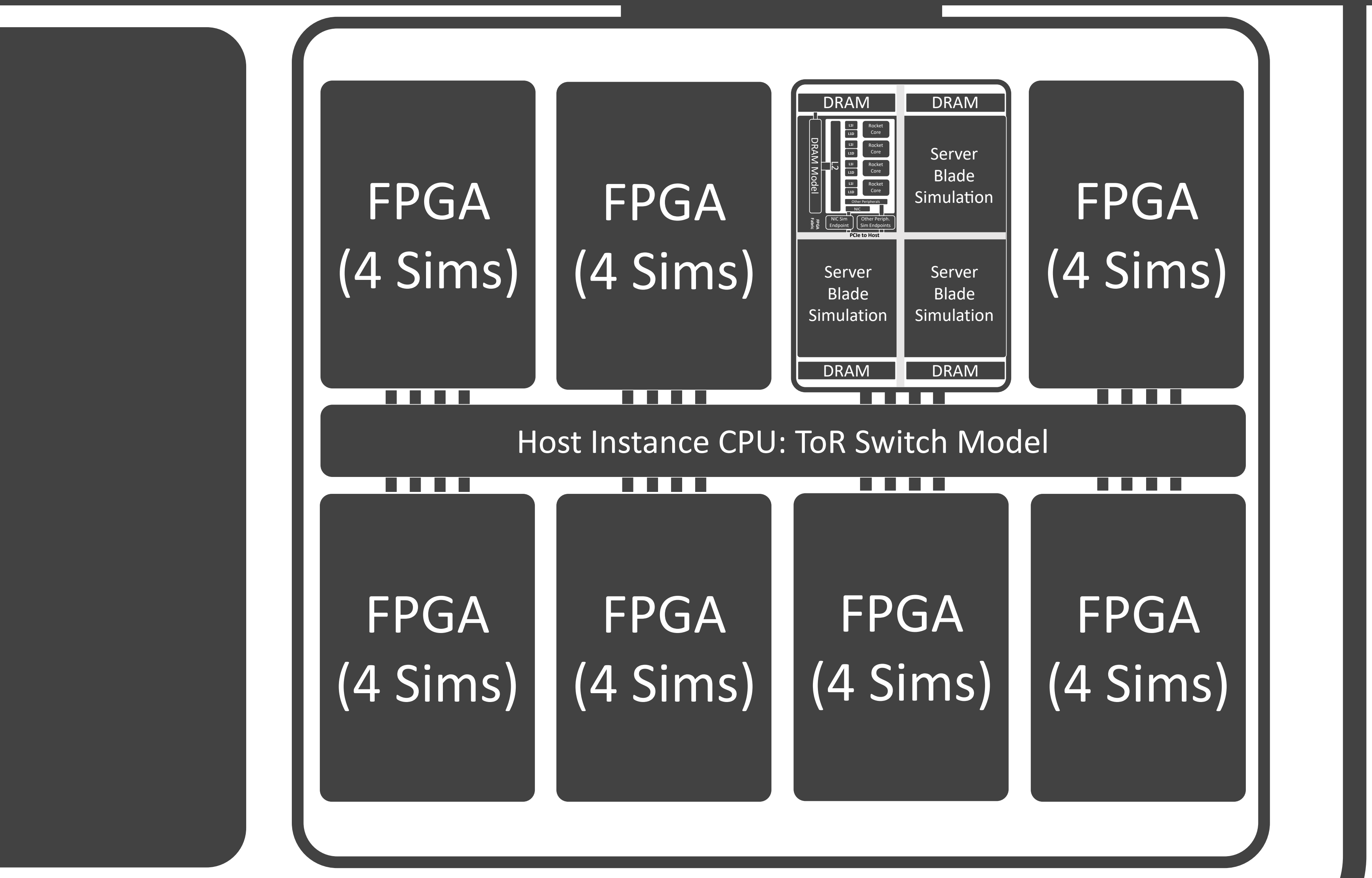
Resource Util.

- 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

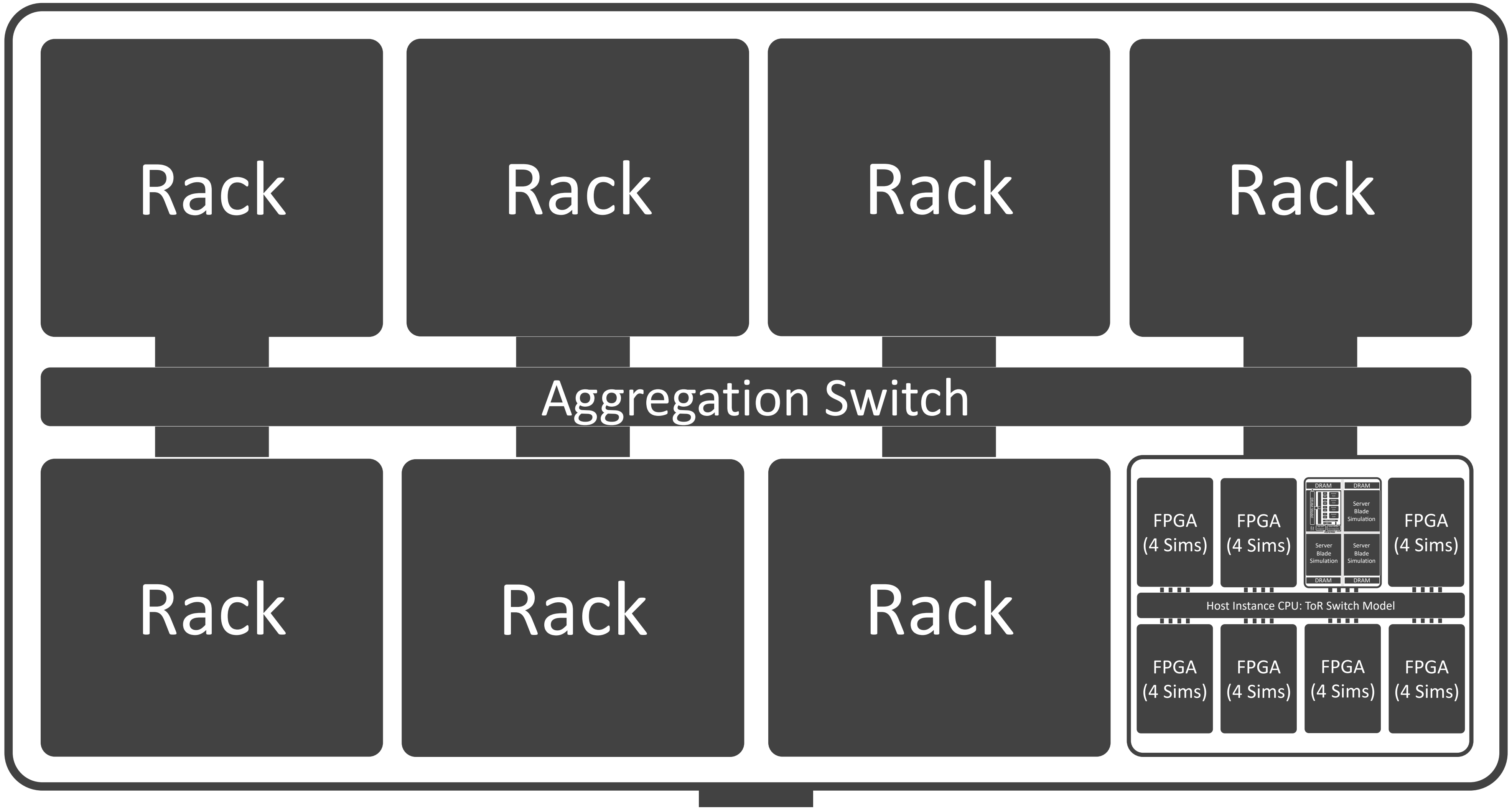
Resource Util.

- 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

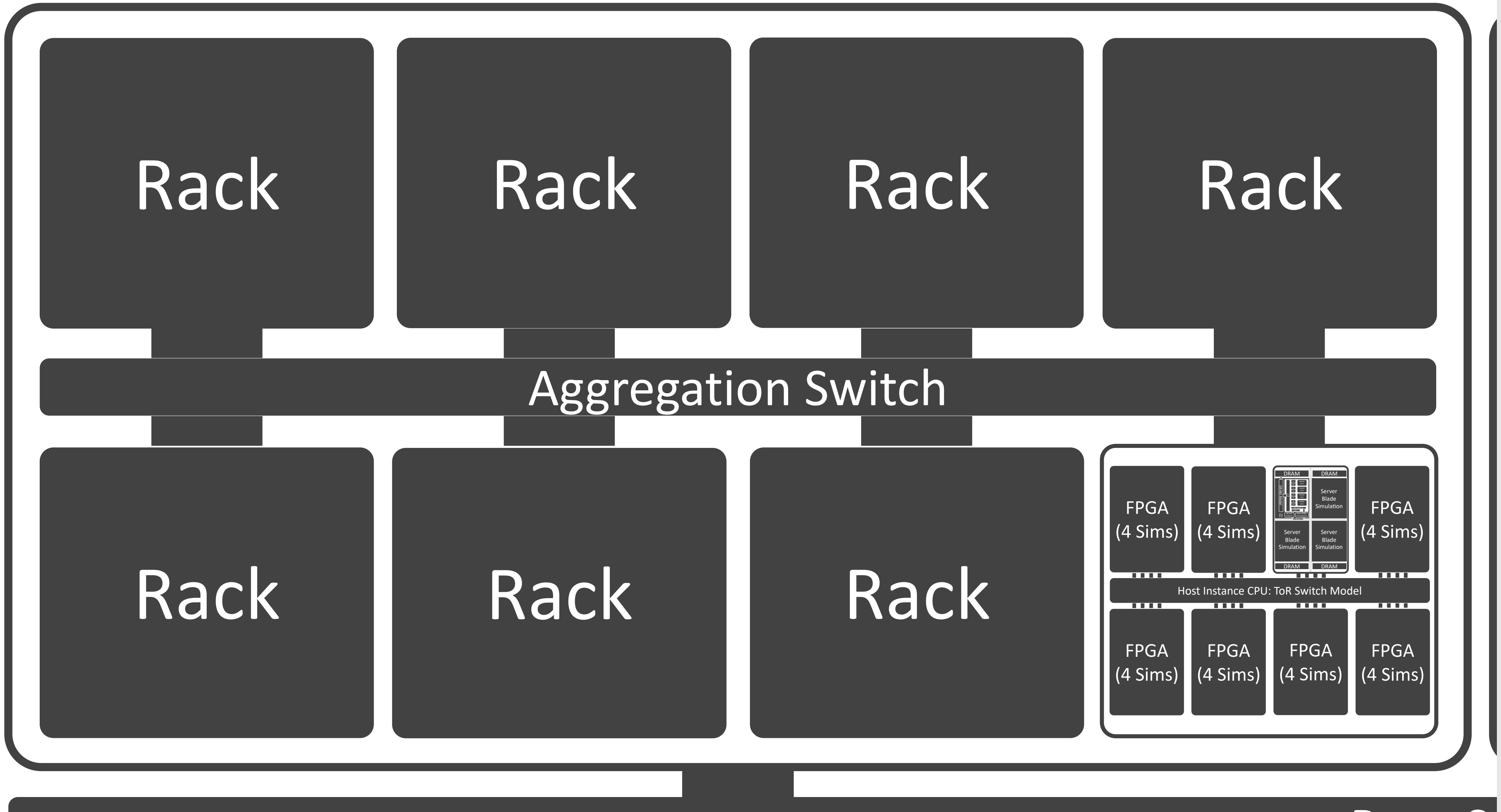
Resource Util.

- 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

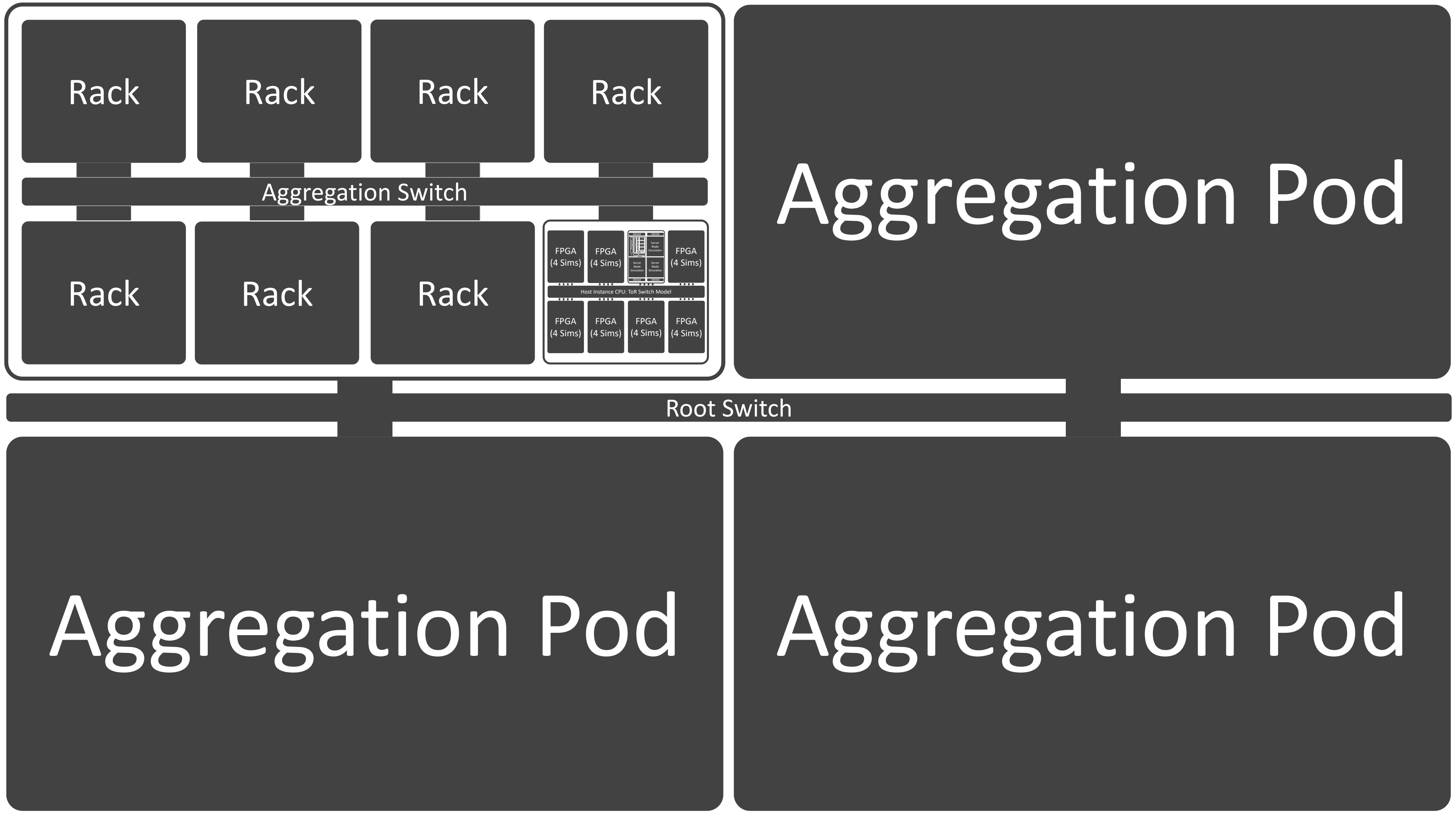
Resource Util.

- 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

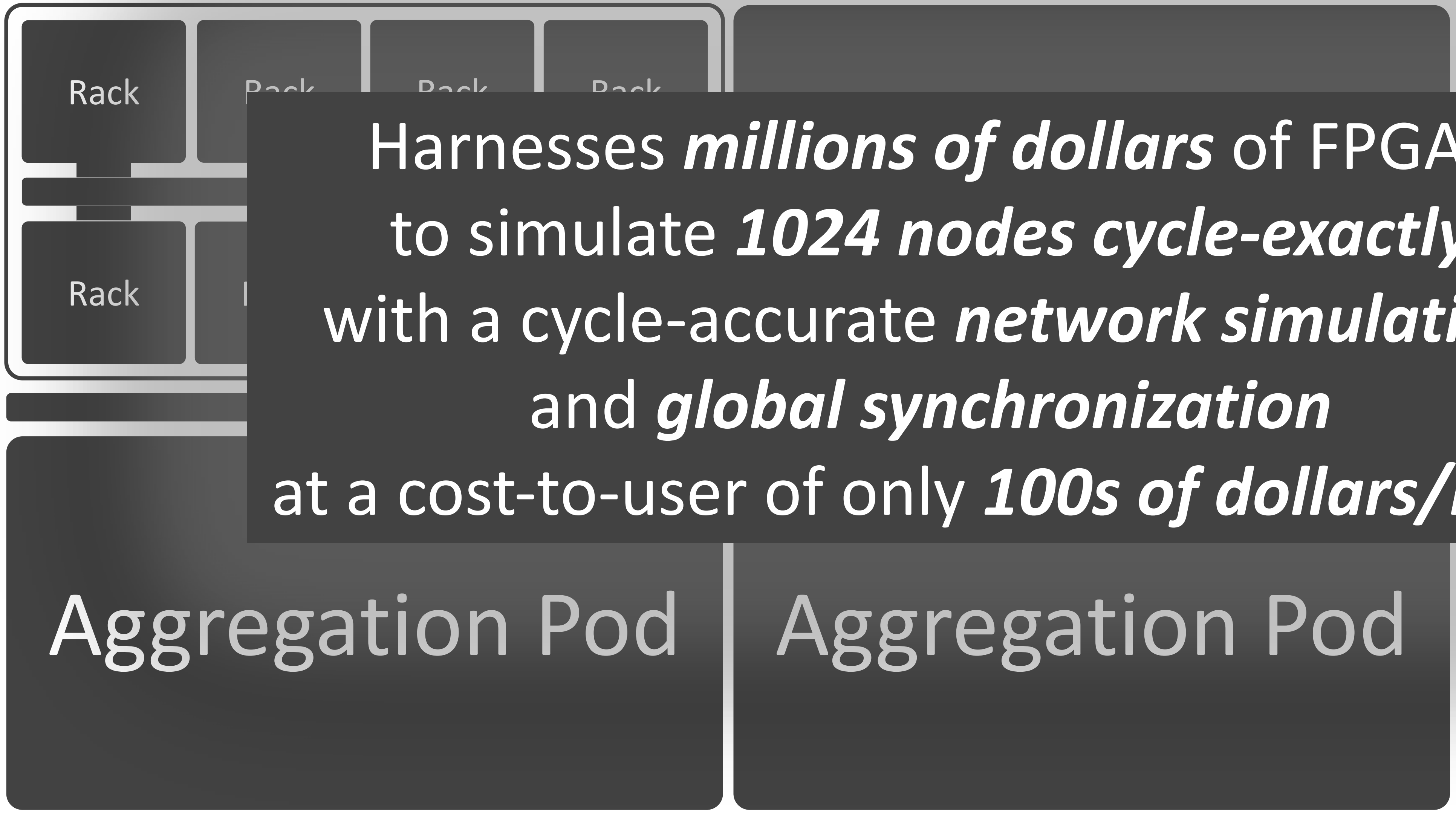
Resource Util.

- 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***

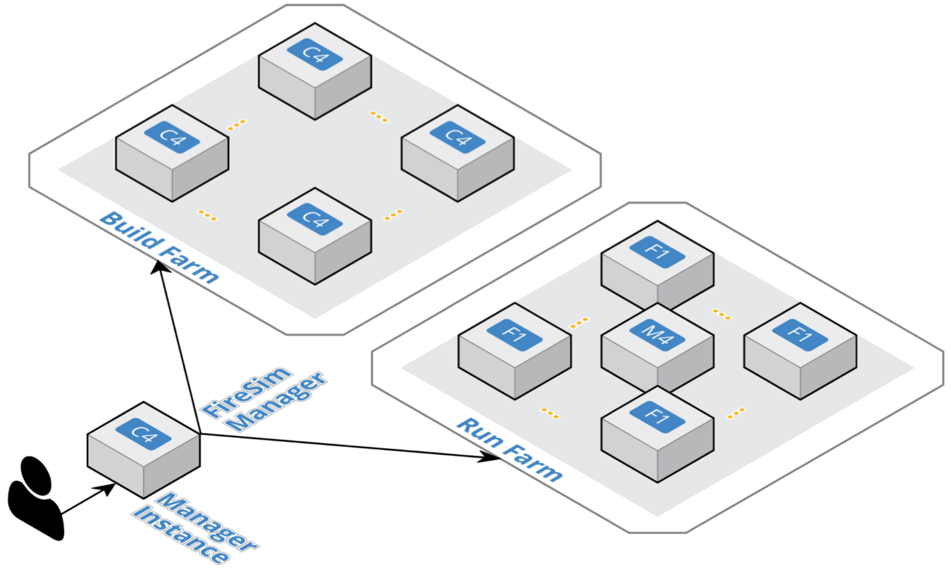
Modeled System
- 1024 Servers
6 Cores
TB DDR3
ToRs, 4 Aggr, 1
Gb/s, 2us

urce Util.
FPGAs =
- 32x f1.16xlarge
- 5x m4.16xlarge

Sim Rate
- ~6.6 MHz (netw)

Open-source: Not just datacenter simulation

- An “easy” button for fast, FPGA-accelerated full-system simulation
  - Replace network endpoints with your own Chisel designs
  - 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

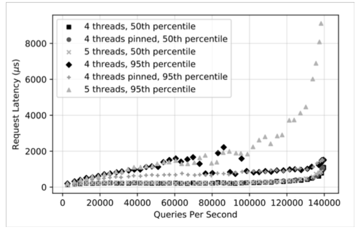


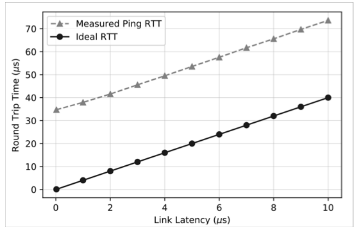
FireSim Developer Environment

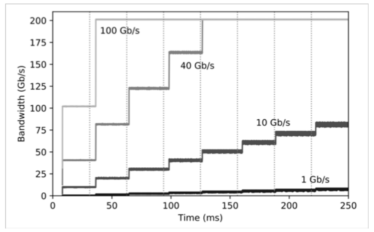
Open-source: Not just datacenter simulation

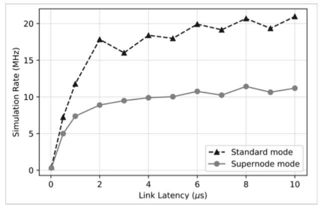
- Scripts can call `firesim` to fully automate distributed FPGA sim
  - **Reproducibility**: included scripts to reproduce ISCA 2018 results
  - e.g. scripts to automatically run SPECInt2017 **reference inputs** in ≈1 day
  - Many others
- 100+ pages of documentation: https://docs.fires.im
- AWS provides grants for researchers: https://aws.amazon.com/grants/

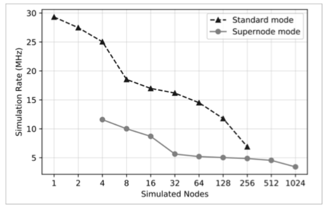
```
$ cd fsim/deploy/workloads
$ ./run-all.sh
```











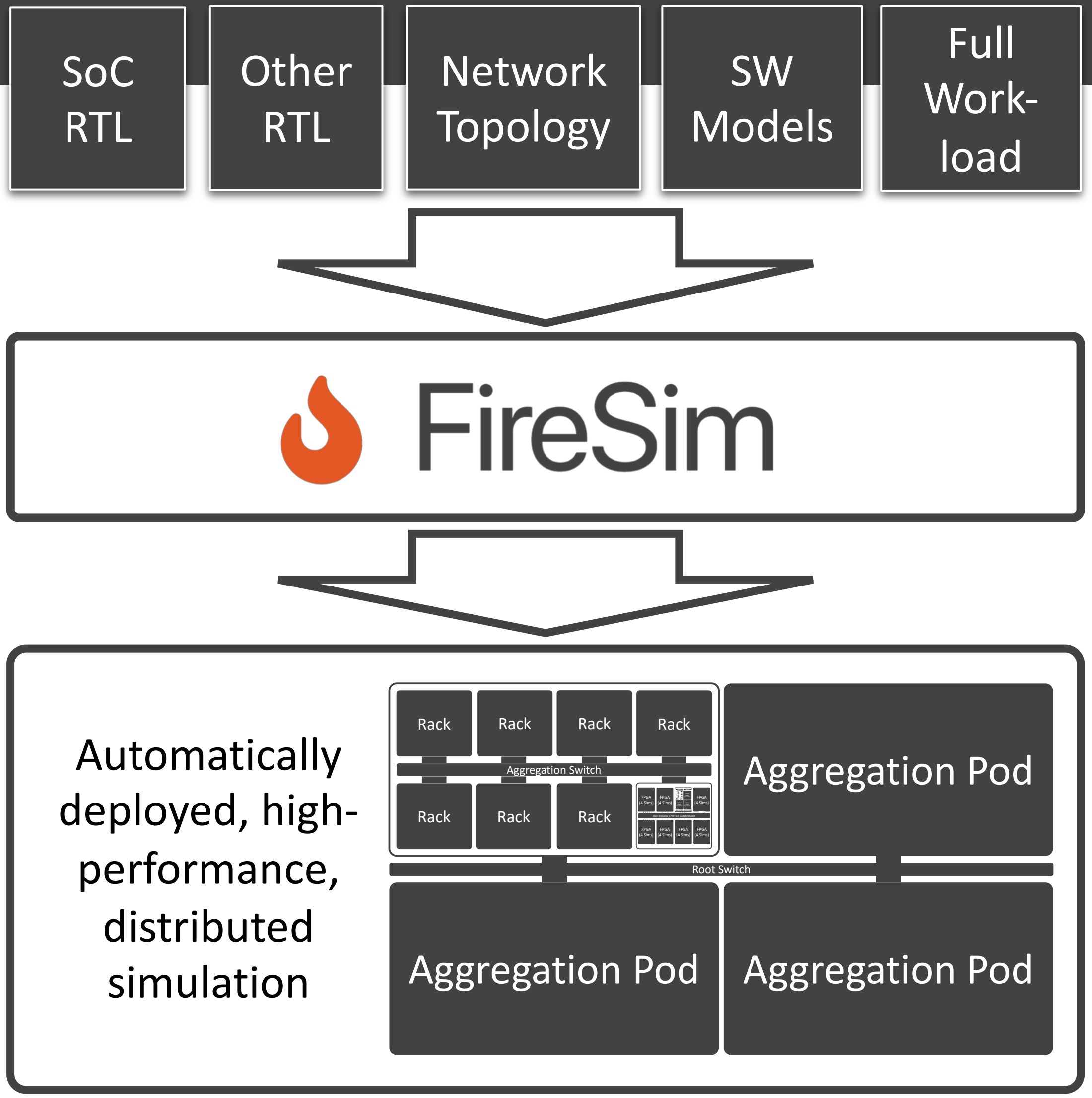
Latest Updates

- Growing ecosystem!
- BOOM (Out-of-order RISC-V core) now available in FireSim
- Projects publicly releasing FireSim images at RISC-V Summit:
  - Hwacha vector accelerator
  - Keystone Secure Enclave
- Berkeley IceNet (photonic DC network) modeling

- New debugging features
  - Auto-ILA: Annotate Chisel and get an ILA automatically wired-up in FireSim
  - Tracer widgets: Collect live instruction traces from FireSim sims
  - Integrating DESSERT [8]
    - Assertion Synthesis now on master
- First Academic User papers:
  - ISCA '18: Maas et. al. "A Hardware Accelerator for Tracing Garbage Collection" (Berkeley)
  - MICRO '18: Zhang et. al. "Composable Building Blocks to Open up Processor Design" (MIT)

Wrapping Up

- We can prototype scalable-systems built on **arbitrary RTL** at **unprecedented scale**
  - + Mix software models when desired
- Simulation is **automatically built and deployed**
- Automatically **deploy real workloads** and collect results
- **Open-source**, runs on Amazon EC2 F1, **no capex**



Administrivia

- Everything we’re going to show you today is documented in excruciating detail at http://docs.fires.im/
- Use these slides as a high-level view of what’s possible
- We’ll also put the slides/videos online

Today's Agenda

- ☑ Intro, what is FireSim capable of?
    - (Sagar, 10 mins)
- Building/Deploying FireSim Simulations
    - (Sagar, 15 mins)
- Using FireSim to simulate your own custom HW
    - (David, 15 mins)
- Testing and Debugging your design in FireSim
    - (Alon, 15 mins)
- Community/Contributing/Conclusion
    - (David, 10 mins)

FireSim

Questions?

Berkeley Architecture Research

Learn More:
Web: https://fires.im
GitHub: https://github.com/firesim
@firesimproject

ISCA'18 Paper:
sagark.org/assets/pubs/firesim-isca2018.pdf

Contact: 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. Research was partially funded by ADEPT Lab industrial sponsor Intel, RISE Lab sponsor Amazon Web Services, and ADEPT Lab affiliates Google, Huawei, Siemens, SK Hynix, and Seagate. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

References

[1] Peter X. Gao, Akshay Narayan, Sagar Karandikar, Joao Carreira, Sangjin Han, Rachit Agarwal, Sylvia Ratnasamy, and Scott Shenker. 2016. Network requirements for resource disaggregation. OSDI'16
[2] Y. Lee *et al*., "An Agile Approach to Building RISC-V Microprocessors," in *IEEE Micro*, vol. 36, no. 2, pp. 8-20, Mar.-Apr. 2016.
[3] Jacob Leverich and Christos Kozyrakis. Reconciling high server utilization and sub-millisecond quality-of-service. EuroSys '14
[4] Zhangxi Tan, Zhenghao Qian, Xi Chen, Krste Asanovic, and David Patterson. DIABLO: A Warehouse-Scale Computer Network Simulator using FPGAs. ASPLOS '15
[5] Tan, Z., Waterman, A., Cook, H., Bird, S., Asanović, K., & Patterson, D. A case for FAME: FPGA architecture model execution. ISCA ’10
[6] Evaluation of RISC-V RTL Designs with FPGA Simulation. Donggyu Kim, Christopher Celio, David Biancolin, Jonathan Bachrach and Krste Asanovic. CARRV '17.
[7] Donggyu Kim, Adam Izraelevitz, Christopher Celio, Hokeun Kim, Brian Zimmer, Yunsup Lee, Jonathan Bachrach, and Krste Asanović. Strober: fast and accurate sample-based energy simulation for arbitrary RTL. ISCA ‘16
[8] Donggyu Kim, Christopher Celio, Sagar Karandikar, David Biancolin, Jonathan Bachrach, and Krste Asanović, “DESSERT: Debugging RTL Effectively with State Snapshotting for Error Replays across Trillions of cycles”, FPL 2018