Welcome to the First FireSim and Chipyard User/Developer Workshop!

Sagar Karandikar

UC Berkeley

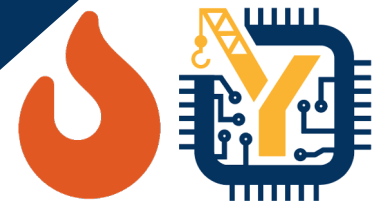
sagark@eecs.berkeley.edu

We'll get started at 9:10am pacific time

Program: https://fires.im/workshop-2023/

Workshop Slack: https://fires.im/workshop-slack/

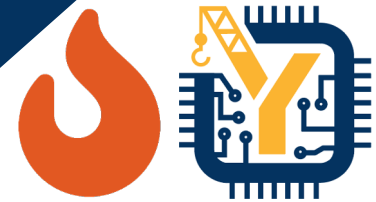
A *Golden Age* in Computer Architecture



- No more traditional scaling…
- An architect's dream: everyone wants custom microarchitectures and HW/SW co-designed systems
- Also, a golden age to have *direct impact* as researchers
  - Exploding open-source hardware environment
  - An open-ISA that can run software we care about

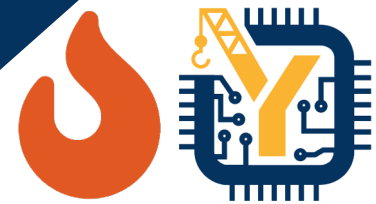
https://cacm.acm.org/magazines/2019/2/234352-a-new-golden-age-for-computer-architecture/fulltext

A *Dark Age* in Computer Architecture tools



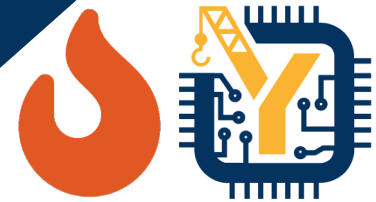
- What do we need to do good architecture research?
  - Need tools that let us evaluate designs on a variety of metrics:
    - Functionality
    - Performance
    - Power
    - Area
    - Frequency
  - Especially in small teams (grad students, startups), these tools need to be *agile*
  - Historically, without good open IP, had to build abstract arch/uarch simulators out of necessity
    - But now, we have much better IP and software compatibility, so what's stopping us?

A *Dark Age* in Computer Architecture tools



- Designed to be operated by hundreds of engineers
- Not, 10s of engineers or 1s-10s of grad students
- Three hard questions:
  - Where do I get a collection of well-tested hardware IP + complex software stacks that run on it?
  - How do I quickly obtain performance measurements for a novel HW/SW system?
  - How do I get ASIC QoR feedback and tape-out a design, with portability across tools and processes?

Three hard questions, answered!



- Where do I get a collection of well-tested hardware IP + complex software stacks that run on it?

- How do I quickly obtain performance measurements for a novel HW/SW system?

- How do I get ASIC QoR feedback and tape out a design, with portability across tools and processes? (and open-source and proprietary flows)

Three hard questions, answered!

+

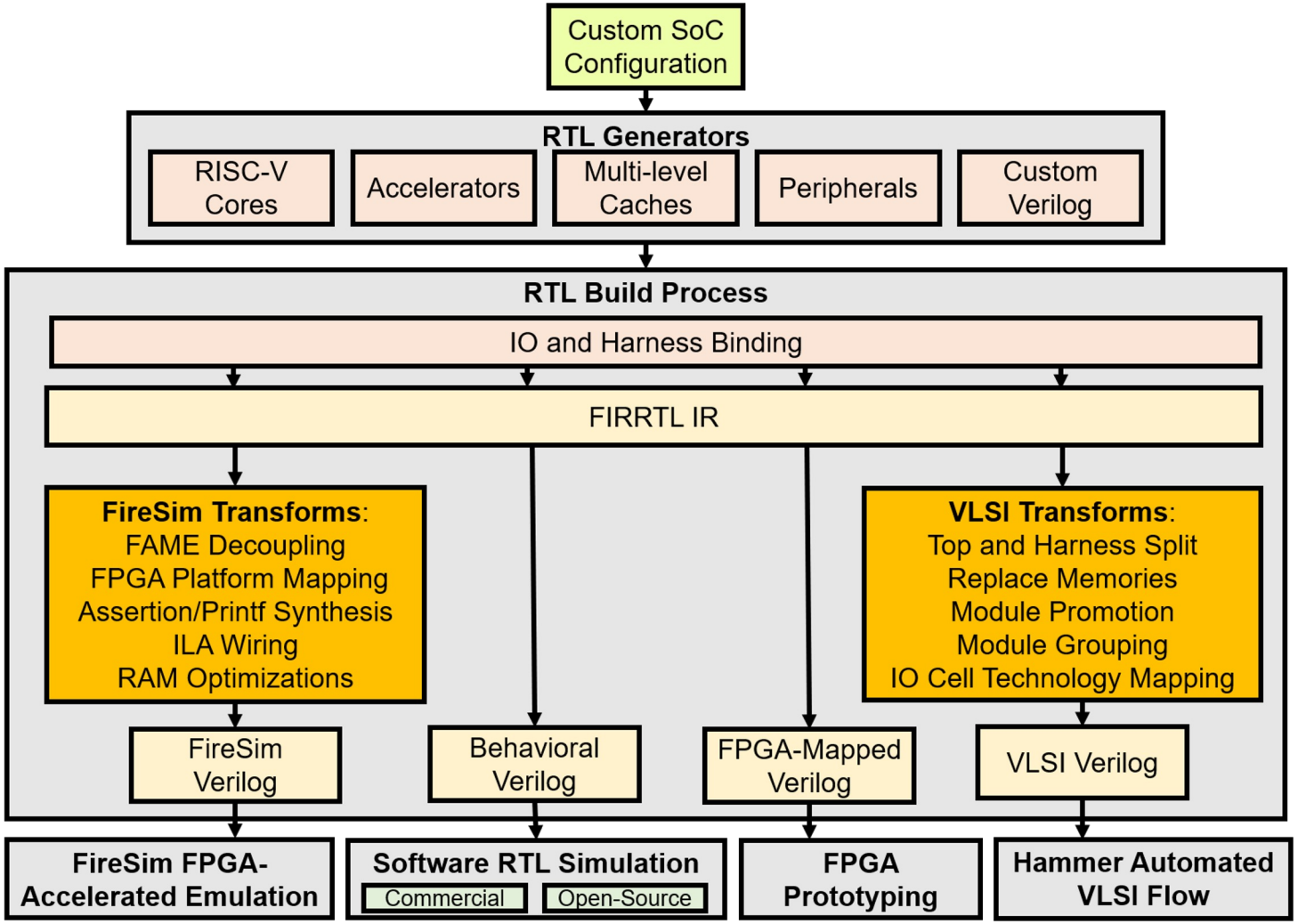
Measure Functionality, Performance, Power, Area, Frequency *for real HW/SW systems*, quickly and easily, with small teams of engineers

What is Chipyard?

CHIPYARD Organization

What is Chipyard?

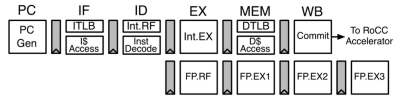
- An organized **framework** for various SoC design tools
- A **curated IP library** of open-source RISC-V SoC components
- A **methodology** for agile SoC architecture design, exploration, and evaluation

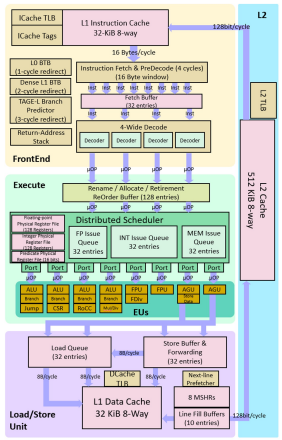


CHIPYARD What's included?

Cores:

Rocket and BOOM





Rocket:

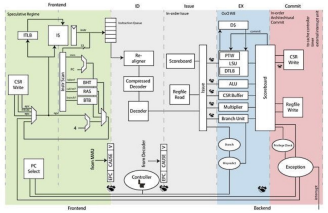
- First open-source RISC-V CPU
- In-order, single-issue RV64GC core
- Efficient design point for low-power devices

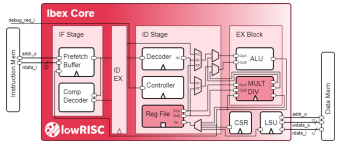
SonicBOOM:

- Superscalar out-of-order RISC-V CPU
- Advanced microarchitectural features to maximize IPC
- TAGE branch prediction, OOO load-store-unit, register renaming
- High-performance design point for general-purpose

7

PULP Cores in CHIPYARD





CVA6 (Formerly Ariane):

- RV64IMAC 6-stage single-issue in-order core
- Open-source
- Implemented in SystemVerilog
- Developed at ETH Zurich as part of PULP,
- Now maintained by OpenHWGroup

Ibex (Formerly Zero-RISCY):

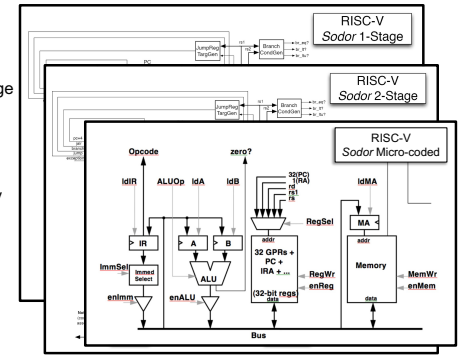
- RV64IMC 2-stage single-issue in-order core
- Open-source
- Implemented in SystemVerilog
- Developed at ETH Zurich as part of PULP
- Now maintained by lowRISC

9

Sodor Education Cores

Sodor Core Collection

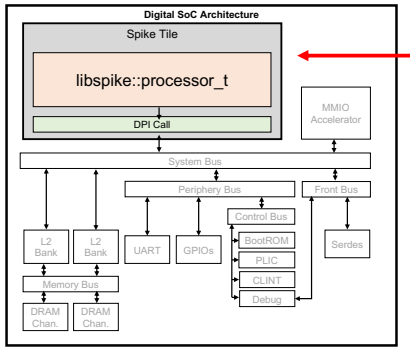
- Collection of RV32IM cores for teaching and education
- 1-stage, 2-stage, 3-stage, 5-stage implementations
- Micro-coded "bus-based" implementation
- Used in introductory computer architecture courses at Berkeley



10

Spike –as –a– Tile

New in Chipyard 1.9.0



Spike:

- Open-source RISC-V ISA simulator
- Fast, extensible, C++ functional model

Spike-as-a-Tile:

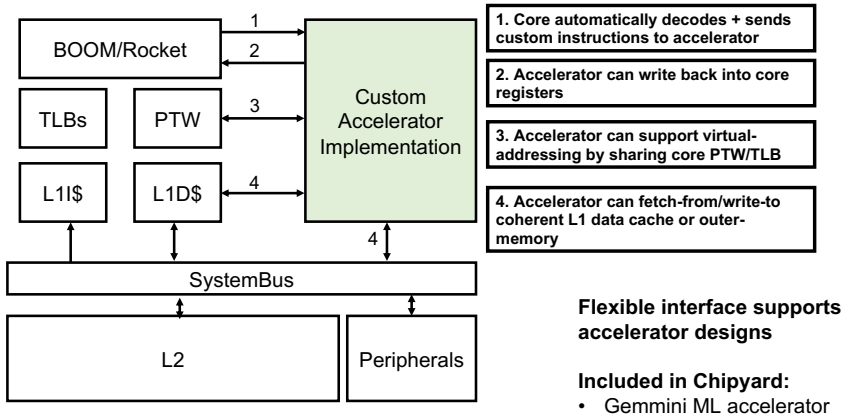
- DPI interface between RTL SoC simulation and SoC functional model
- Spike "virtual platform"
- Enables testing of complex device software in RTL simulation

11

CHIPYARD What's included?

Accelerator Sockets:

RoCC Accelerators



1. Core automatically decodes + sends custom instructions to accelerator

2. Accelerator can write back into core registers

3. Accelerator can support virtual-addressing by sharing core PTW/TLB

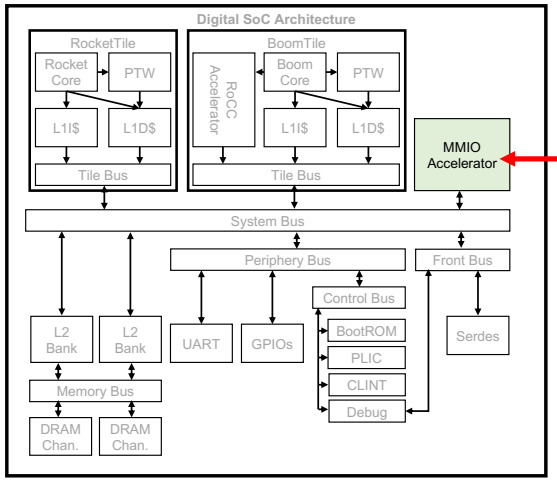
4. Accelerator can fetch-from/write-to coherent L1 data cache or outer-memory

Flexible interface supports a variety of accelerator designs

Included in Chipyard:
- Gemmini ML accelerator
- Hwacha vector accelerator
- SHA3 accelerator

13

MMIO Accelerators



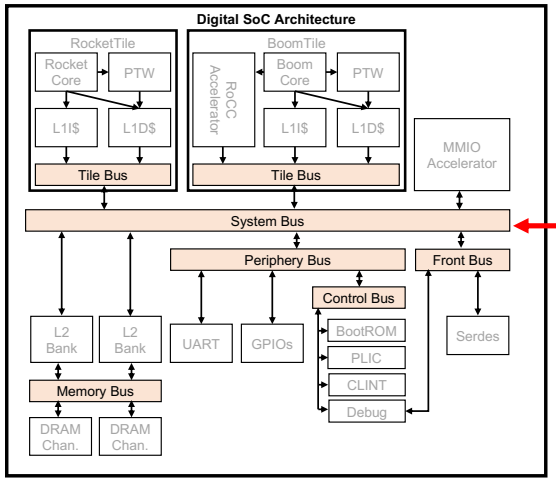
MMIO Accelerators:
- Controlled by MMIO-mapped registers
- Supports DMA to memory system
- Examples:
  - Nvidia NVDLA accelerator
  - FFT accelerator generator

14

CHIPYARD What's included?

Interconnect IP:

Coherent Interconnect



TileLink Standard:
- TileLink is open-source chip-scale interconnect standard
- Comparable to AXI/ACE
- Supports multi-core, accelerators, peripherals, DMA, etc

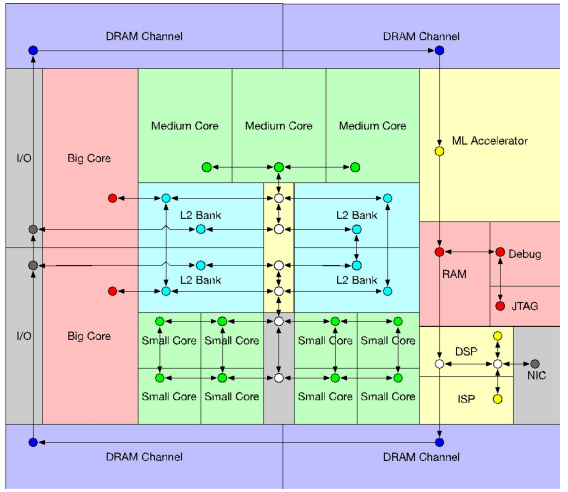
Interconnect IP:
- Library of TileLink RTL generators provided in RocketChip
- RTL generators for crossbar-based buses
- Width-adapters, clock-crossings, etc.
- Adapters to AXI4, APB

15

Constellation NoC Generator

A parameterized Chisel generator for SoC interconnects

- Protocol-independent transport layer
- Supports TileLink, AXI-4
- Highly parameterized
- Deadlock-freedom
- Virtual-channel wormhole-routing

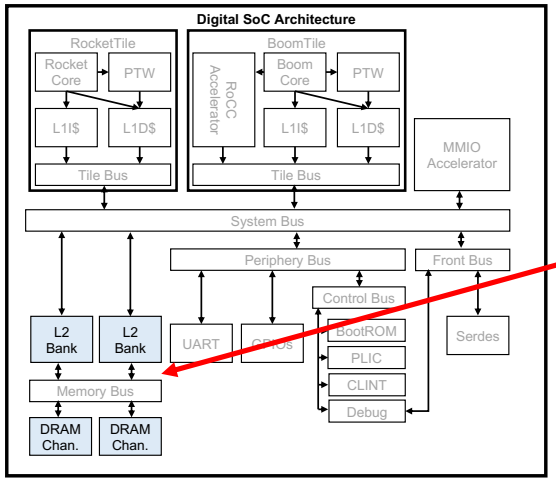


18

CHIPYARD What's included?

Interconnect IP:

L2/DRAM

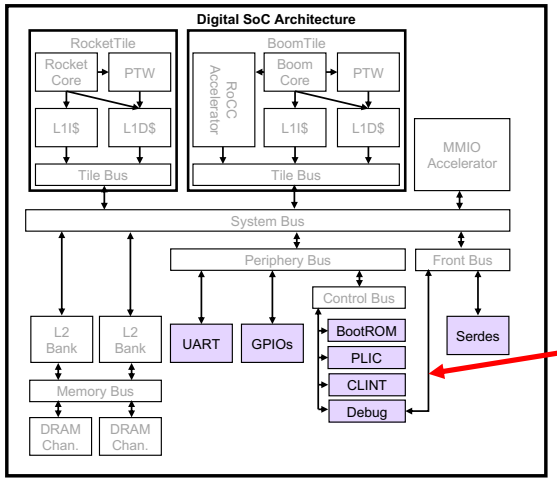


Shared memory:
- Open-source TileLink L2 developed by SiFive
  - Directory-based coherence with MOESI-like protocol
  - Configurable capacity/banking
- Support broadcast-based coherence in no-L2 systems
- Support incoherent memory systems

DRAM:
- AXI-4 DRAM interface to external memory controller
- Interfaces with DRAMSim

19

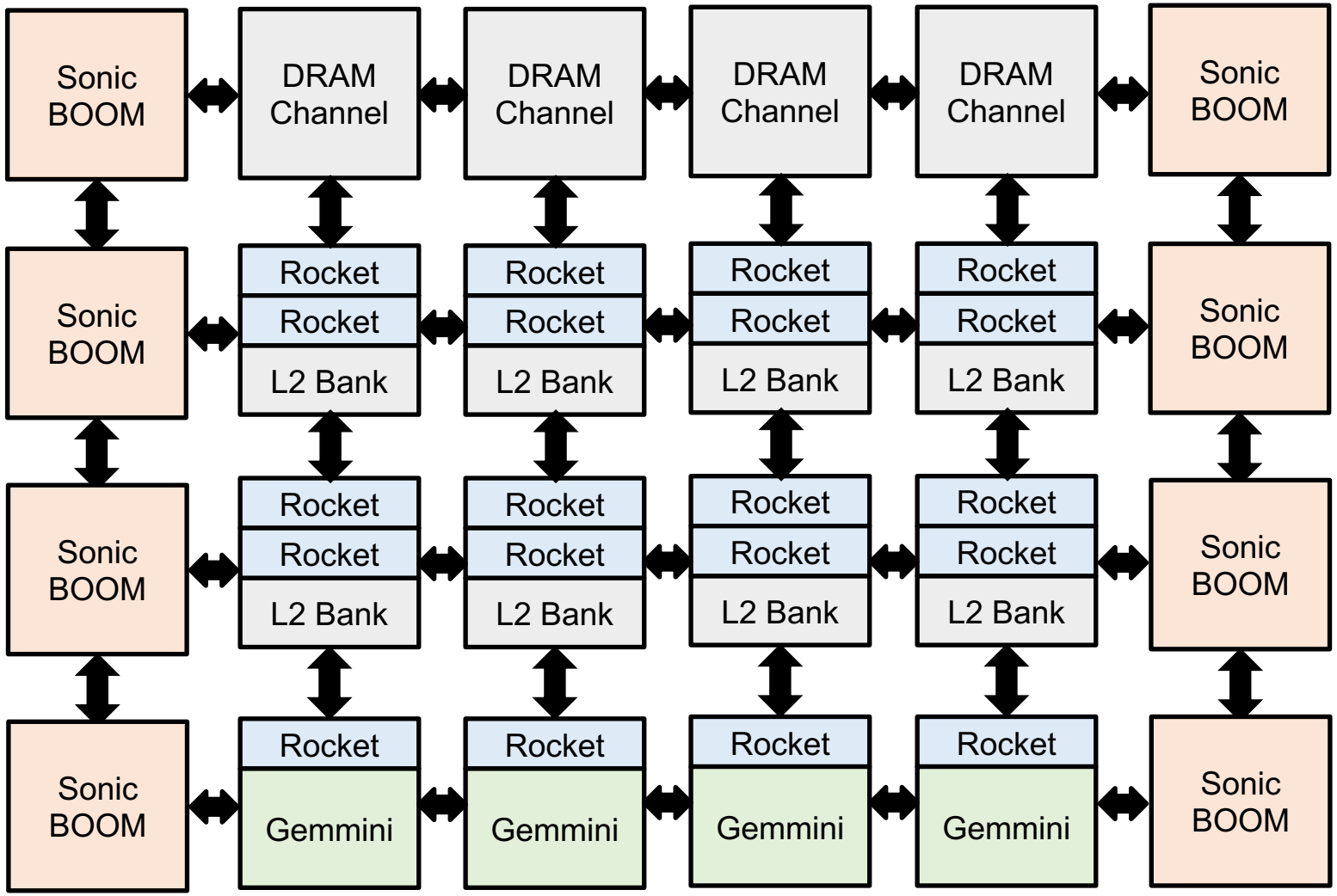
Peripherals and IO



Peripherals and IO:
- Open-source RocketChip blocks
  - Interrupt controllers
  - JTAG, Debug module, BootROM
  - UART, GPIOs, SPI, I2C, PWM, etc.
- TestChipIP: useful IP for test chips
  - Clock-management devices
  - SerDes
  - Scratchpads

20

Example from yesterday's tutorial: TutorialManyCoreNoCConfig



- 6 x 4 mesh NoC
- **4 x** Rocket + Gemmini Accelerator Tiles
- **16 x** Rocket Cores
- **8 x** 10-wide SonicBoom Tiles
- **8 x** Banks of L2 cache

- Total 28 coherent cores

Tutorial attendees yesterday ran through compiling and simulating this live, in <20 mins!

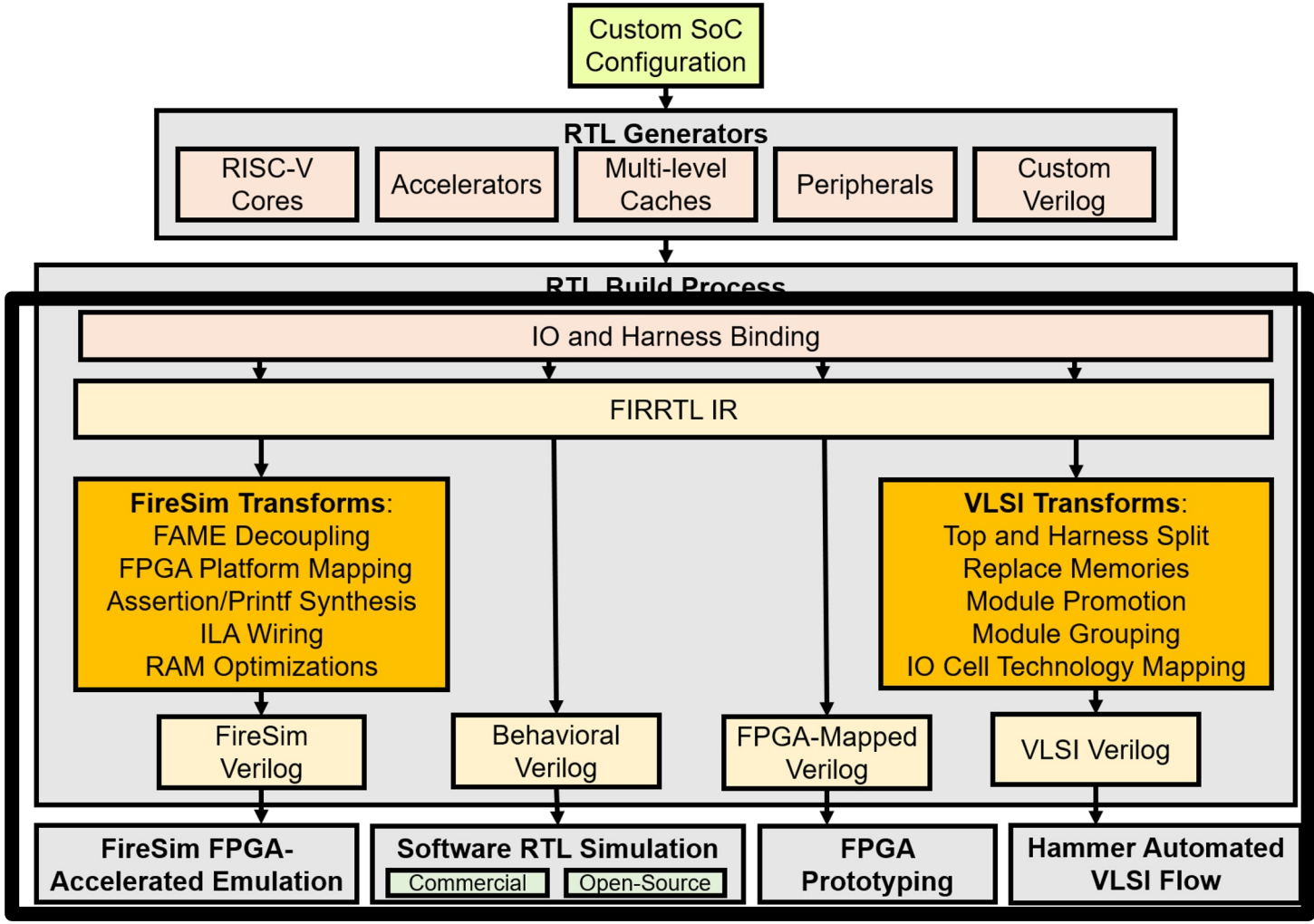
CHIPYARD Flows

SW RTL Simulation:
- RTL-level simulation with Verilator, Xcelium, or VCS

Hammer VLSI flow:
- Tapeout a custom config in some process technology
- Portable across processes/tools

FireSim:
- Fast, cycle-exact, and deterministic FPGA-accelerated simulations
- Scale from 150+ MHz single-SoC sims to 10+ MHz 1024 SoC datacenter sims

FPGA prototyping:
- If your final deployment target is an FPGA or as a bringup platform for taped-out chips



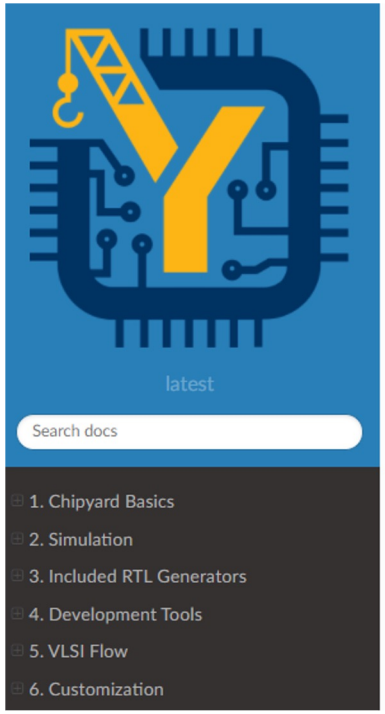
CHIPYARD Community

Documentation:

- https://chipyard.readthedocs.io
- 140+ pages

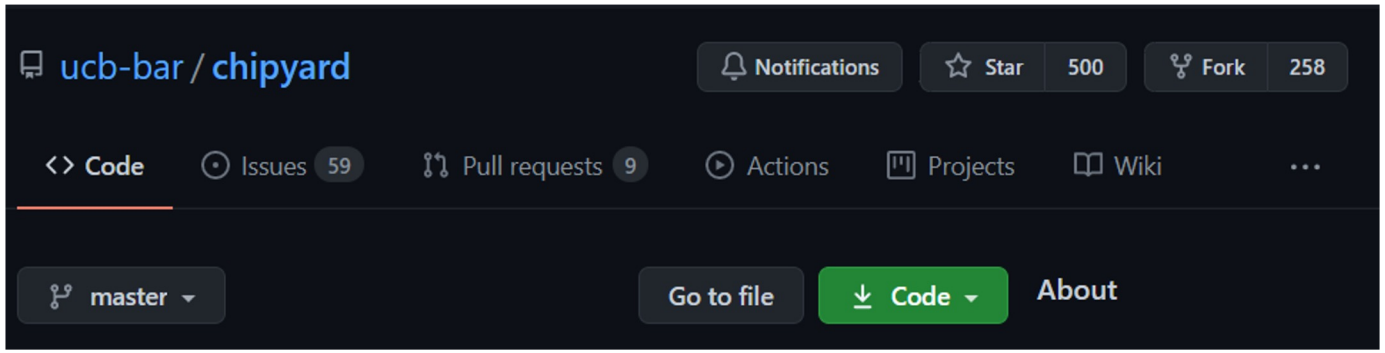
Mailing List:

- google.com/forum/#!forum/chipyard



Open-sourced:

- All code is hosted on GitHub
- Issues, feature-requests, PRs are welcomed



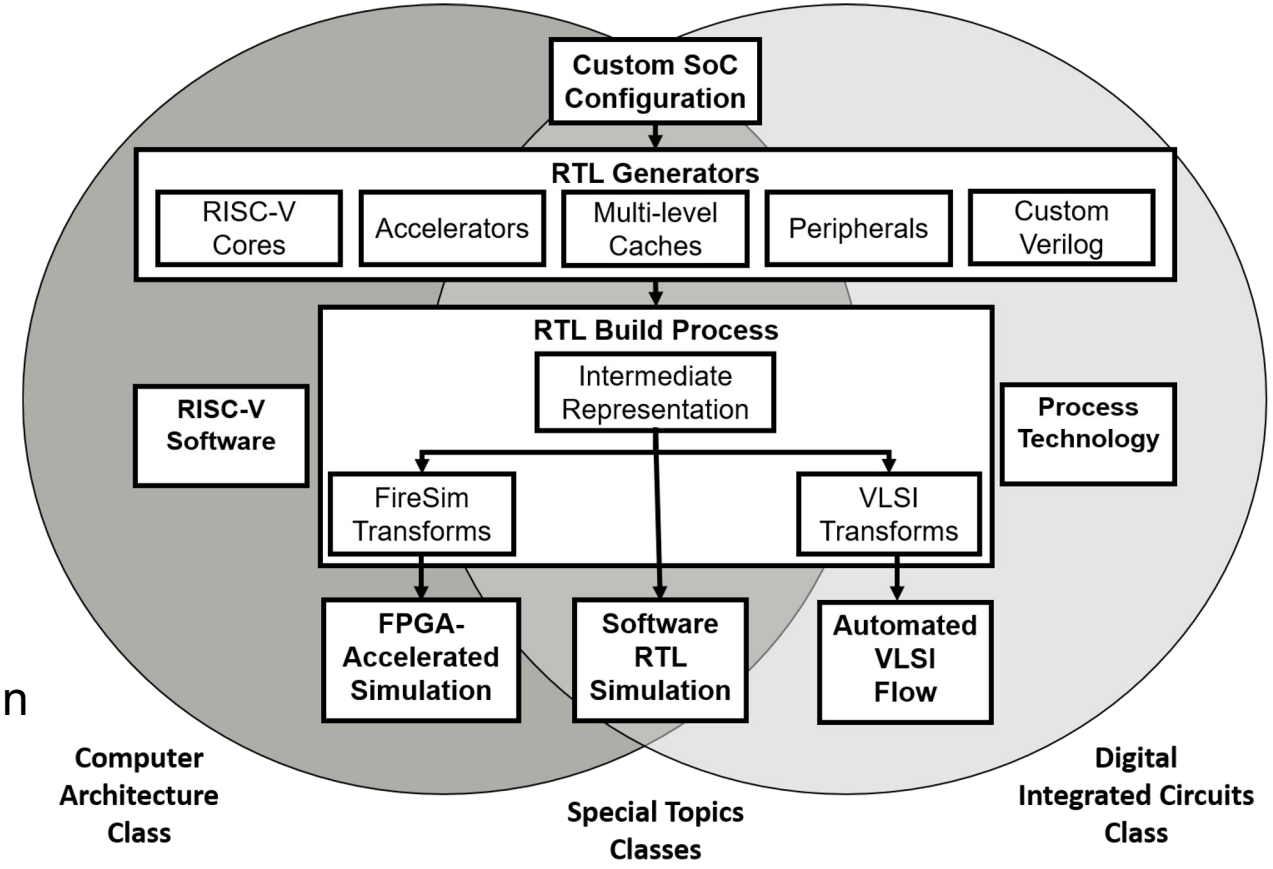
CHIPYARD For Education

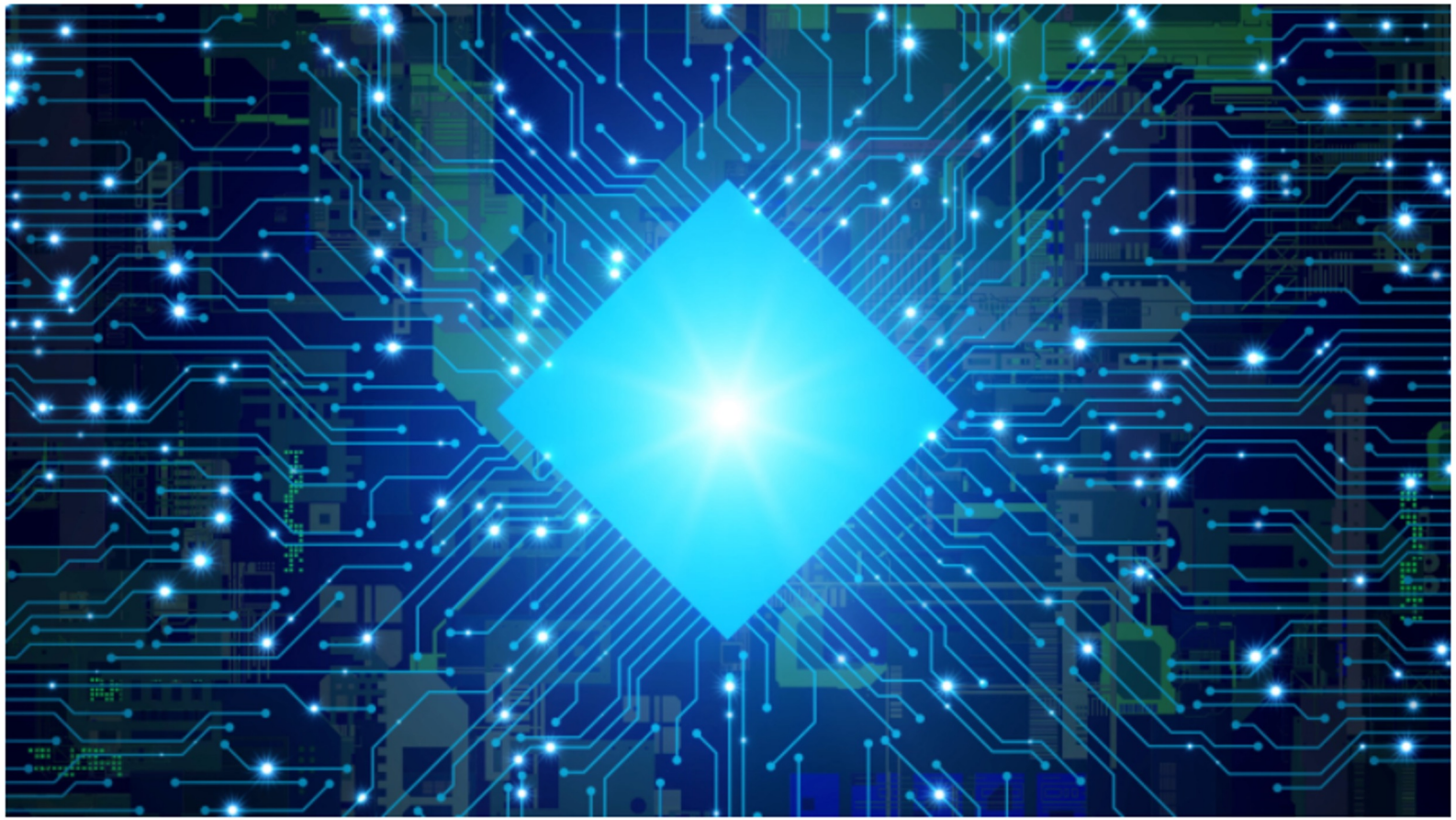
Proven in many Berkeley Architecture courses

- Hardware for Machine Learning
- Undergraduate Computer Architecture
- Graduate Computer Architecture
- Advanced Digital ICs
- Tapeout HW design course

Advantages of common shared HW framework

- Reduced ramp-up time for students
- Students learn framework once, reuse it in later courses
- Enables more advanced course projects (tapeout a chip in 1 semester)





Berkeley Engineering students pull off novel chip design in a single semester. The class shows successful model for expanding entry into field of semiconductor design

Berkeley engineering students pull off novel chip design in a single semester

Class shows successful model for expanding entry into field of semiconductor design

What is FireSim?

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), 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 on-prem or cloud FPGAs
  - E.g., Xilinx Alveo or AWS EC2 F1

FireSim within an architect/chip-dev's toolkit

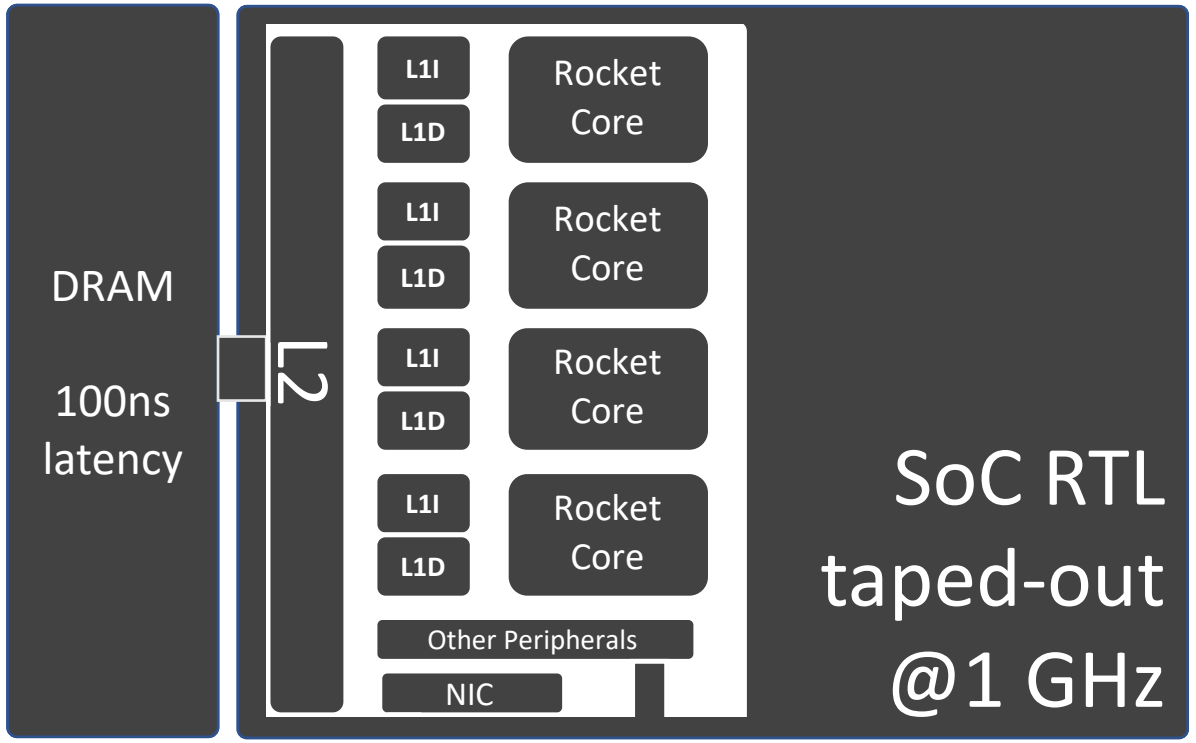
1. High-level Simulation
2. Write RTL + Software, plug into your favorite ecosystem (e.g. Chipyard)
3. Co-design in software RTL sim (e.g. Verilator, VCS, etc.)
   - Run microbenchmarks
4. **Co-design in FPGA-accelerated simulation**
   - **Boot an OS and run the complete software stack, obtain realistic performance measurements**

FireSim

5. Tapeout → Chip
   - Boot OS and run applications, but no more opportunity for co-design

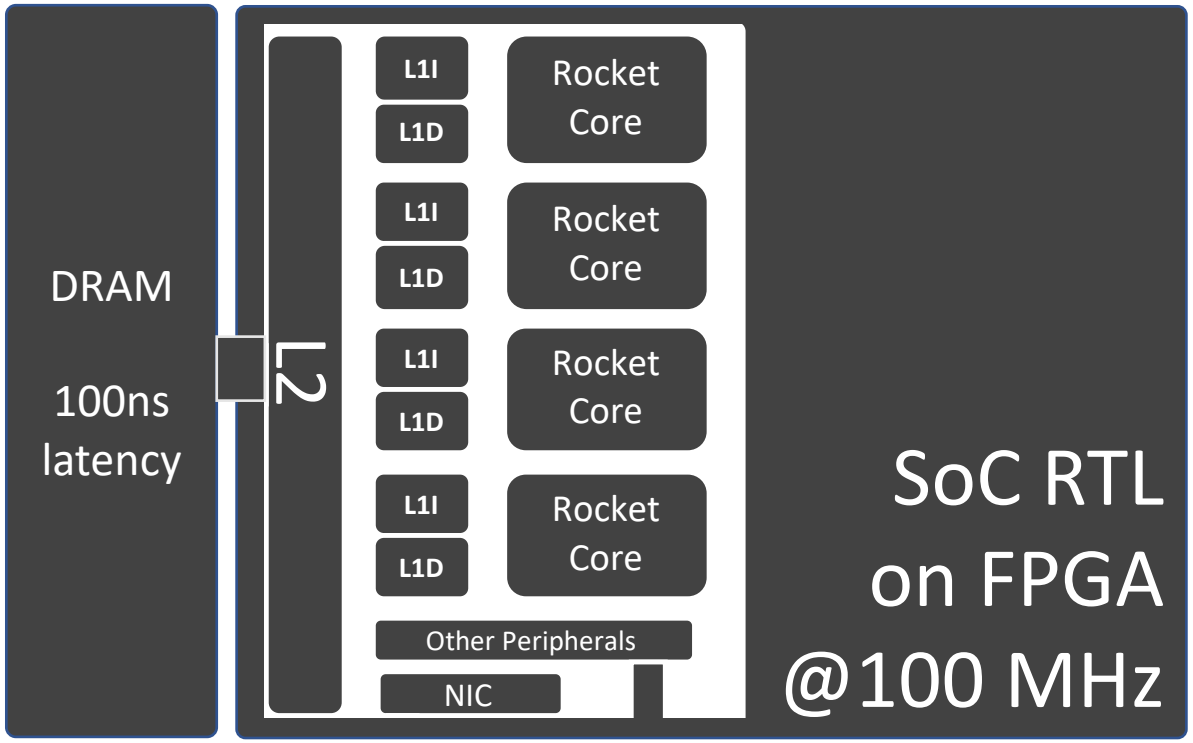
What about FPGA *prototyping*?

Taped-out SoC



SoC sees 100 cycle DRAM latency

FPGA Prototype of SoC



SoC sees 10 cycle DRAM latency

Incorrect by a factor of 10!

Difficulties with FPGA Prototypes

In an FPGA prototype:

- Every FPGA clock executes one cycle of the simulated target
- Performance of FPGA-attached resources is exposed to the simulated world, e.g. DRAM, SD Card, UART, Ethernet, etc.

This leads to three problems:

1) Incorrect performance modeling: FPGA resources probably not an accurate representation of target system
   a) E.g., DRAM performance off by ***10x*** on previous slide
2) Simulations are non-deterministic
3) Different host FPGAs produce different simulation results

Three Distinguishing Features of FireSim

1) Not FPGA prototypes, rather FPGA-accelerated simulators
   - Automatic transformation of RTL designs into FPGA-accelerated simulators via Host Decoupling
   - Enables new debugging, resource optimization, and profiling capabilities
2) Flexible scaling from on-prem to cloud FPGAs
   - Scale easily from one or more on-prem FPGAs to massively parallel simulations on elastic supply of cloud FPGAs
   - Standardized host platforms = easy to collaborate with other researchers and perform artifact evaluation
   - Heavy automation to hide FPGA complexity, regardless of on-prem or cloud platform
3) Open-source (https://fires.im)

Benefits of Host Decoupling on FPGAs

Simulations will now:

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

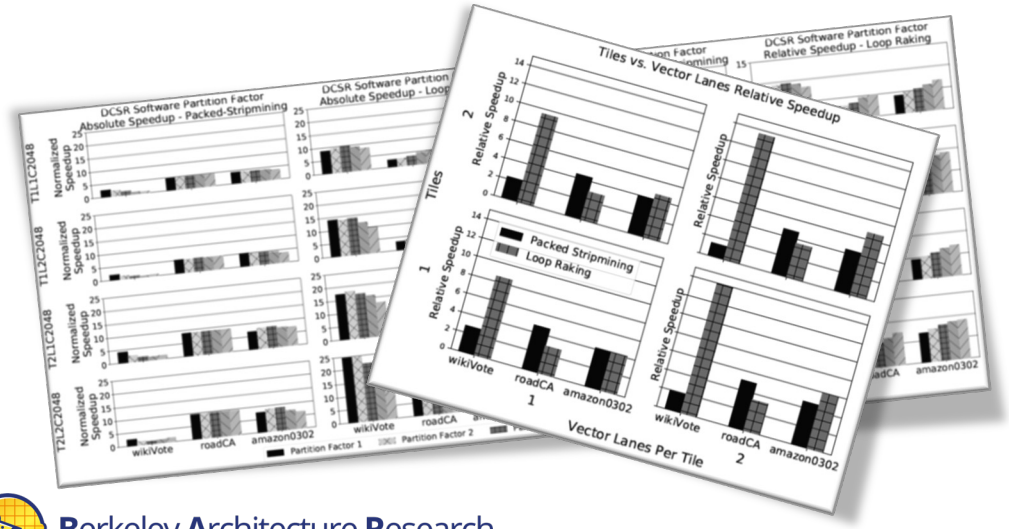
Decoupling enables support for:

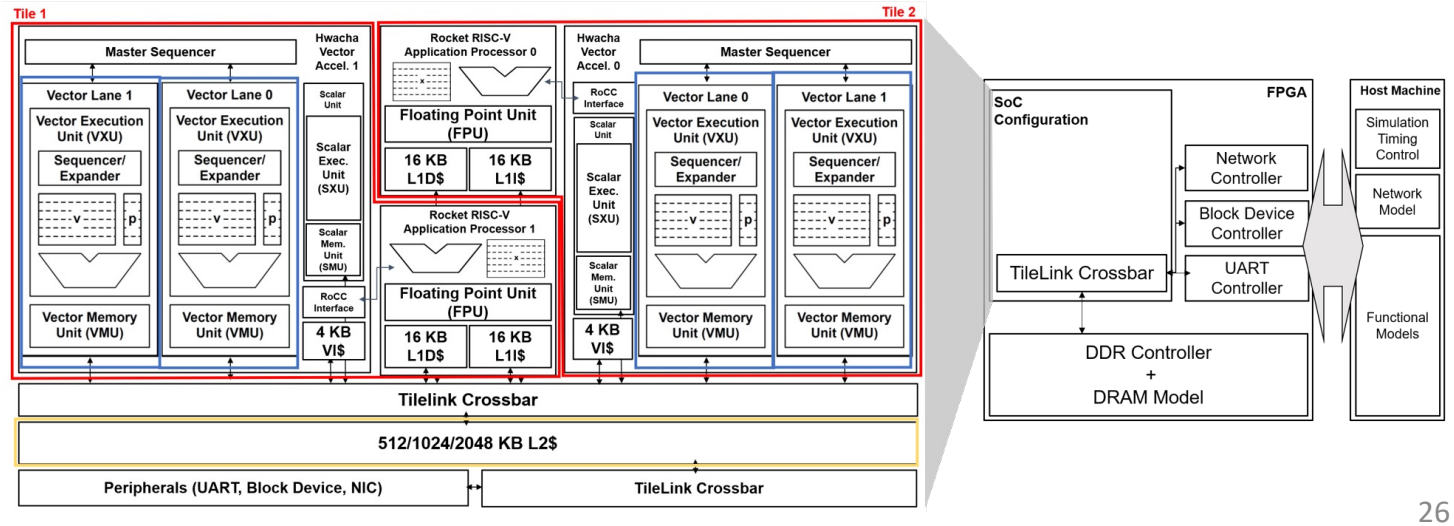
1. SW co-simulation (e.g. block device, network models)
2. Simulating large targets over distributed hosts (ISCA ‘18, Top Picks ‘18)
3. Non-invasive debugging and instrumentation (FPL ‘18, ASPLOS ’20, ASPLOS ‘23)
4. Multi-cycle resource optimizations (ICCAD ‘19)

What Can You Do With FireSim?

Example use cases: Evaluating SoC Designs

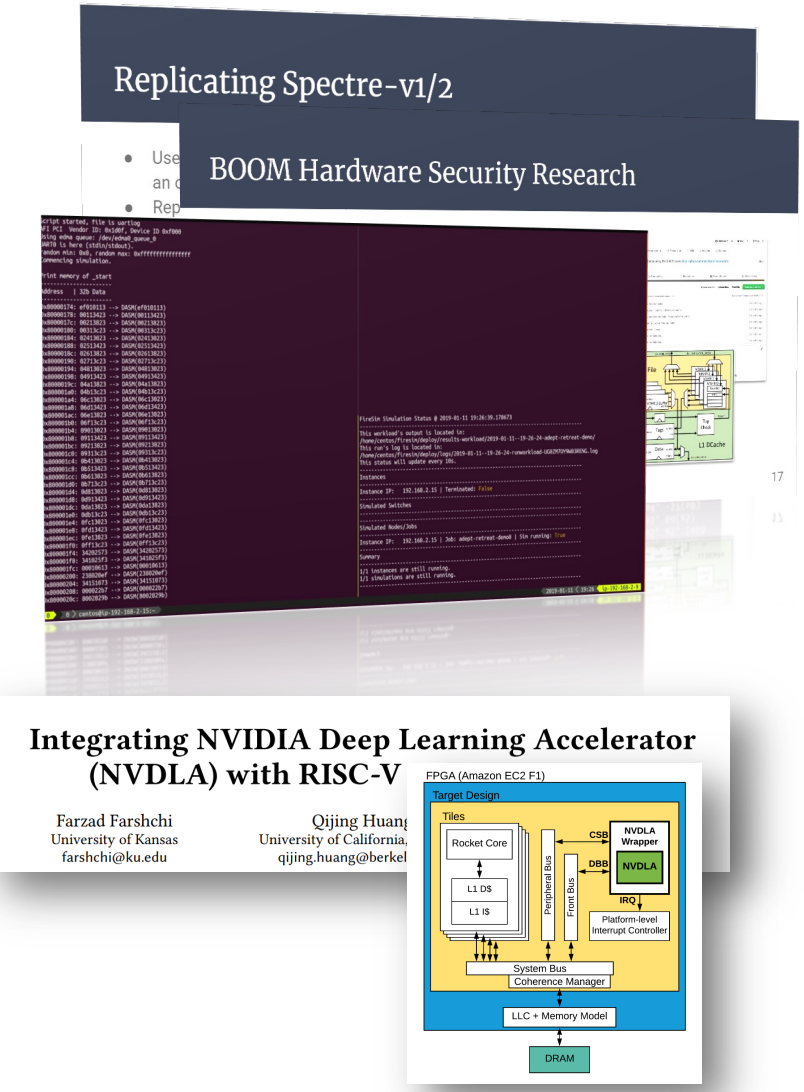
- "Classical" Performance Measurement
  - Run SPECint 2017 with full reference inputs on Rocket Chip in parallel on ~10 FPGAs running at 150+ Mhz. Run the entire suite within a day! (e.g., in D. Biancolin, et. al., *FASED*, FPGA '19)
- Rapid Full-System Design Space Exploration
  - Can rapidly sweep parameter space of a design with FireSim automation
  - Data-parallel accelerators (Hwacha) and multi-core processors
  - Complex software stacks (Linux, OpenMP, GraphMat, Caffe)





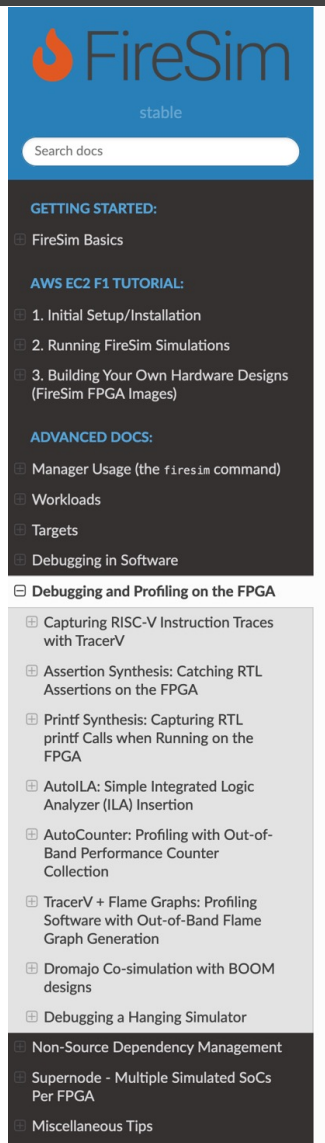
Example use cases: Evaluating SoC Designs

- 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.)



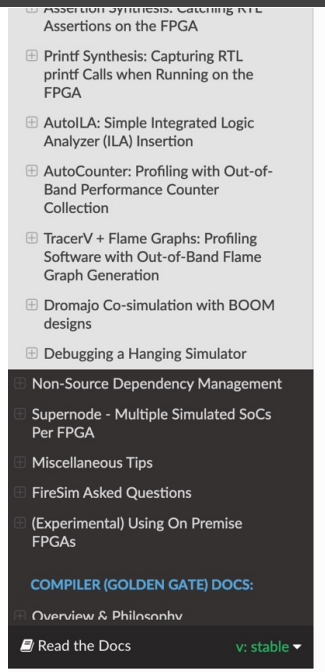
Example use cases: Debugging and Profiling SoC Designs

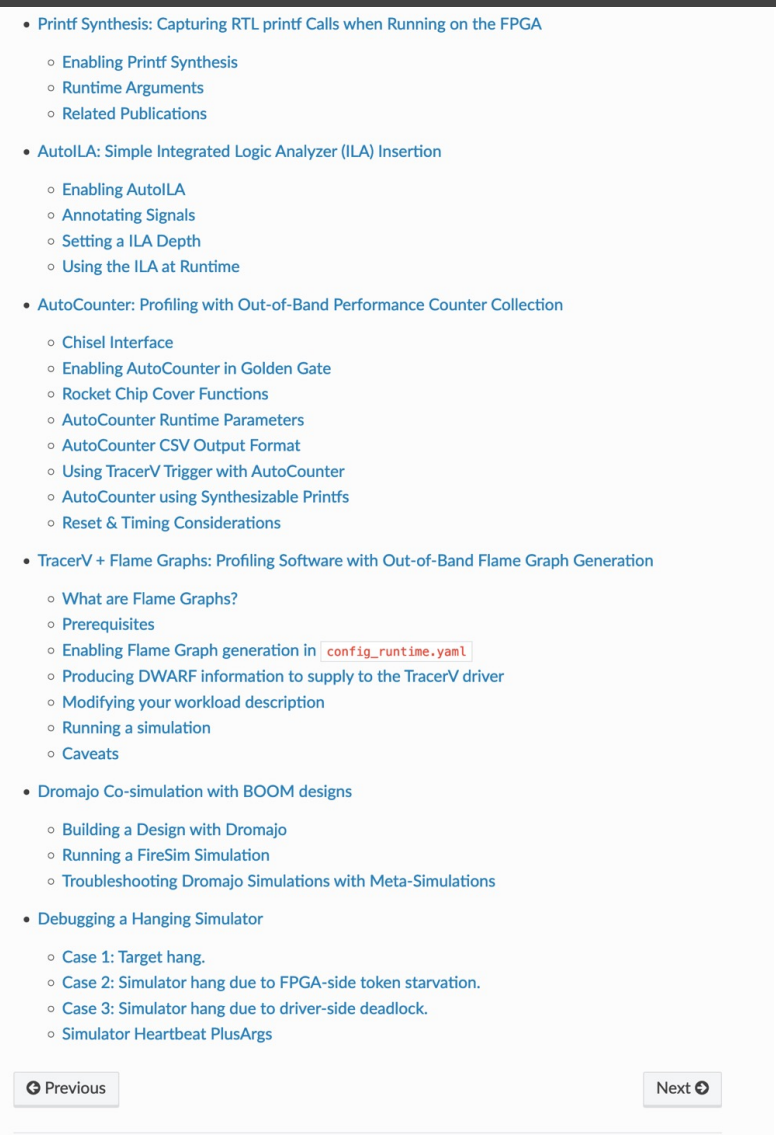
- Debugging a Chisel design at FPGA-speeds
  - Many FireSim debugging features: Assertion synthesis, printf synthesis, ILA insertion, etc.
  - e.g. FireSim Debugging Docs



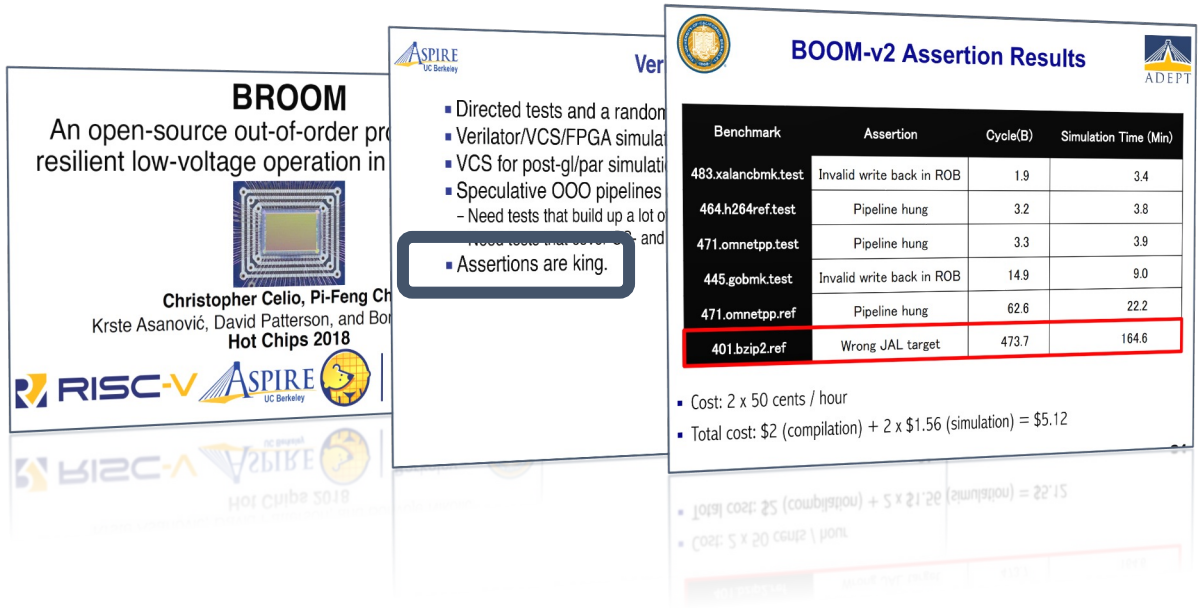
Example use cases: Debugging and Profiling SoC Designs

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





Example use cases: Debugging and Profiling SoC Designs



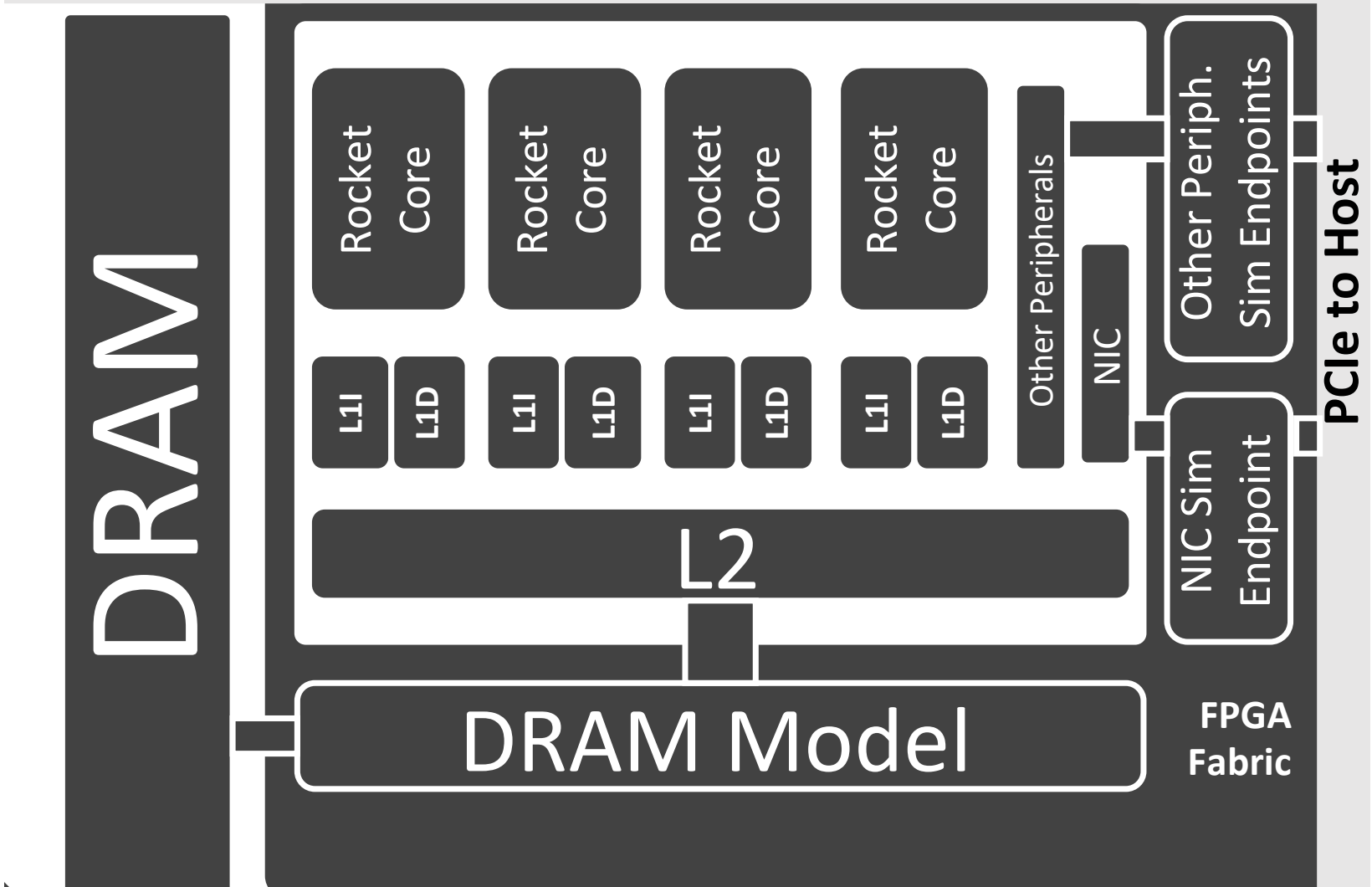
- Debugging a Chisel design at FPGA-speeds
  - Many FireSim debugging features: Assertion synthesis, printf synthesis, ILA insertion, etc.
  - e.g. FireSim Debugging Docs
  - e.g. Fixing BOOM Bugs (D. Kim, et. al., *DESSERT*, FPL '18)
- Profiling a custom RISC-V SoC at FPGA-speeds
  - e.g. HW/SW Co-design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

FirePerf

Scaling from a 150+ MHz Single SoC simulation to 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*

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

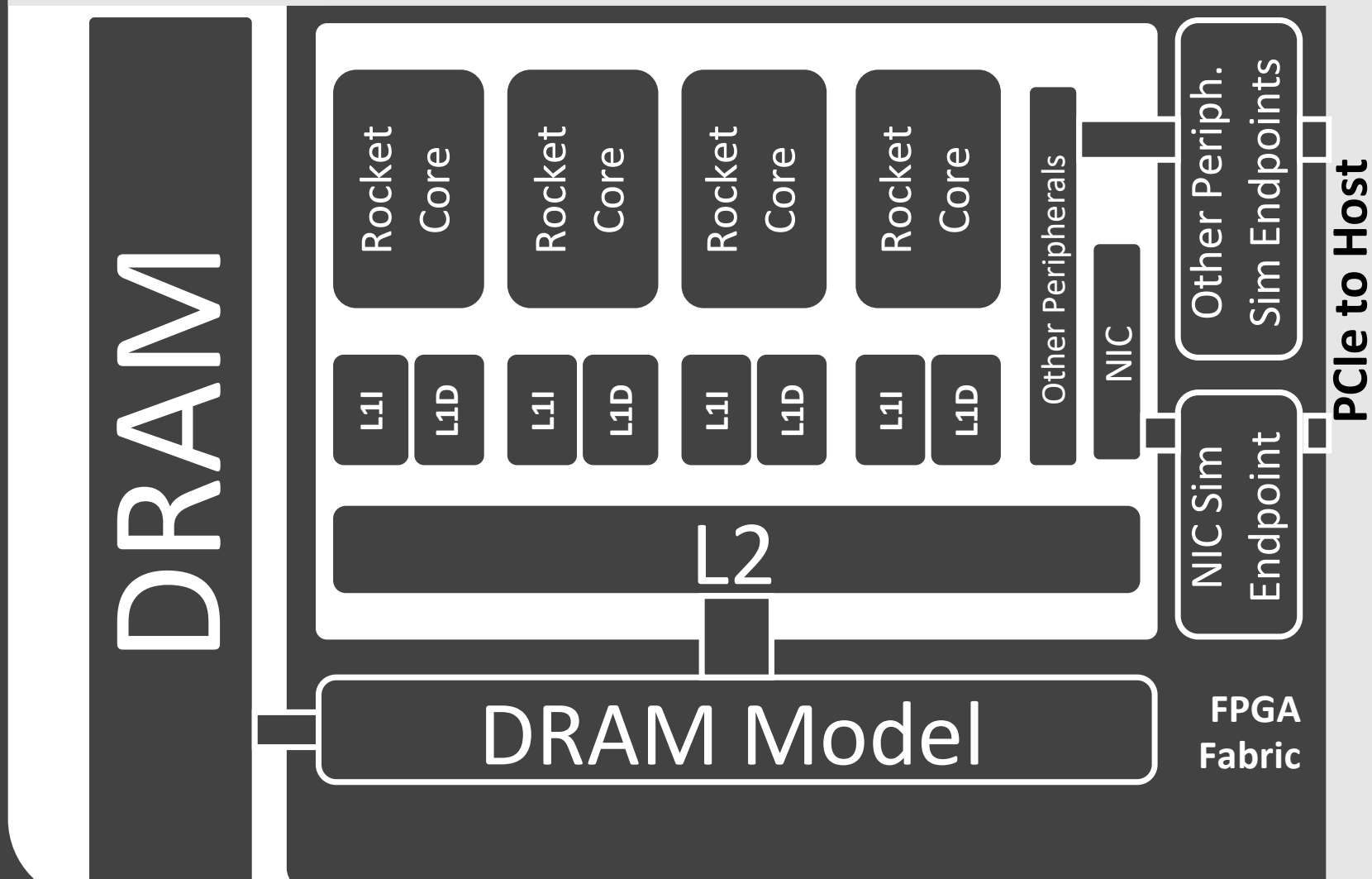
Resource Util.

- < 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

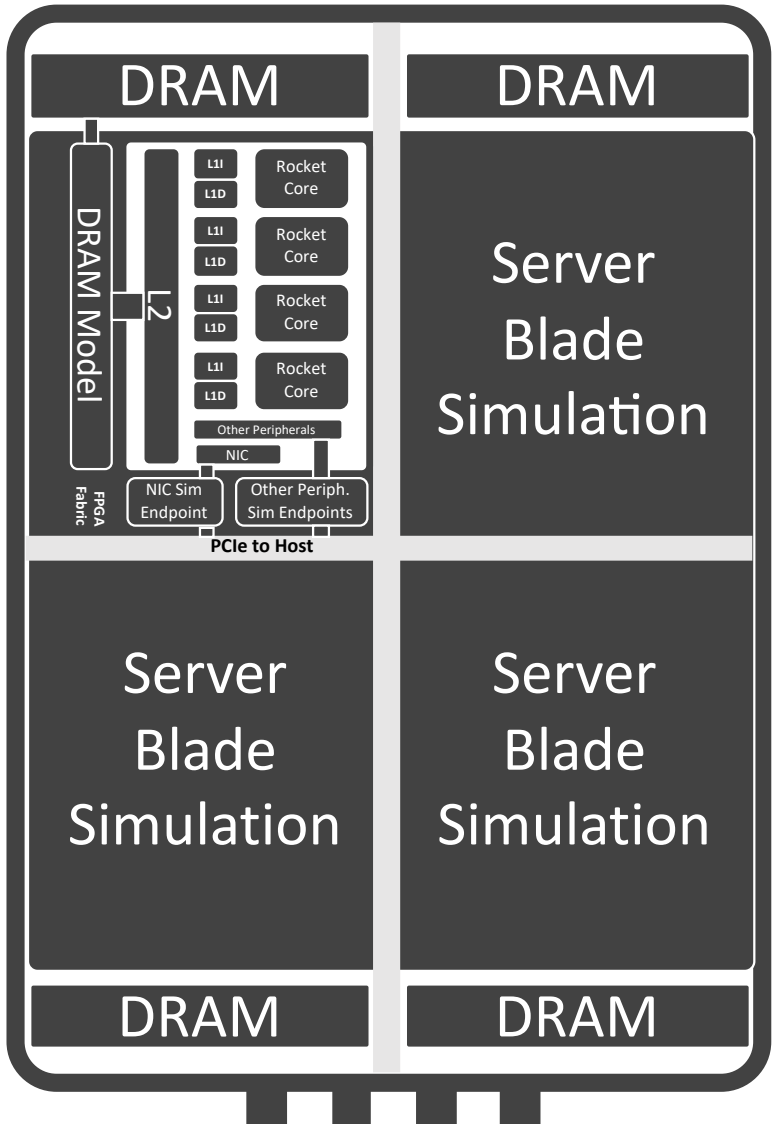
Sim Rate

- ~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)



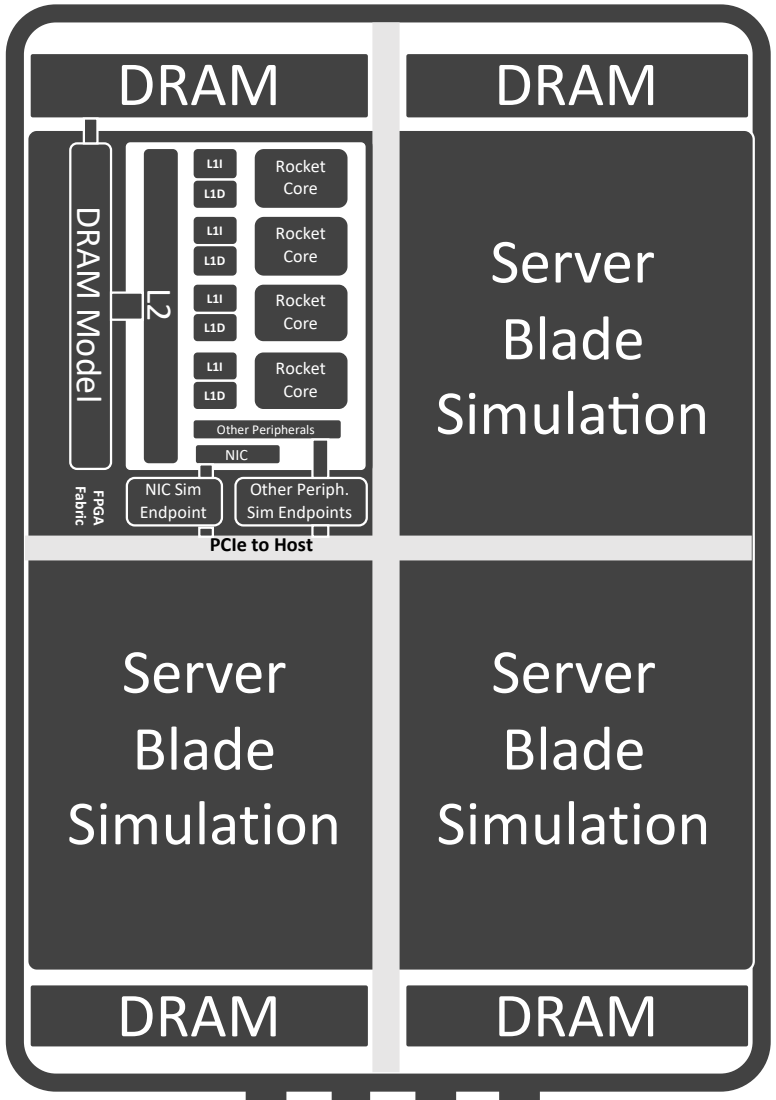
Modeled System
- 4 Server Blades
- 16 Cores
- 64 GB DDR3

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

Sim Rate
- ~14.3 MHz (netw)

Step 3: FPGA Simulation of 4 server blades

FPGA
4 Sims)



FPGA
(4 Sim

Modeled System

- 4 Server Blades
- 16 Cores
- 64 GB DDR3

Resource Util.

- < 1 FPGA
- 4/4 Mem Chans

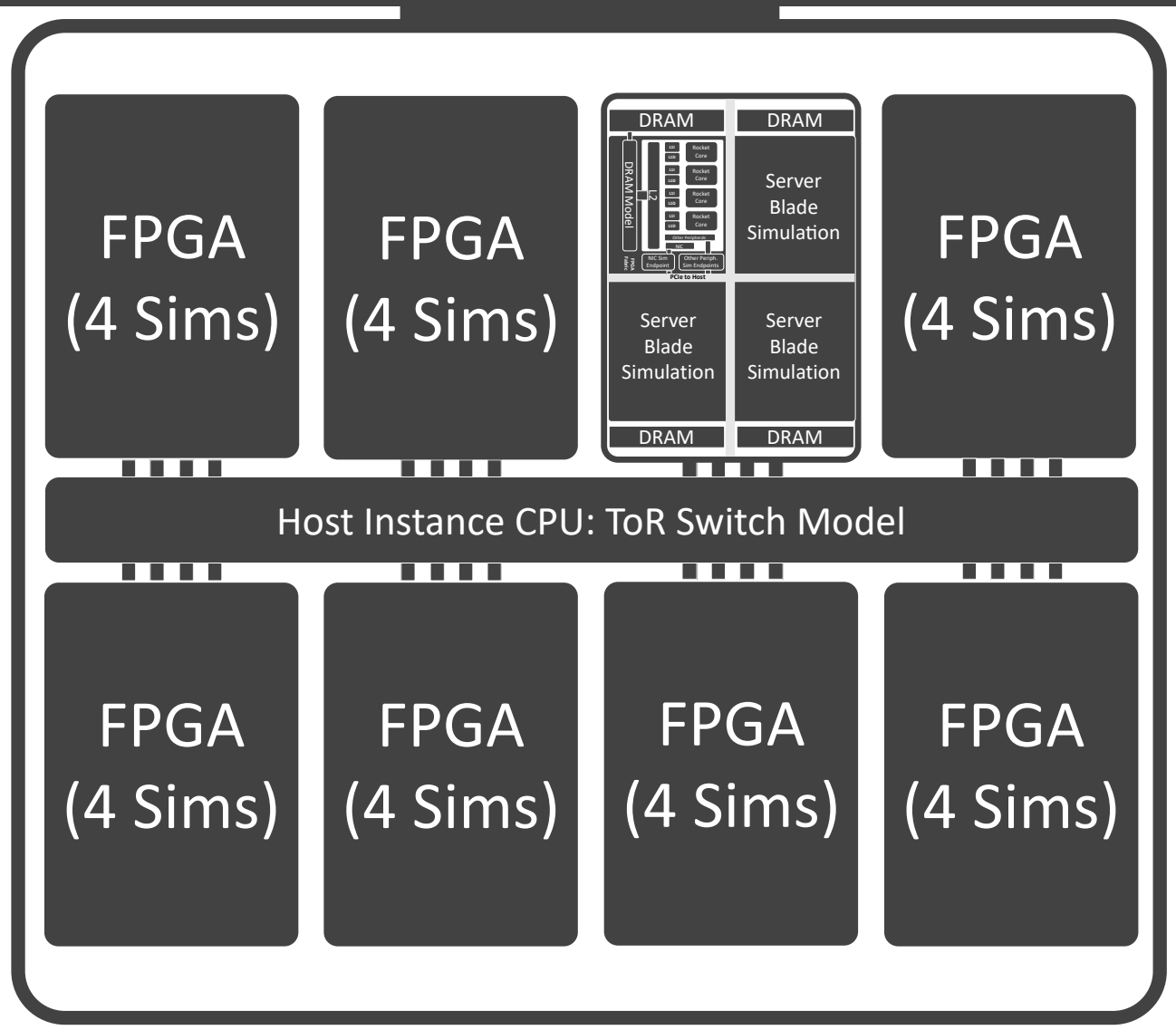
Sim Rate

- ~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

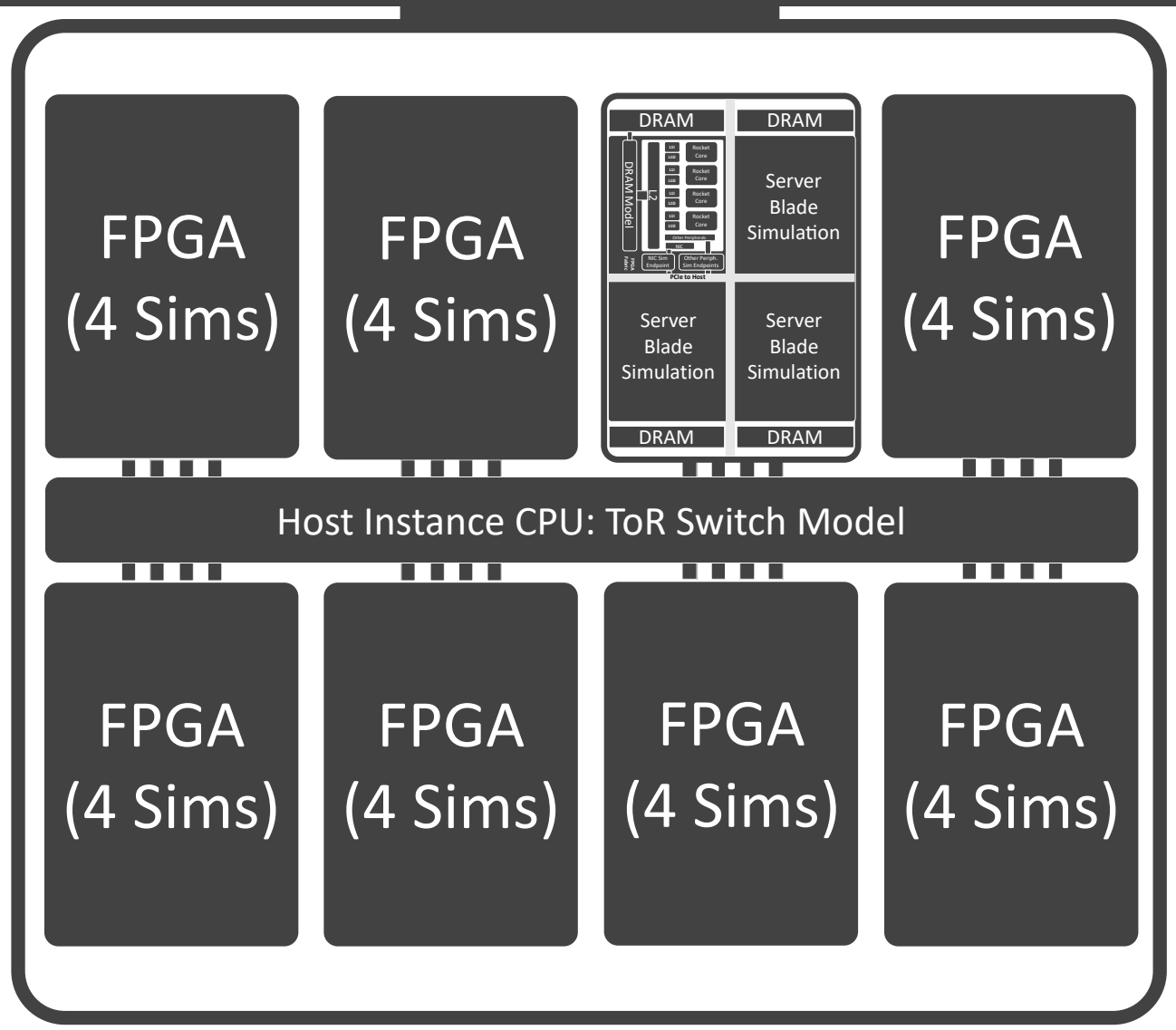
Sim Rate

- ~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

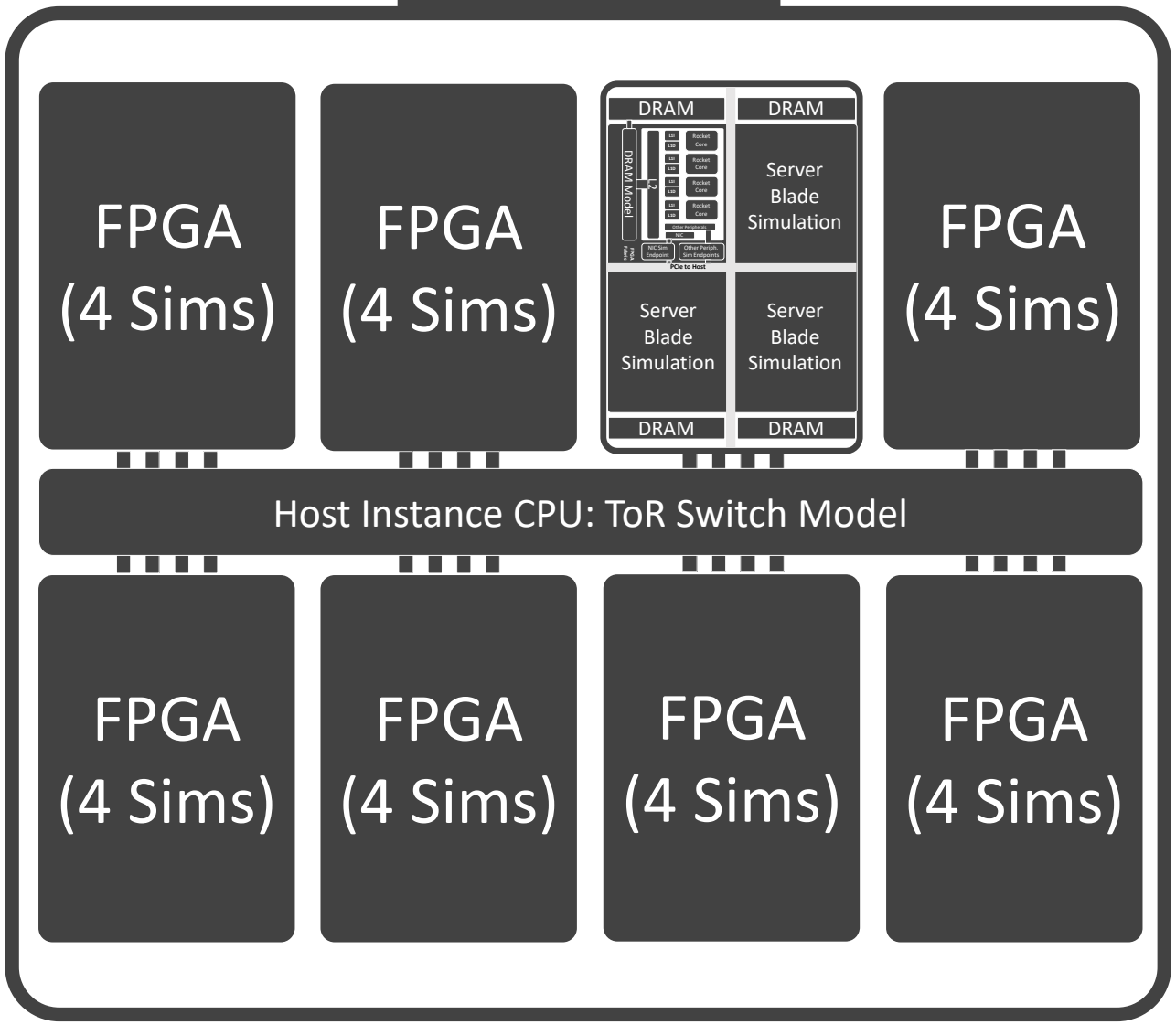
Resource Util.

- 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

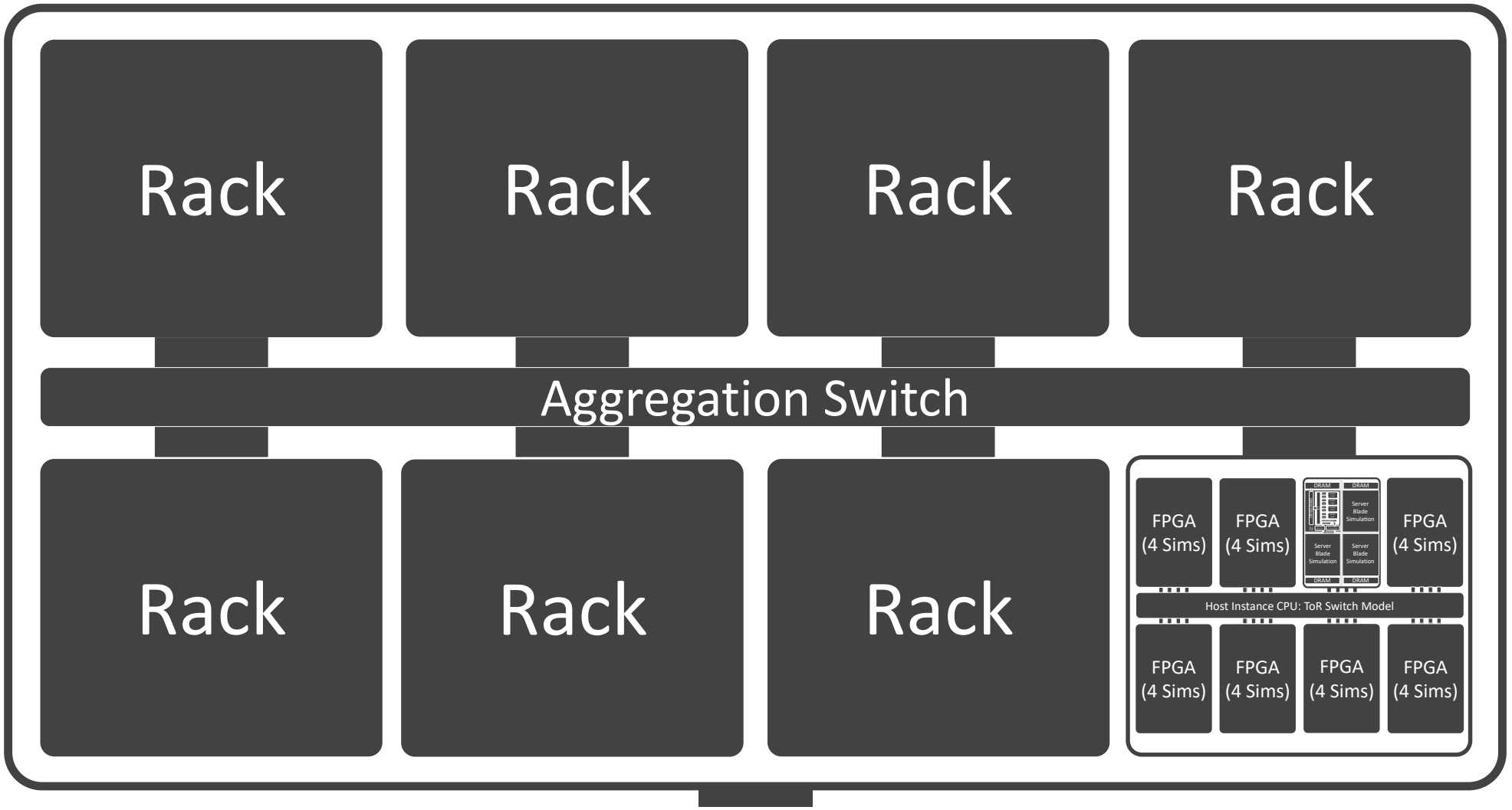
Resource Util.

- 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

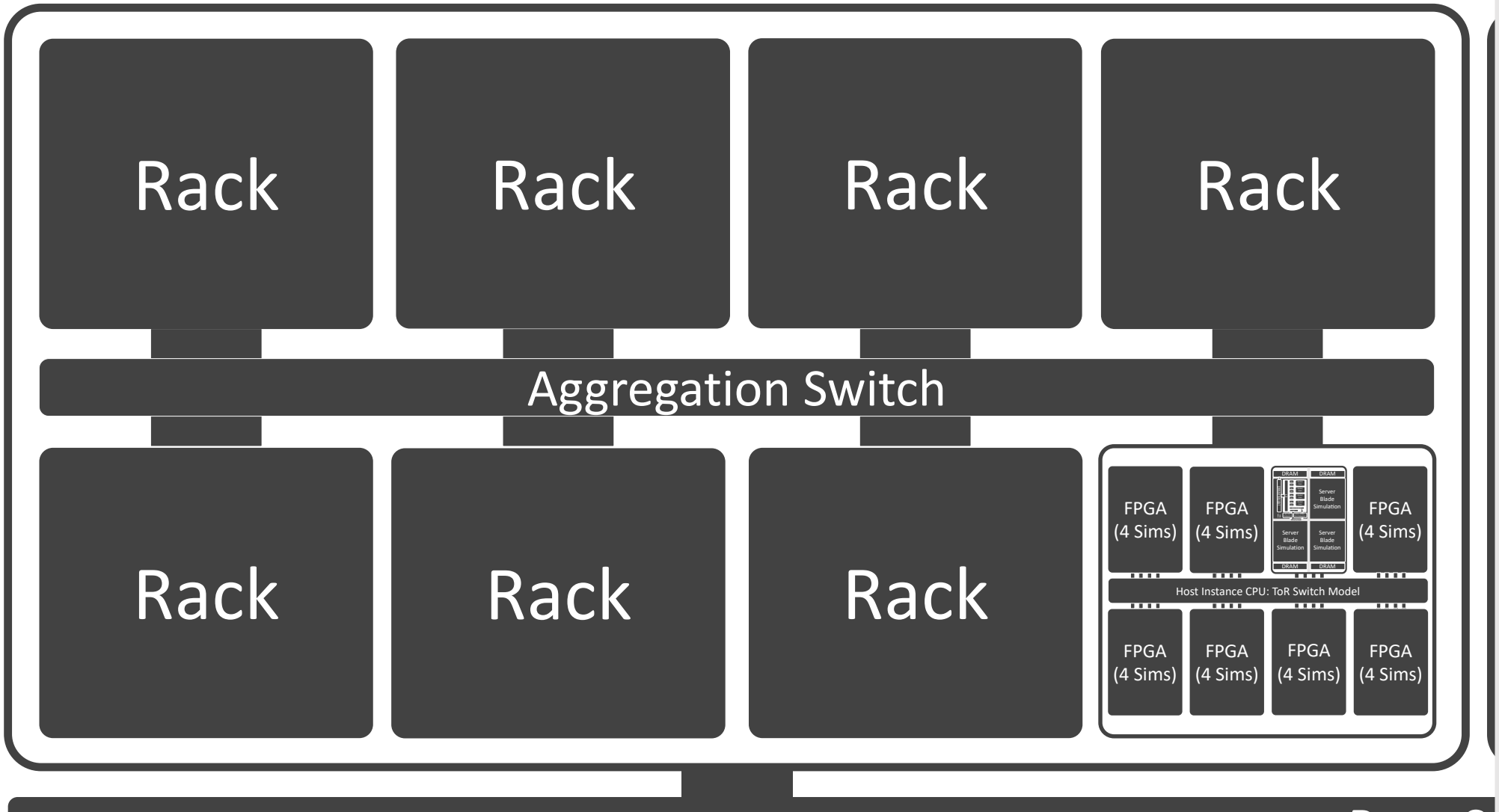
Resource Util.

- 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

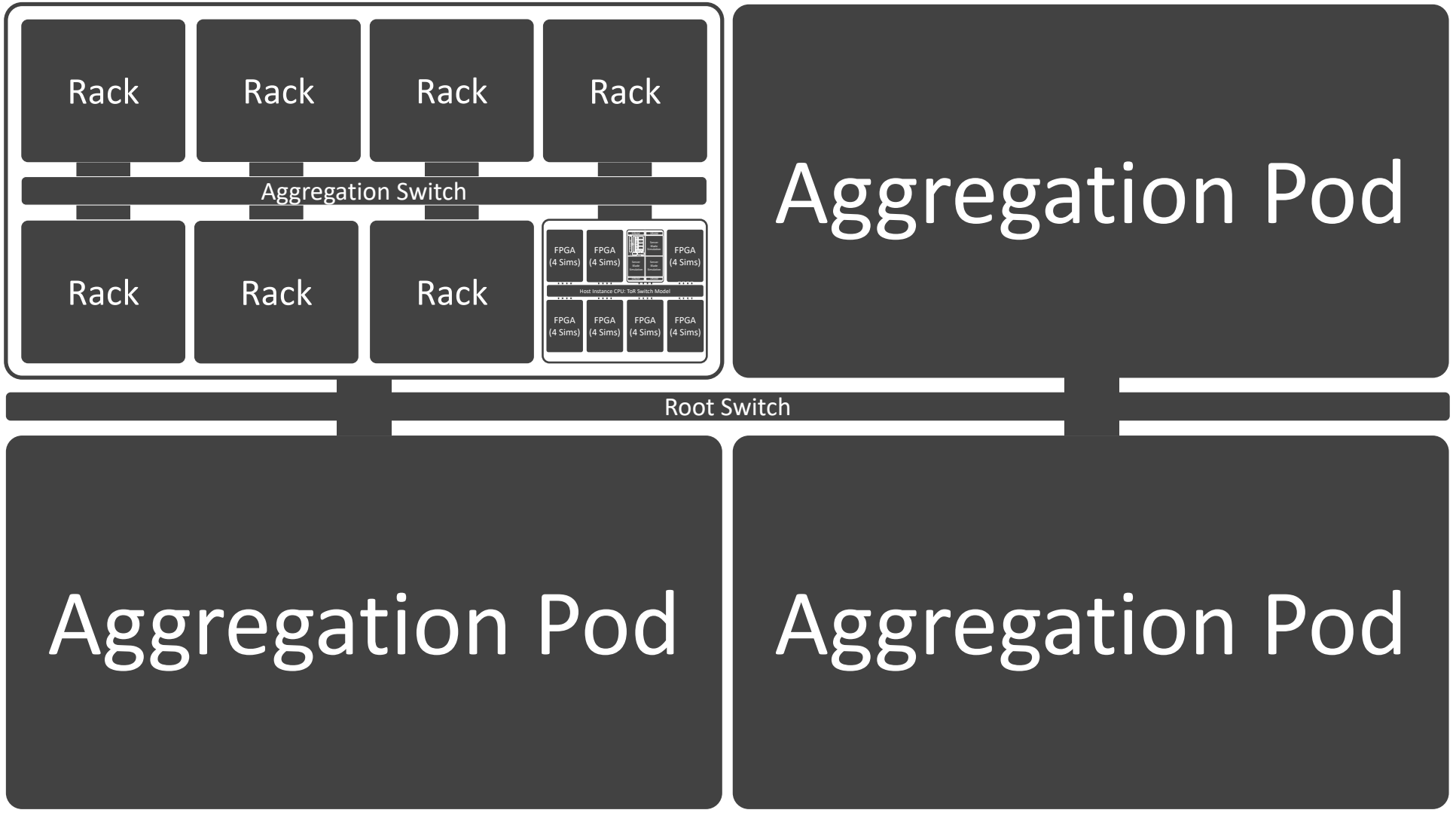
Resource Util.

- 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

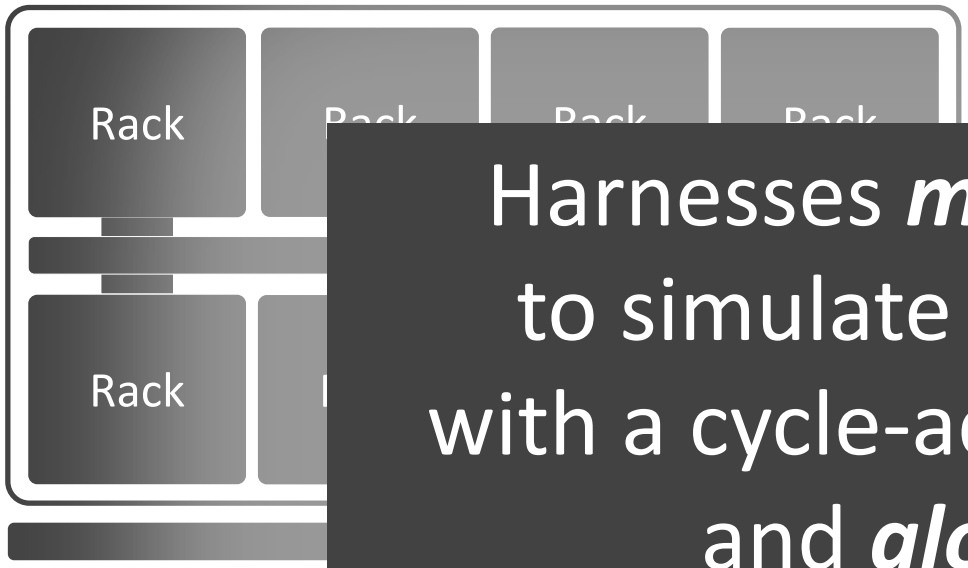
Resource Util.

- 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

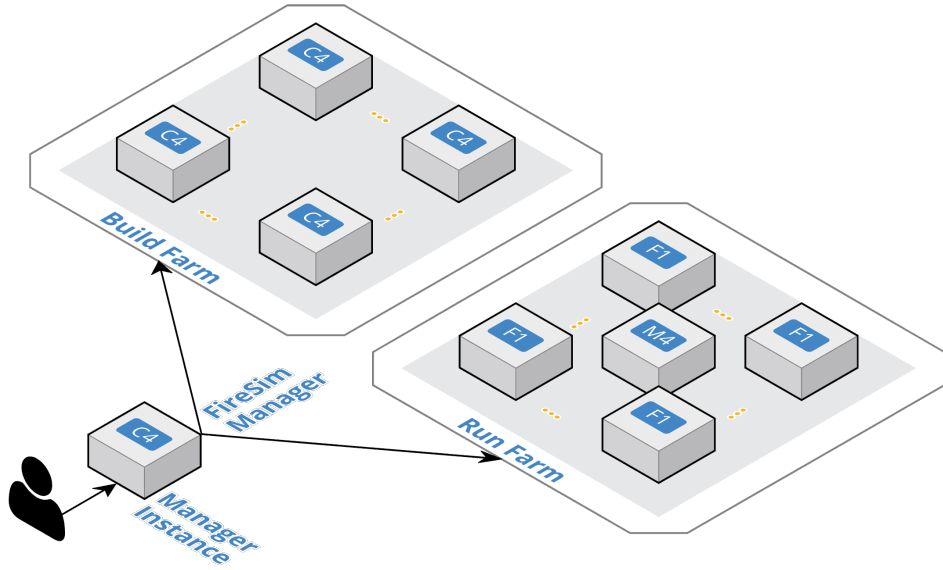
Modeled System
- 1024 Servers
- 6 Cores
- TB DDR3
- oRs, 4 Aggr, 1
- Gb/s, 2us

urce Util.
- 256 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

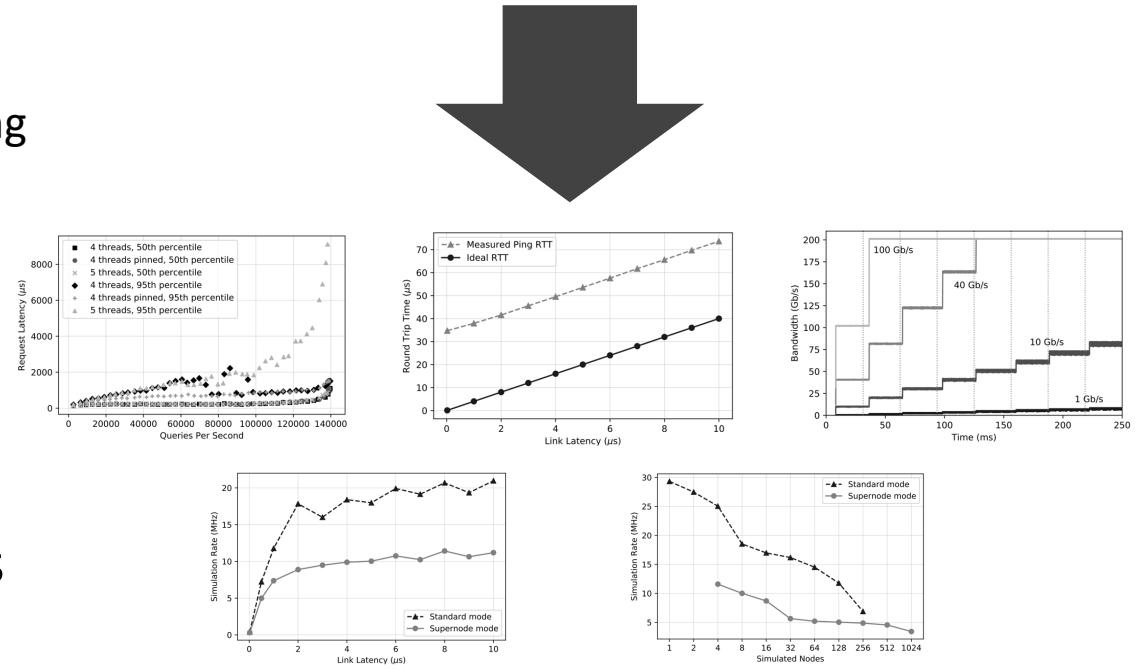


FireSim Developer Environment

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 with full **reference inputs** in ≈1 day
  - Many others included
  - Several user papers have gone through artifact evaluation using FireSim (*nanoPU*, *FirePerf*, *Protobuf accel.*, *MoCA*, *Simulator Independent Coverage*, etc.)
- 200+ pages of documentation: https://docs.fires.im
- 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
```



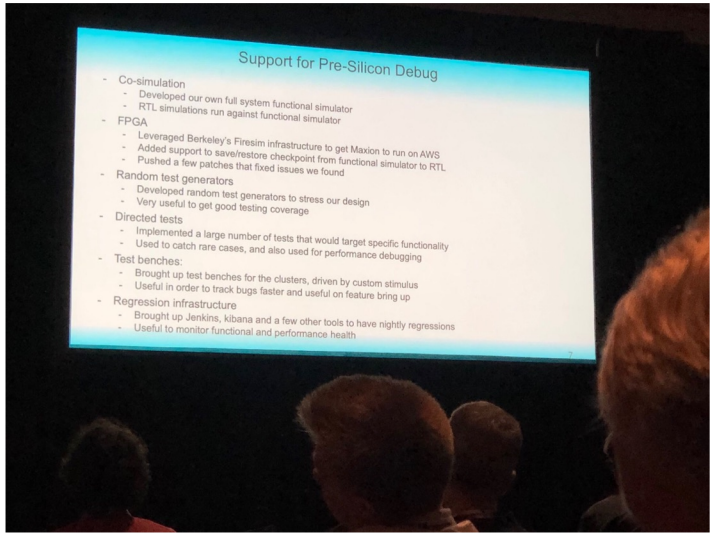
On-premises FPGA support now available!

- High-level of automation/reproducibility enabled by FireSim on AWS F1 cloud now extended to local/on-prem FPGAs:
  - Went from new machine with no FPGA attached to working FPGA-accelerated simulation in 1 hour and 40 mins
- Use existing FireSim features at-scale and locally!
  - Cycle-accurate simulation
  - Debugging
    - Integrated logic analyzers, trace dumps, synth. assert/prints, co-simulation
  - Software support
    - FireMarshal workload management
  - … and more!

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

- More than 200 mailing list members and 850 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/nvdla/
  - 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 (DARPA SSITH/FETT)



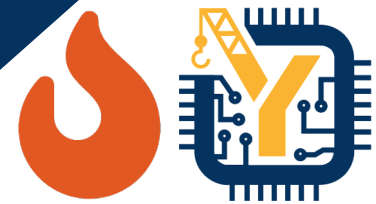
Esperanto announcement at RISC-V Summit 2018

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**)
- **USENIX Security ‘21**: Saileshwar et. al. MIRAGE (**Georgia Tech**)
- **CCS ‘21**: Ding et. al. “Hardware Support to Improve Fuzzing Performance and Precision” (**Georgia Tech**)
- **MICRO ’21:** Karandikar et. al. “A Hardware Accelerator for Protocol Buffers” (**Berkeley/Google**)
- **MICRO ‘21**: Gottschall et. al. TIP (**NTNU**)
- **Over 20 *additional* user papers on the FireSim website:**
  - 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**: Zhang et. al. “Composable Building Blocks to Open up Processor Design” (**MIT**)
- **RTAS ‘20**: Fa
- **EuroSys ‘20**:
- **OSDI ‘21**: Iba
- **USENIX Secu**
- **CCS ‘21**: Din
  Performance
- **MICRO ’21:** k
  Buffers” (**Berkeley/Google**)
- **MICRO ‘21**: Gottschall et. al. TIP (**NTNU**)
- **Over 20 *additional* user papers on the FireSim website:**
  - https://fires.im/publications/#userpapers

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

**actually used, not only cited*

- Awards: FireSim ISCA ‘18 paper:
  - IEEE Micro Top Pick
  - CACM Research Highlights
  - Nominee from ISCA ‘18
- s:
- al.:
- l et. al.:
  - er runner-up
- ar et. al.:
  - shed Artifact winner
  - IEEE Micro Top Pick Honorable Mention
- DAC ‘21 Genc et. al.:
  - DAC 2021 Best Paper winner

New to FireSim/Chipyard?

- Check-out the tutorials!
  - A full-day hands-on introduction to FireSim and Chipyard on AWS EC2 F1
  - https://fires.im/tutorial/
- Slides from yesterday's tutorial are already available
- We'll post videos and getting started scripts tomorrow

Let’s get started!

Program: https://fires.im/workshop-2023/
Workshop Slack: https://fires.im/workshop-slack/