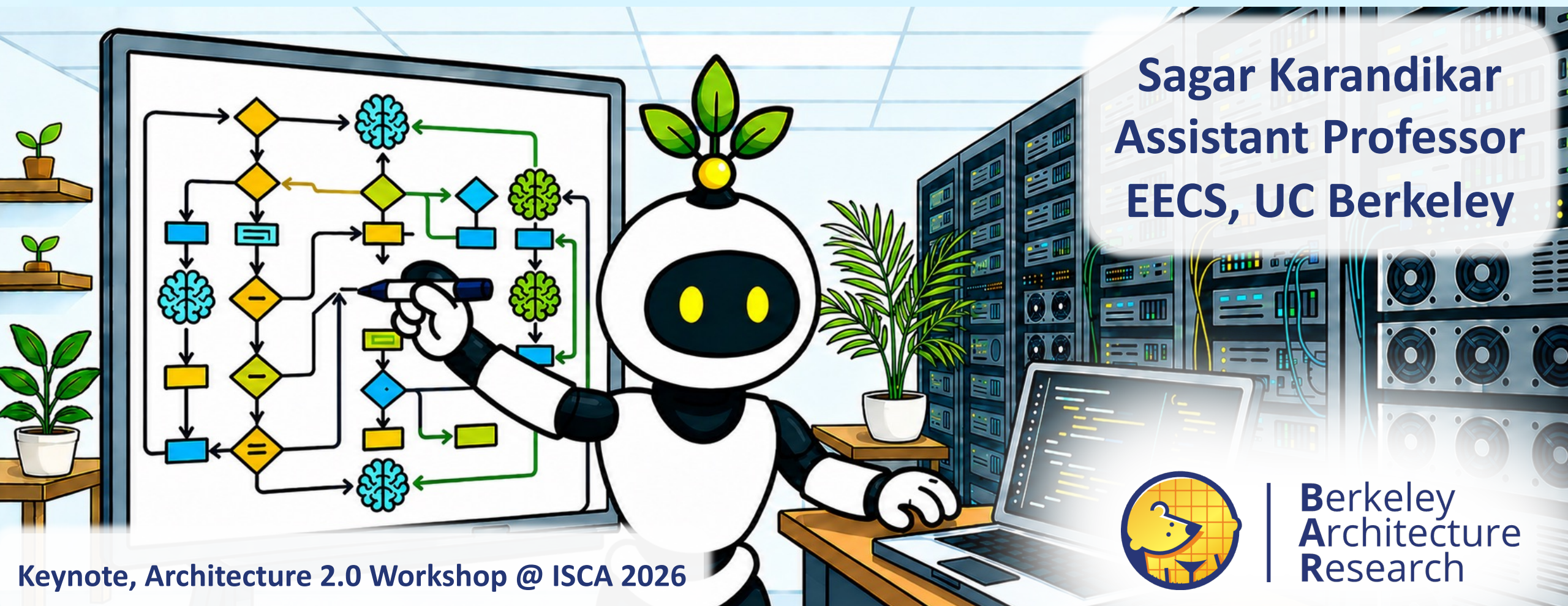


CHIA

<https://chialoops.ai>

An open-source framework for
principled, agentic AI-driven
HW/SW co-design research



Sagar Karandikar
Assistant Professor
EECS, UC Berkeley



Berkeley
Architecture
Research

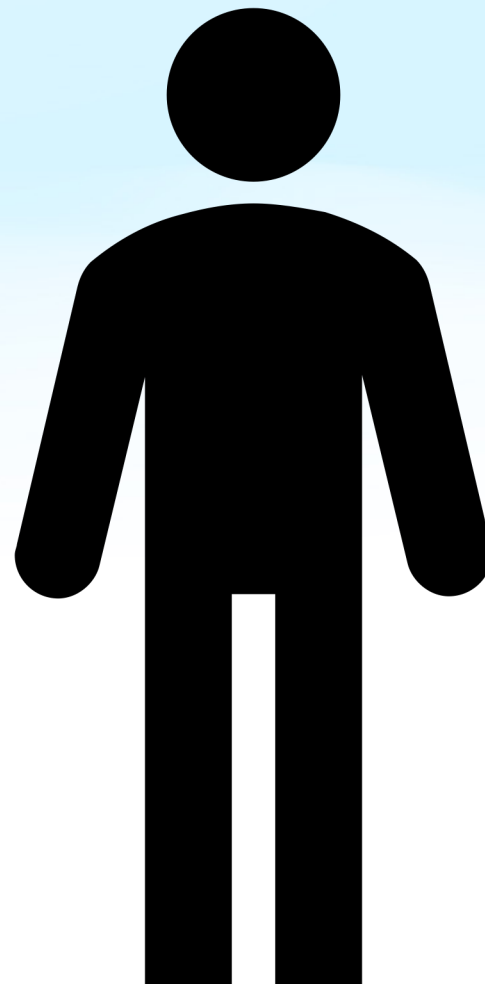
Keynote, Architecture 2.0 Workshop @ ISCA 2026

Me, in the dark ages

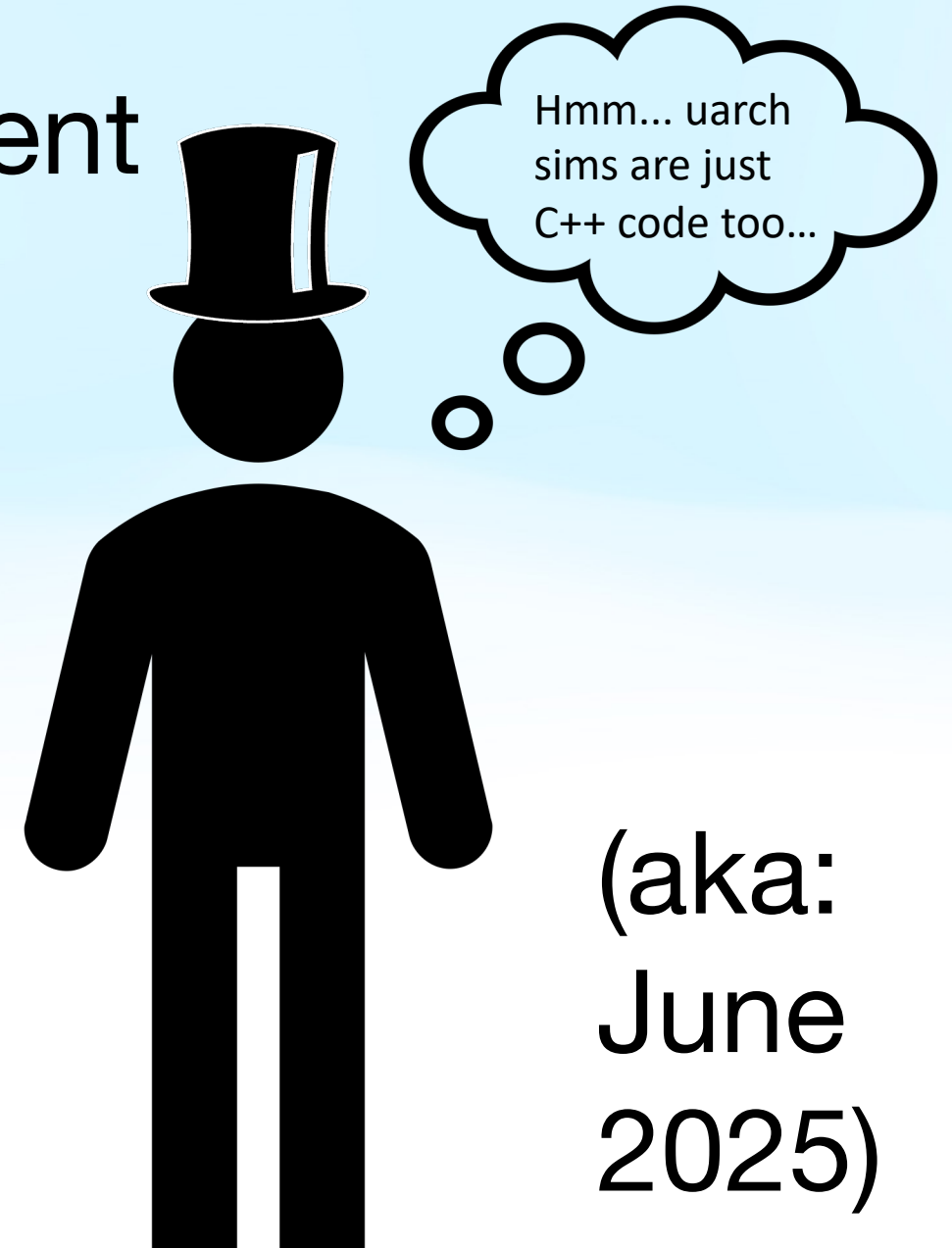
(aka: May 2025)



Me, in age of enlightenment



Me, in age of enlightenment



Google DeepMind

AlphaEvolve: A coding agent for scientific and algorithmic discovery

Alexander Novikov*, Ng  n V  *, Marvin Eisenberger*, Emilien Dupont*, Po-Sen Huang*, Adam Zsolt Wagner*, Sergey Shirobokov*, Borislav Kozlovskii*, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli and Matej Balog*
Google DeepMind¹

<https://arxiv.org/abs/2506.13131>

ArchAgent: An Agentic AI-based Approach to Computer Architecture Discovery

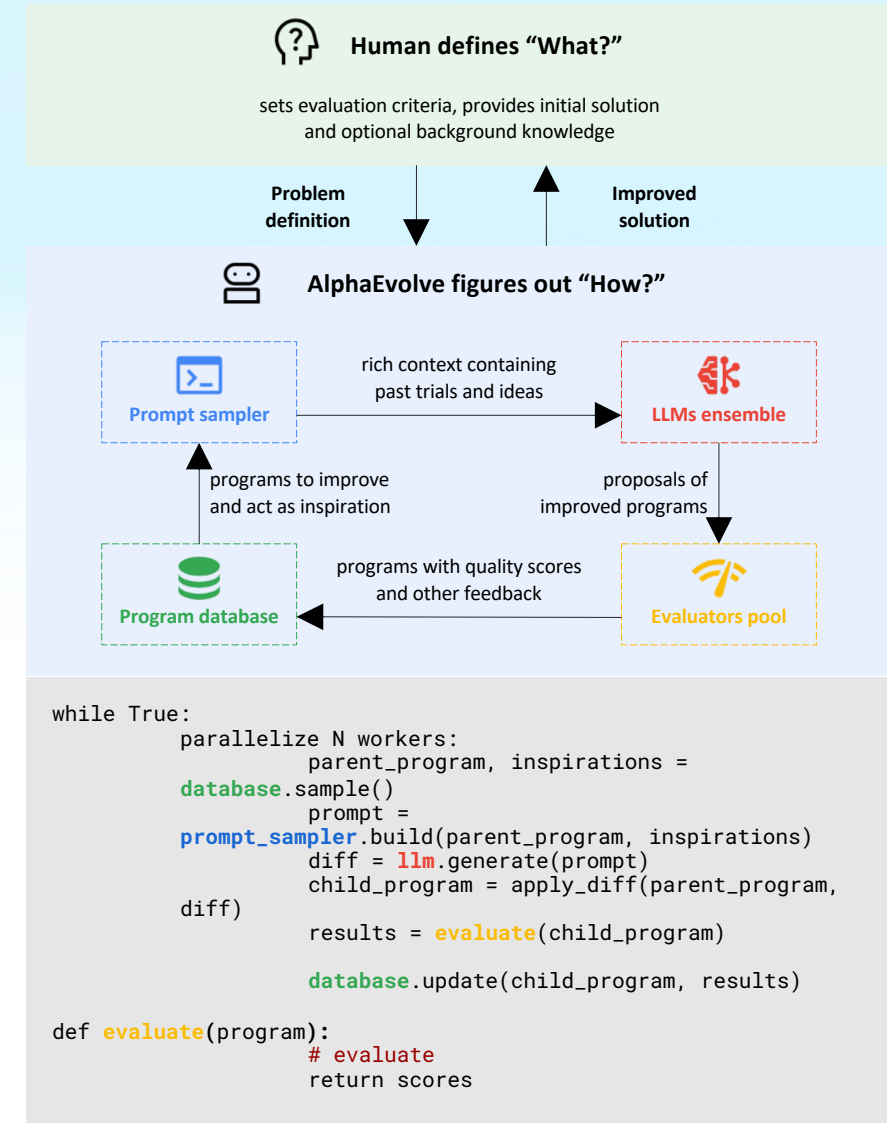
Raghav Gupta¹, Akanksha Jain², Abraham Gonzalez², Alexander Novikov³, Po-Sen Huang³, Matej Balog³, Marvin Eisenberger³, Sergey Shirobokov³, Ngân Vũ³, Martin Dixon², Borivoje Nikolić¹, Parthasarathy Ranganathan², Sagar Karandikar¹

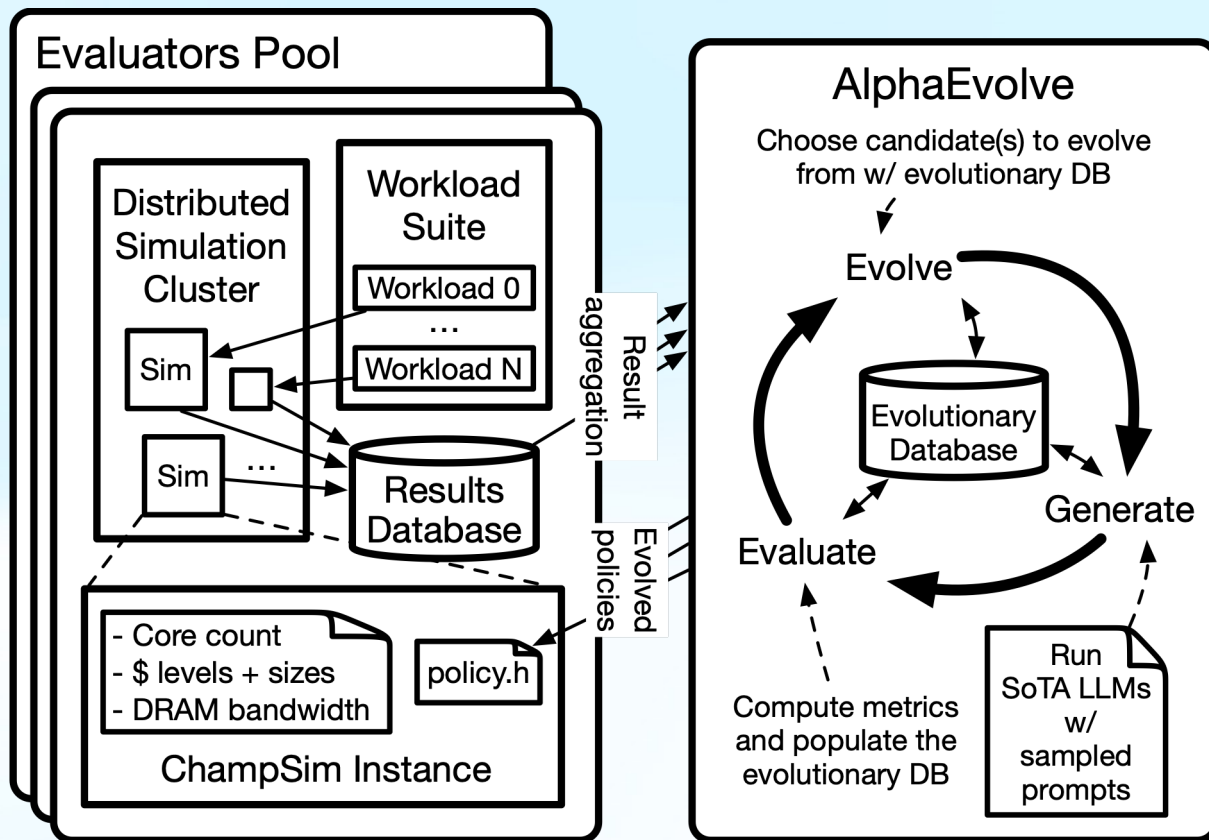
UC Berkeley¹ / Google² / Google Deepmind³

OpenReview, Nov. 2025: <https://openreview.net/forum?id=SVIXvewImv>

AlphaEvolve: LLM-Driven Evolutionary Search

- An evolutionary coding agent for scientific and algorithmic discovery
- LLM-Driven
 - Semantic awareness
 - Cross-domain knowledge
- Evolutionary search
 - Stochastic + stateful
 - Islands
 - Independent sub-populations/lineages
 - Periodic reset
- MAP-Elites
 - Global best on each fitness metric





We can use AI agents to automatically discover new (micro)architectural mechanisms!

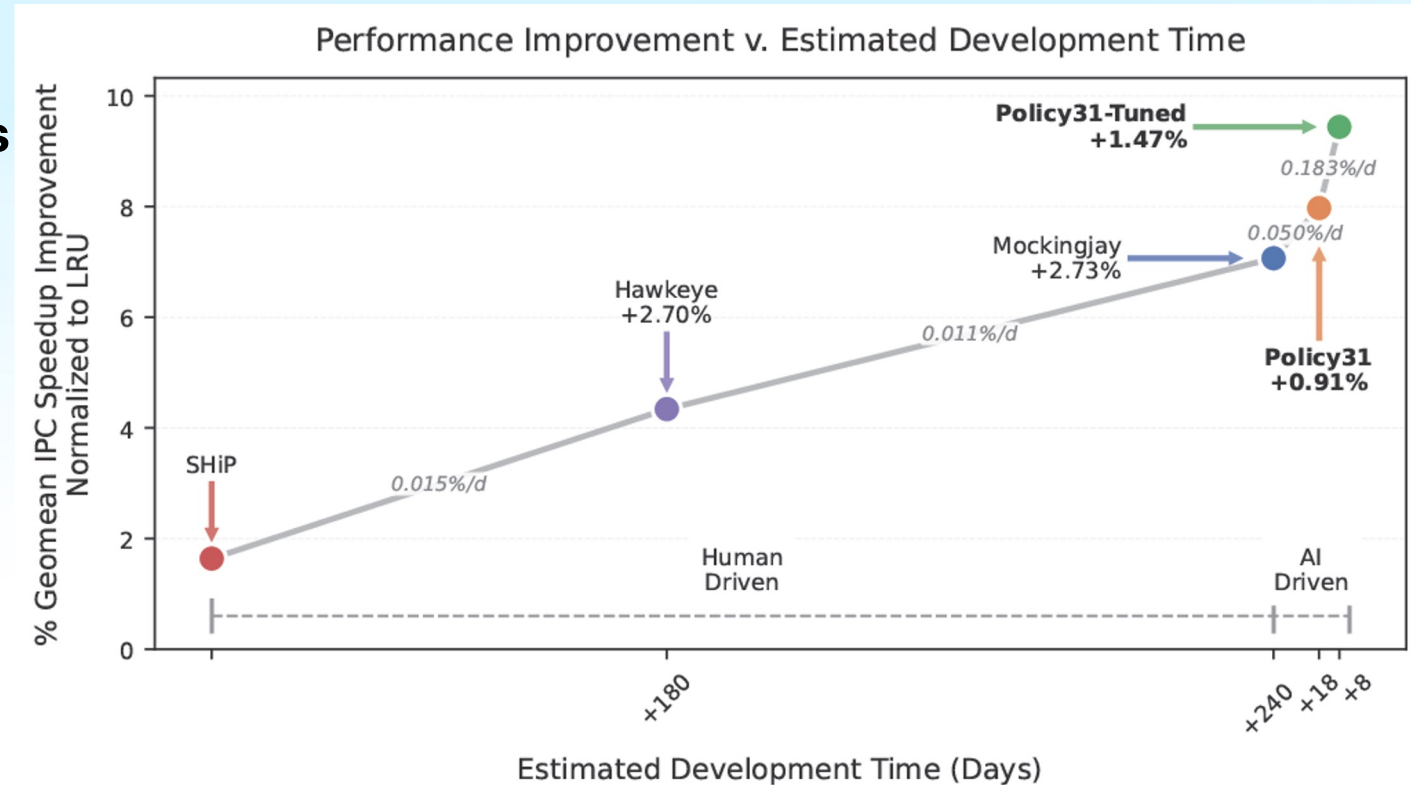
Single-Core SPEC06

Policy31

- **0.9% IPC speedup improvement vs. Mockingjay (prior SoTA) within 18 days**
- **3-5x rate of improvement v human designed SoTA policies**

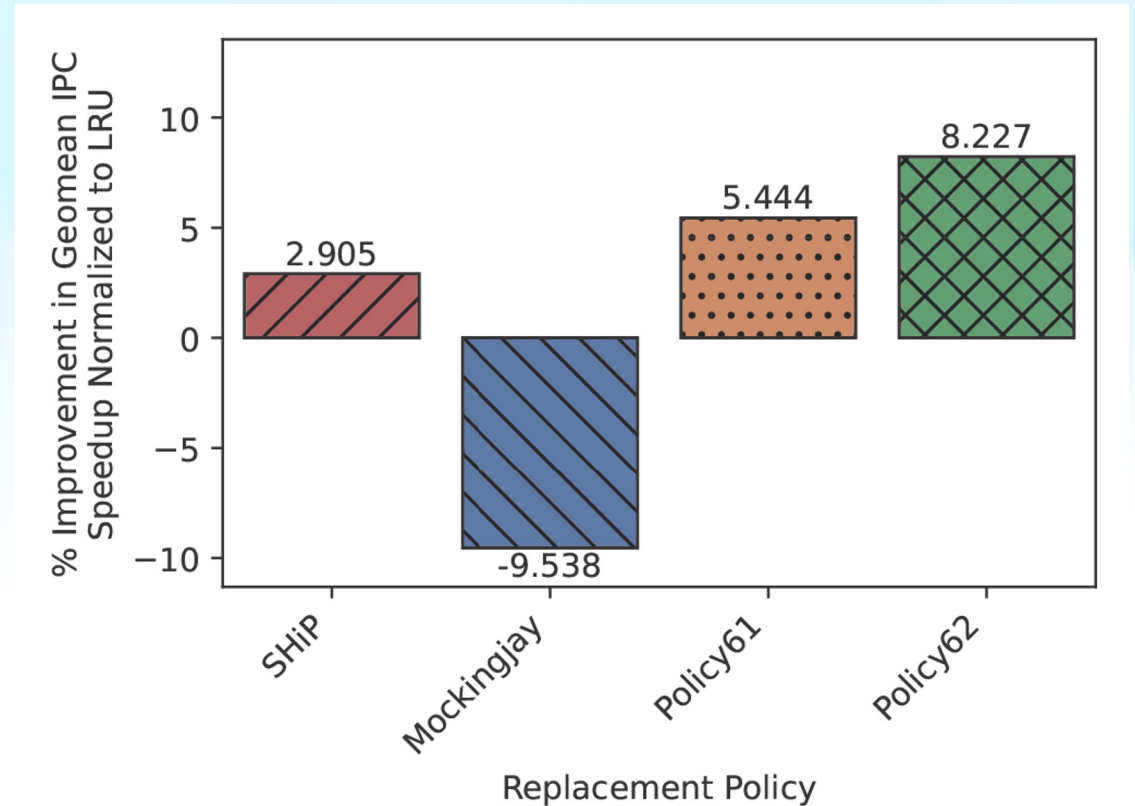
Policy31-Tuned

- Can agentic AI do “PGO” for runtime parameters of a policy, per-workload, on-the-fly?
- **2.4% IPC speedup improvement v Mockingjay in 8 more days**



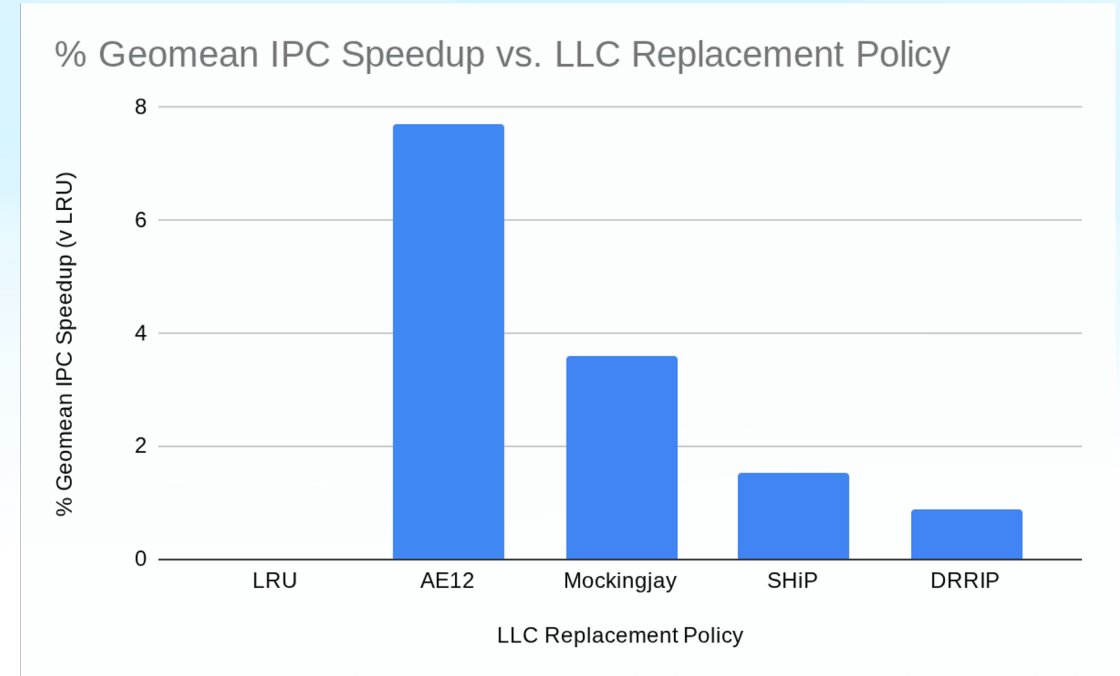
Multi-Core Google Workload Traces v2

- Prior SoTA on SPEC collapses on hyperscale workloads
- ArchAgent evolves **Policy62** in **2 days**
- **5.3% IPC speedup improvement** v SHiP (+17.8% v Mockingjay)
- Workload specialization of hardware in a matter of days!



AI Can Break Comp Arch Simulators

- Early experiment: wins against SoTA on SPEC06 with over 4% IPC speedup!!!!
 - (~1% = paper acceptance at a top computer architecture conference for cache replacement)
- Simulator does not support write-bypassing → assertions disabled in optimized simulator build → AI solution drops write traffic with no visible error → IPC 📈
- Correctness of evaluation setup is critical. AI can easily reward hack ($\approx$ cheat).



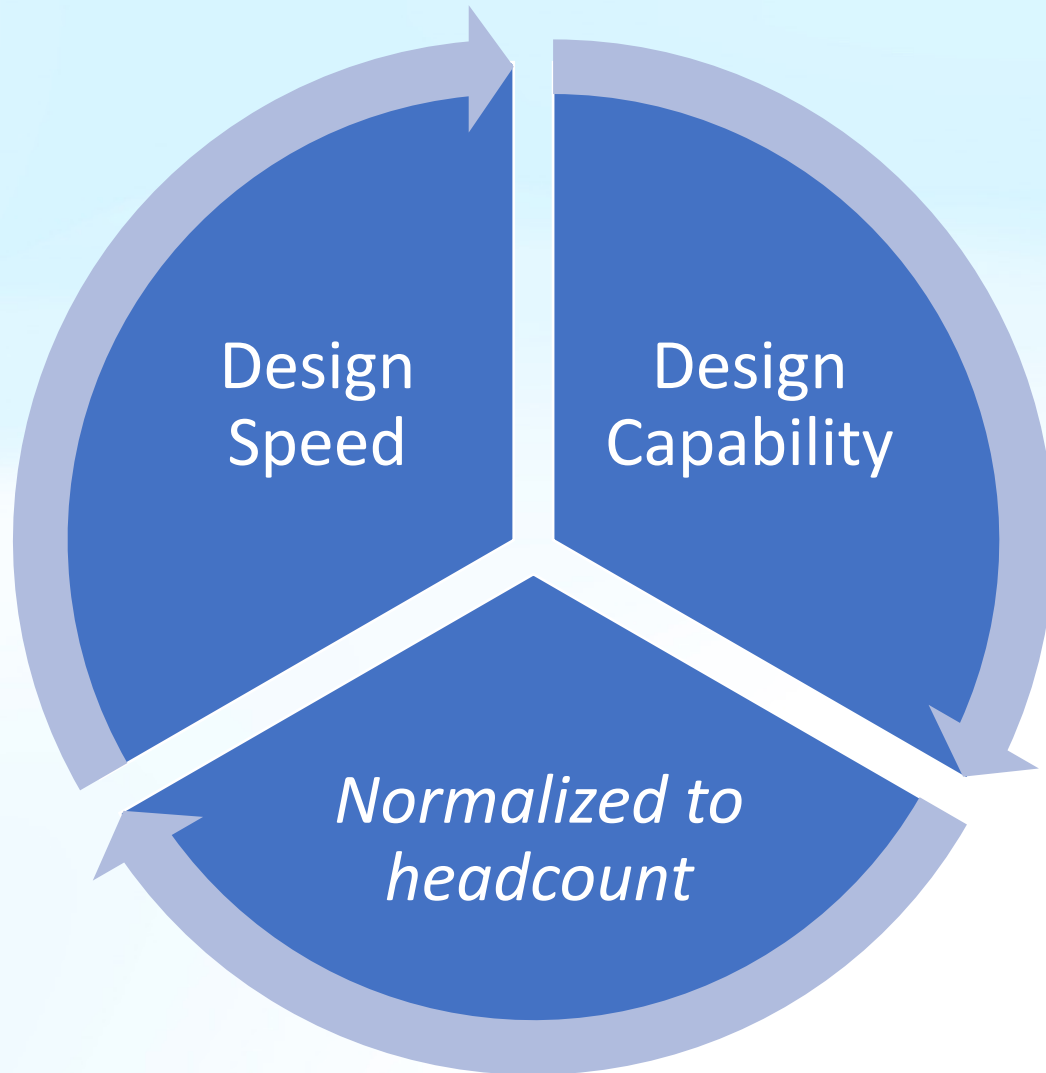
Challenges

- The whole point of applying AI: We want the computers to be doing the work for us
 - High-level uarch sim won't cut it as the evaluation endpoint
- Many proposed evolutionary flows/algorithms
 - Algorithmic differences are interesting
 - But the flows are static; most expect one contained SW binary to eval
- Immense engineering/"good science" challenges for long-running evals (i.e., all of HW/SW co-design)
 - Other areas have IMMENSE work on validating the sims they are using in these areas

What can save us?

Agile hardware design

Agile HW/SW co-design in a nutshell



2010-2025:
Apply **software
engineering best-
practices**
to HW/SW co-
design.

The last 15 years of agile design

1. High-level Simulation: understand the domain
2. Write Register-Transfer Level (RTL) “code” for HW design + corresponding SW
3. Co-design in software RTL simulation (e.g., Verilator, VCS, etc.)
 - Can only run microbenchmarks. OS boot takes days/weeks.
4. Co-design on FPGA
 - Boot an OS, run complete SW, get perf. measurements
5. Tapeout to get a chip
 - Can boot OS and run applications, but no more opportunity for co-design

CHISEL

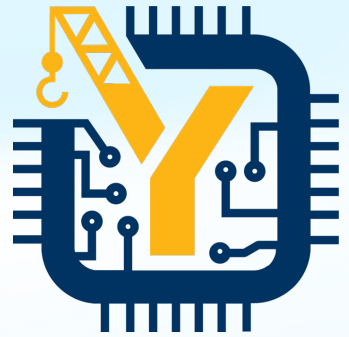
FIRRTL

IR
T

FireSim

HAMMER

The last 15 years of agile design



1. High-level Simulation: understand the domain
2. Write Register-Transfer Level (RTL) “code” for HW design corresponding SW
3. Co-design in software RTL simulation (e.g., Verilator, VCS)
 - Can only run microbenchmarks. OS boot takes days/weeks.
4. Co-design on FPGA
 - Boot an OS, run complete SW, get perf. measurements
5. Tapeout to get a chip
 - Can boot OS and run applications, but no more opportunity for





(a four-slide overview)

An Agile,
Generator-driven,
RISC-V SoC
Development
Framework

<https://github.com/ucb-bar/chipyard>

IEEE Micro 2020.4, DAC '20 (invited)

Widely-used Open-Source Project:

Used in 150+ pubs from 65+ institutions

Used in over 20 tape-outs at multiple academic institutions



Chipyard Use Cases

I have an idea for the next big hardware widget but I need to evaluate it in a **complete system** and **fast!**

You

Library of HW/SW components that are flexible and battle-tested

RTL Simulation running test binaries/micro-benchmarks

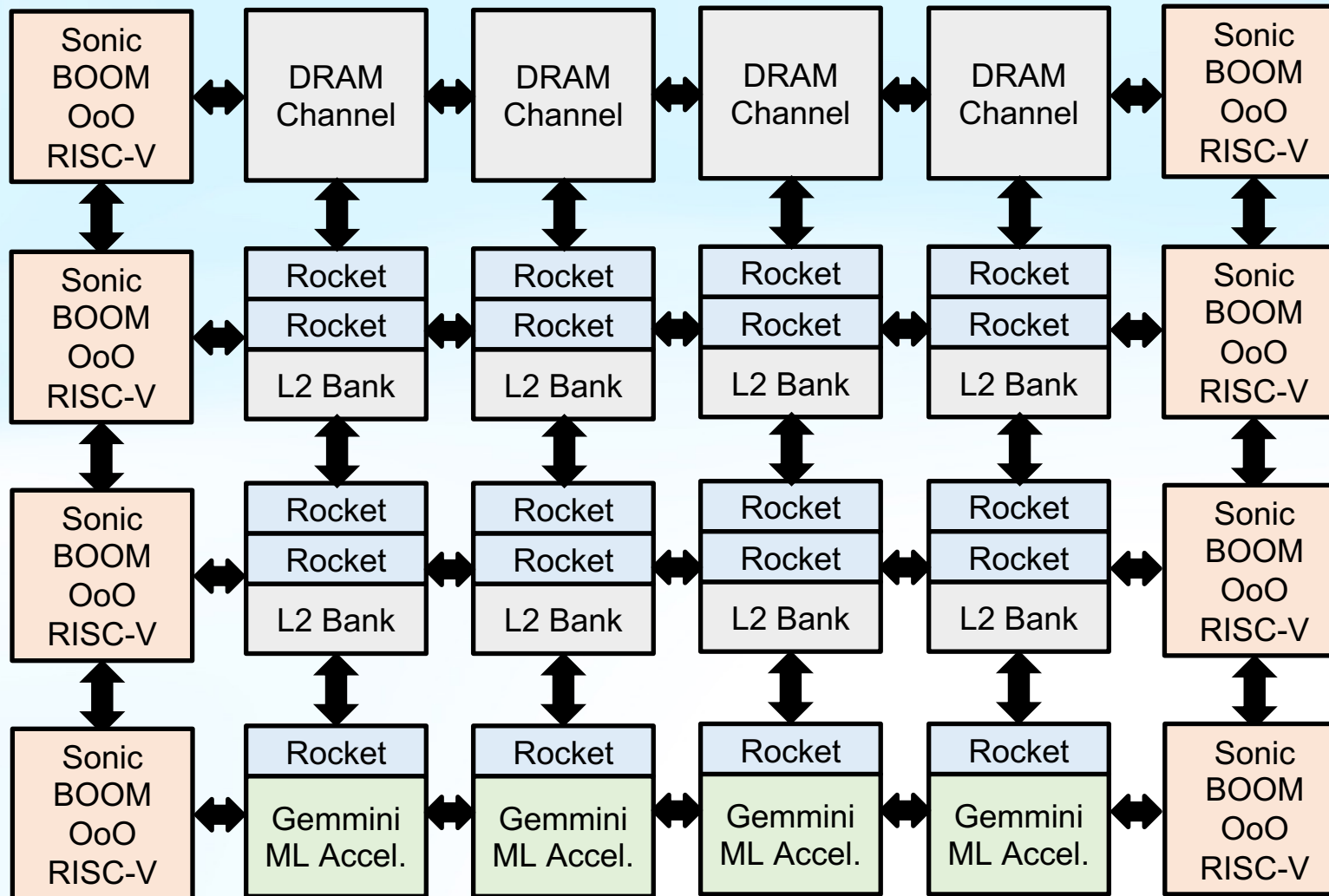
FPGA-accelerated simulation running full workloads

FPGA prototyping for chip bring-up / FPGA system development

Tape-out the SoC to get actual silicon results



An Example Chipyard-generated SoC



- 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

Not hypothetical!

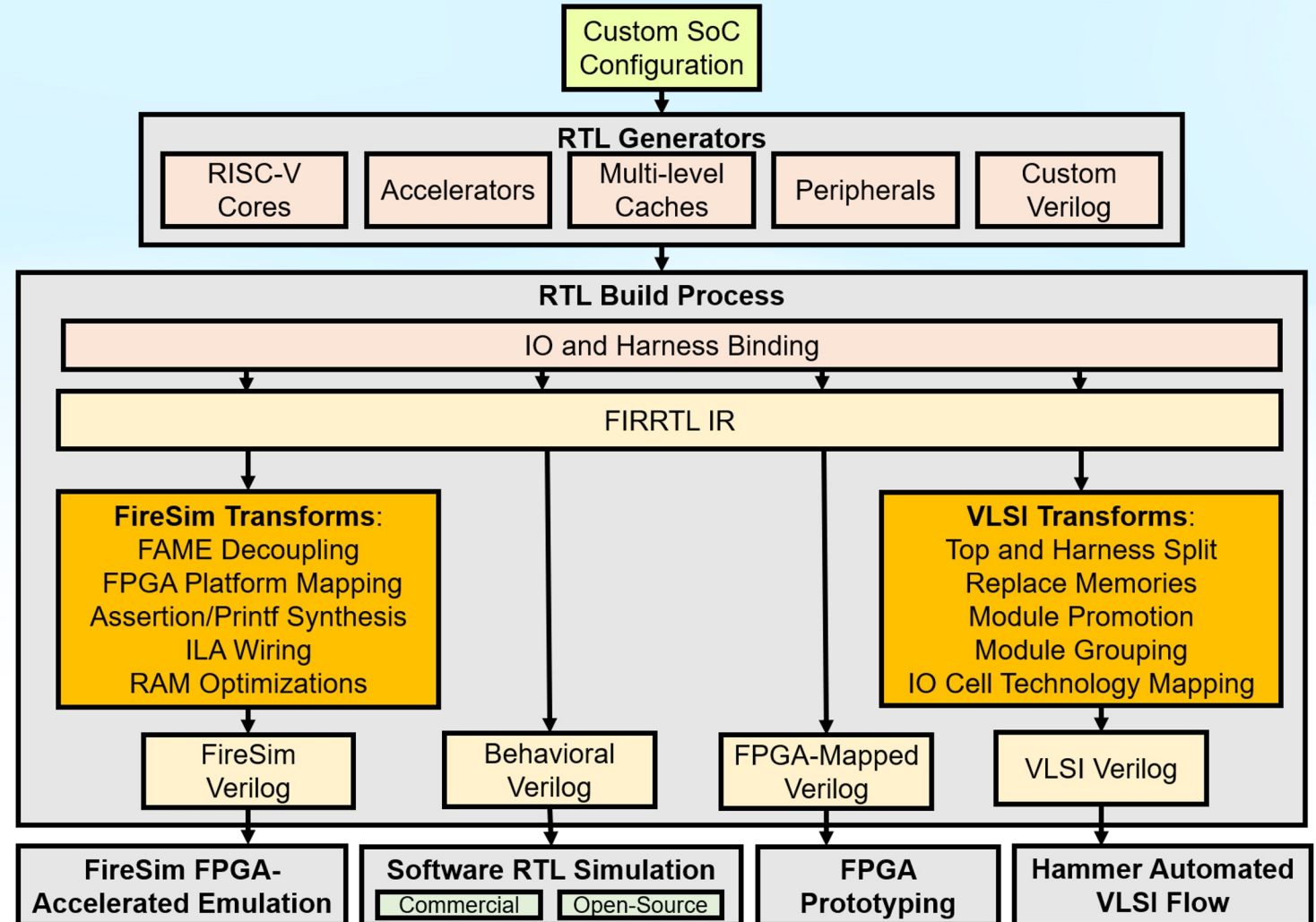
Modified, built, and simulated by tutorial attendees



Chipyard Organization

What is Chipyard?

- An organized **framework** for various SoC design tools and software
- A **curated IP library** of open-source, generator-based RISC-V SoC components + compatible software
- A **curated library of software** (e.g., RISC-V Ubuntu Linux) compatible with generated hardware
- A **methodology** for agile SoC architecture design, exploration, and evaluation





Scalable, High-
Performance
FPGA-Accelerated
Hardware
Simulation

ISCA '18

Micro Top Picks '18

ISCA@50 25-Year Retrospective

Widely-used Open-Source Project:

Used in 85+ pubs from 35+ institutions

Used in commercial chip development

Standard host platform in DARPA/IARPA programs

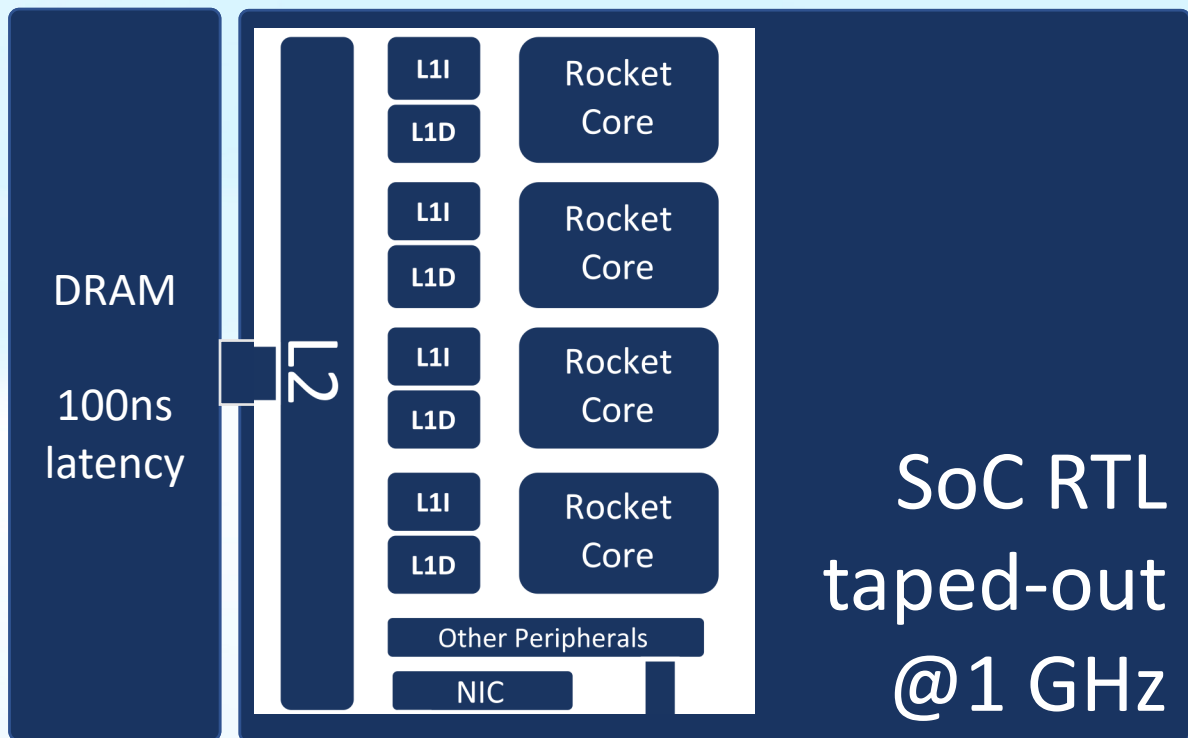
<https://fires.im>



The problem with FPGA prototypes

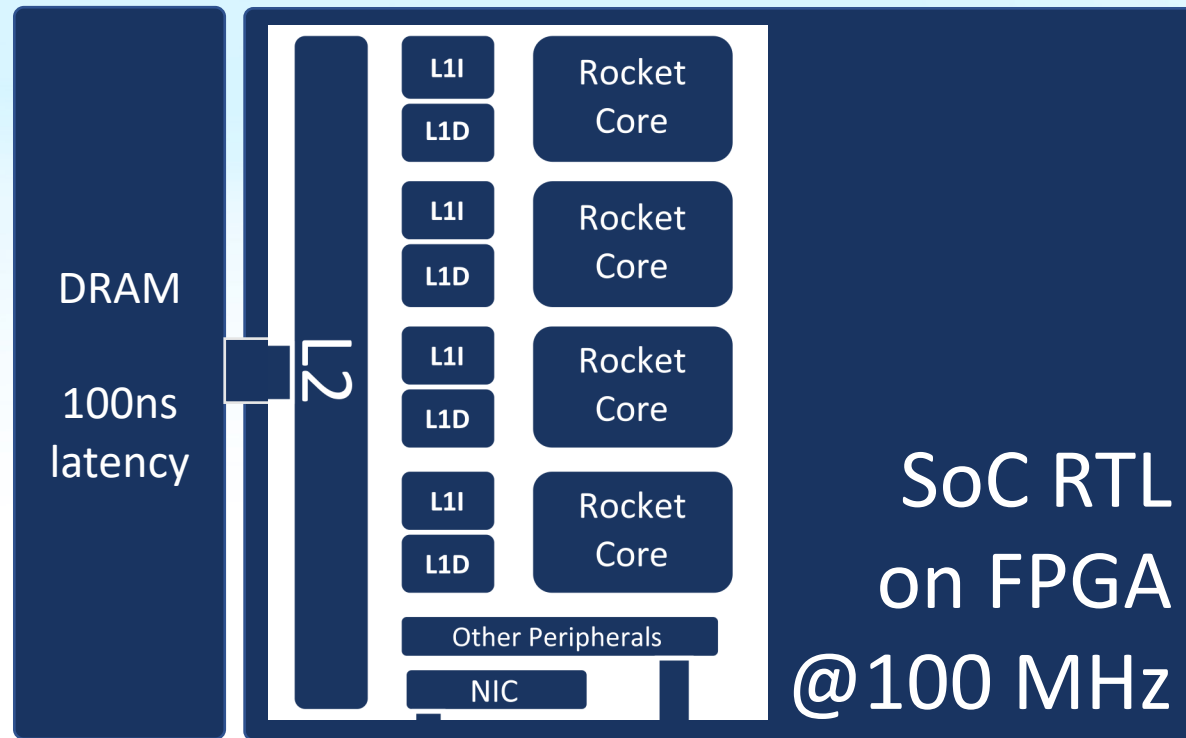
Incorrect I/O performance modeling

Taped-out System-on-Chip (SoC)



SoC sees 100 cycle
DRAM latency

FPGA Prototype of SoC



SoC sees 10 cycle DRAM latency
Incorrect by a factor of 10!

Step 1: 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.

- < 1/4 of an FPGA

Sim Rate

- N/A

Step 1: 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.

- < 1/4 of an FPGA

Sim Rate

- N/A



Step 2: FPGA-simulation of one SoC

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.

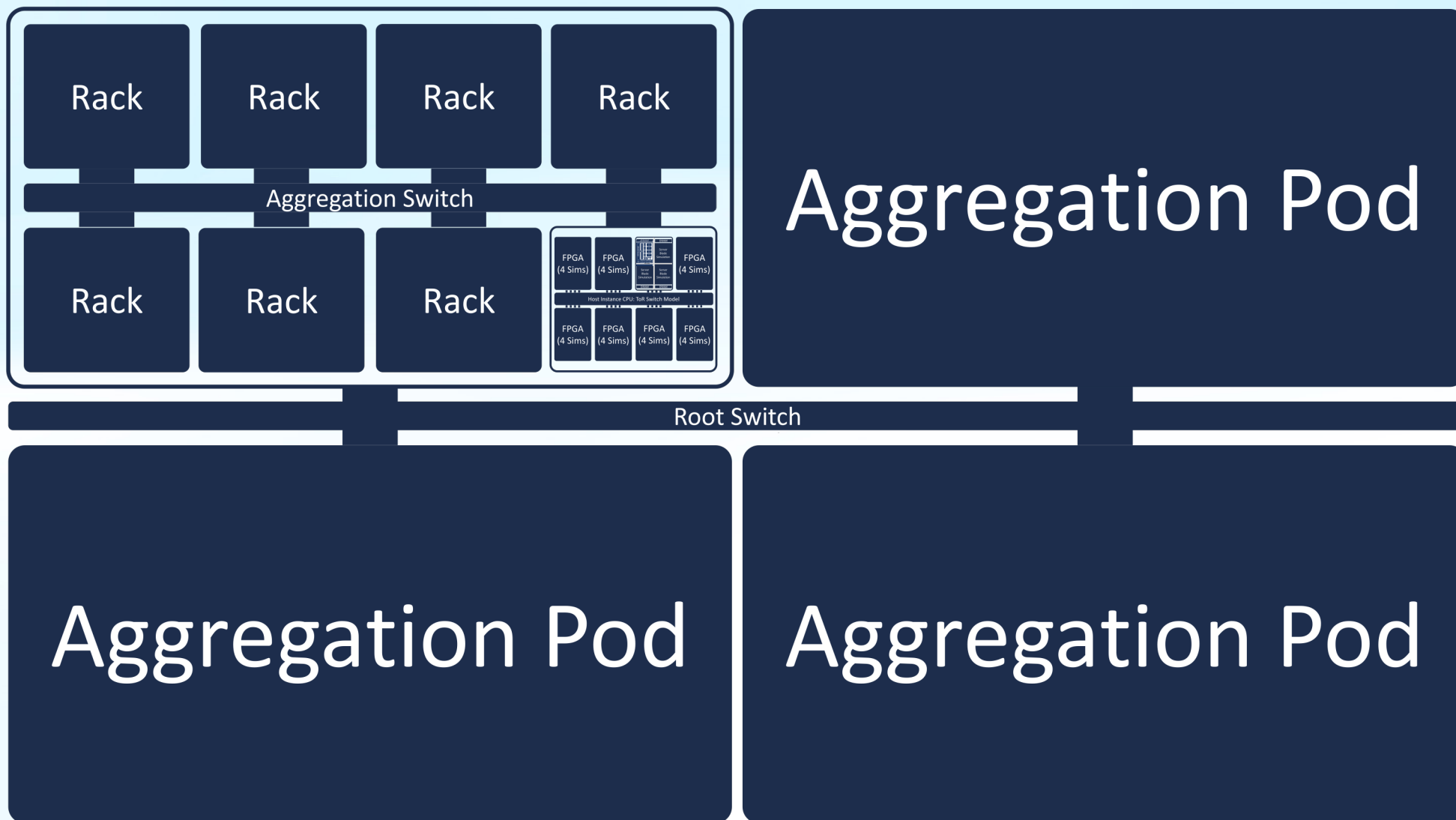
- $< \frac{1}{4}$ of an FPGA
- $\frac{1}{4}$ Mem Chans

Sim Rate

- ~150 MHz
- ~40 MHz (netw)



Step 6: Simulating a 1024-node datacenter



Modeled System

- 1024 Server SoCs
- 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 MHz²⁵(netw)

Step 6: Simulating a 1024-node datacenter



Not only enable fast simulation for individual hardware designs, but
unlock the ability to drive hardware innovation at scale.

Aggregation Pod

Aggregation Pod

Modeled System

- 1024 Server SoCs
- 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 MHz²⁹(netw)



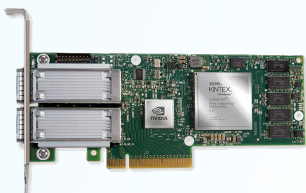
Productive Open-Source FPGA Simulation

Simulating designs from single SoCs to datacenters

- 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, e.g., previous datacenter example (performance depends on scale)
 - Non-networked, e.g., most users (perf. usually 50-150 MHz)
- Easy scaling from on-prem to cloud:



\$150



\$250



\$5K



\$10K



\$15K



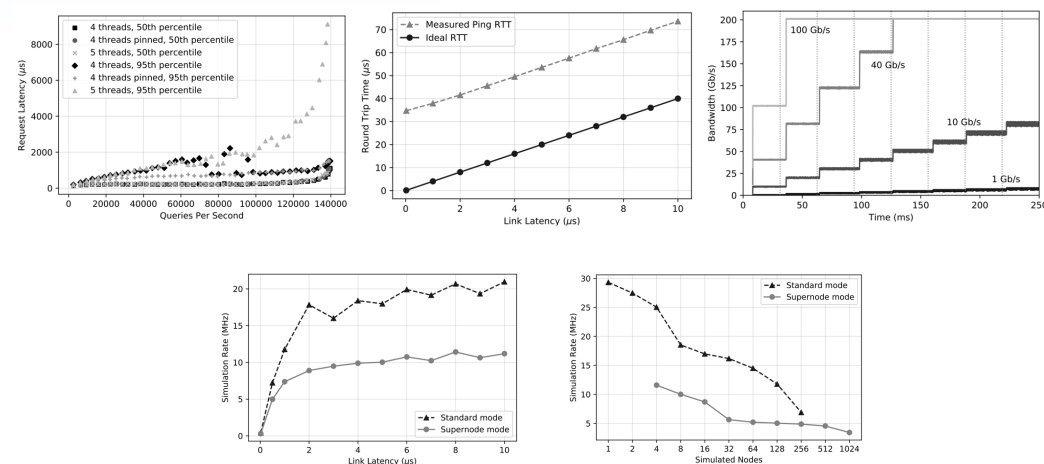
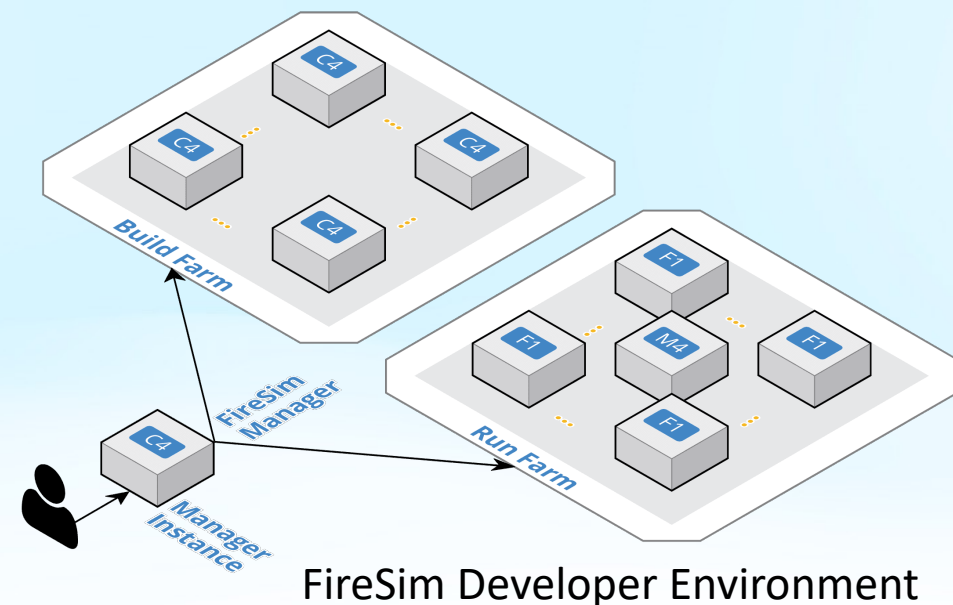
\$0.50-\$1.65/hr



Productive Open-Source FPGA Simulation

Automation and Reproducibility

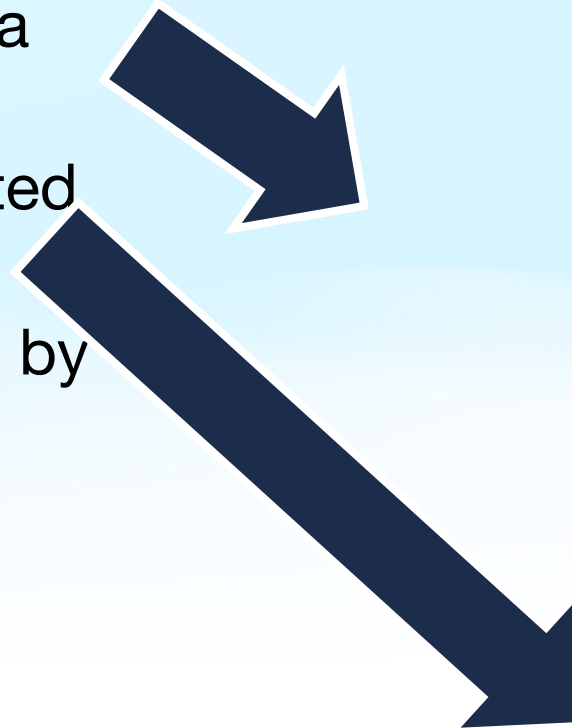
- `firesim` command line program
 - Like `docker` or `vagrant`, but for FPGA sims
 - User doesn't need to care about distributed magic happening behind the scenes
- Can programmatically call `firesim` to fully automate distributed FPGA sim
 - **Reproducibility**: included script to reproduce ISCA 2018 results
 - ```
$ cd fsim/deploy/workloads
```
    - ```
$ ./run-all.sh
```
 - Common workload flows built-in, e.g. automatically run SPECInt 2017 with full **reference inputs** in ≈ 1 day
 - Several user papers have gone through artifact evaluation using FireSim
- 200+ pages of documentation:
<https://docs.fires.im>



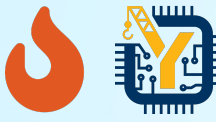
Summary of Key Insights behind FireSim



- User RTL is *automatically transformed* into a simulator
- Abstract models (when required) are validated + re-used
- FPGA simulation “tricks” should be applied by a compiler
 - e.g. multi-ported RAM and multi-threading optimizations
- Instrumentation and debuggability are key
 - Automated synthesis of printf, assertions
 - Automated insertion of ILAs
 - FirePerf (ASPLOS ‘20) added unlimited fidelity tracing, upleveled to software-profiling
- Automation extends beyond a single FPGA



FireSim/Chipyard Community Building



Nine hands-on tutorials at conferences like ISCA, ASPLOS, HPCA, MICRO.
e.g., <https://fires.im/asplos-2023-tutorial/>

1st FireSim/Chipyard Workshop @ ASPLOS 2023:
10 external talks from users/devs from industry + academia
Videos/Slides/Program @ <https://fires.im/workshop-2023/>

1st FireSim/Chipyard Tutorial (MICRO 2019)



First FireSim and Chipyard User and Developer Workshop at ASPLOS 2023

March 26, 2023 - Vancouver, BC, Canada

ASPLOS 2023

Table of Contents

- | | | |
|-----------------------------|-------------------------------------|-------------------------------|
| 1. Overview | 3. Program/Schedule | 5. Workshop |
| 2. Keynote | 4. Registration | 6. Is there a |

Overview

The FireSim and Chipyard user and developer community has extensive developer collaborations. This full-day workshop at ASPLOS 2023 will discuss the future direction of this ecosystem and spawn new collaborations.

This workshop will feature talks from academic and industrial users, covering hardware architecture, systems, programming languages, and VLSI research. It will inspire lively discussion of FireSim/Chipyard governance, features, and more.

Keynote

FireSim in High-Profile Action—FETT: DARPA's First Ever Bug Bounty Program
[Joe Kiniry](#), Principal Scientist, Galois



Bio: Dr. Kiniry is a Principal Scientist at Galois, where he leads the Secure Hardware/Firmware Assurance Model-Based Systems and Verifiable Elections, High-assurance Cloud and Chief Scientist of Free & Fair technologies and services.

Program/Schedule

Time (PDT)	Talk Title and Authors
9:00am - 9:30am	Intro and Welcome Workshop Organizers
9:30am - 10:20am	Keynote: "FireSim in High-Profile Action—FETT: DARPA's First Ever Bug Bounty Program" Joe Kiniry (Galois, Inc.)
10:20am - 10:40am	Coffee Break
10:40am - 11:05am	"TraceDoctor: Versatile High-Performance Tracing for FireSim" Björn Gottschall (Norwegian University of Science and Technology), Magnus Jahre (Norwegian University of Science and Technology)
11:05am - 11:30am	"Integrating a high performance instruction set simulator with FireSim to cosimulate operating system boots" Jiahao Zhang (Tenstorrent Inc.), Varun Koyyalagunta (Tenstorrent Inc.), Joe Rahmeh (Tenstorrent Inc.), Divyanshu Agrawal (Tenstorrent Inc.)
11:30am - 12:00pm	"Developing and Evaluating the nanoPU and nanoSort using Chipyard and FireSim" Stephen Ibanez (Stanford University & Intel), Theo Jepsen (Stanford University & Intel)
12:00pm - 1:40pm	Lunch
1:40pm - 2:05pm	"MIRAGE: Mitigating Cache Attacks with a Randomized Fully-Associative Cache" Gururaj Sureshwar (NVIDIA Research & University of Toronto), Moinuddin Qureshi (Georgia Tech)
2:05pm - 2:30pm	"ChipShop: A Cloud-Based GUI for Accelerating SoC Design" Shahzaib Kashif (Usman Institute of Technology), Talha Ahmed (Usman Institute of Technology), Farhan Ahmed Karim (Universiti Kebangsaan Malaysia)
2:30pm - 2:55pm	"Profiling an Architectural Simulator" Johnson Umeike (University of Kansas), Alex Manley (University of Kansas), Neel Patel (University of Kansas), Mohammad Alian (University of Kansas)
2:55pm - 3:20pm	"Berkeley eXtensible Environment: A Cloud-Based Open-Source Computer Architecture Simulation Environment" Farzad Fatollahi-Fard (Lawrence Berkeley National Laboratory), Nirmalendu Patra (Lawrence Berkeley National Laboratory), Angelos Ioannou (Lawrence Berkeley National Laboratory), John Shalf (Lawrence Berkeley National Laboratory)
3:20pm - 3:40pm	Coffee Break
3:40pm - 4:05pm	"FireSim on Xilinx U250 and Other Custom Host Platforms" David Christoph Metz (Norwegian University of Science and Technology), Magnus Sjölander (Norwegian University of Science and Technology)

FireSim Impact

A unifying research platform enabling cross-domain collaboration.

User papers have appeared at: ASPLOS, CCS, DAC, DATE, ESSCIRC, EuroSys, HOST, HPCA, IEEE SP, IISWC, ISCA, ISCAS, ISPASS, MICRO, NDSS, OSDI, PASC, RTAS, SOSR, USENIX Security, VLSI, etc.



Awards:

Micro Top Picks '18
CACM Research Highlights Nominee
ISCA@50 25-Year Retrospective

RETROSPECTIVE: FireSim: FPGA-Accelerated
Cycle-Exact Scale-Out System Simulation in the
Public Cloud

Sagar Karandikar¹, Howard Mao^{2*}, Donggyu Kim^{3*}, David Biancolin^{4*}, Alon Amid^{5*}, Dayeol Lee^{6*},

Used (not only cited) in **85+ peer-reviewed publications** from first-authors at **35+ institutions** in academia and industry

<https://fires.im/userpubs/>

Used in the development of commercial silicon and validated against commercial silicon

Used as a standard host platform in DARPA and IARPA programs



Chipyard Impact

Used in over 20 tape-outs at multiple academic institutions

Used (not only cited) in **150+ peer-reviewed publications** from first-authors at **65+ institutions** in academia and industry

Used extensively in education, e.g. in UCB Tapeout Class

2000+ GitHub Stars
800 Public Forks
650 mailing list members

These are *not* the right HW design tools for the AI-era...

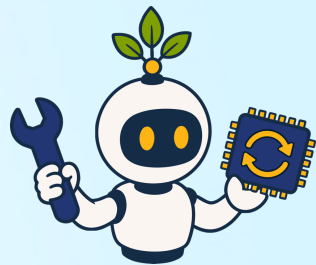
Example of a *human-centric* flow, that makes assumptions such as:

- Generating new ideas is slow
- Writing the RTL implementation is slow
- The human needs to sleep. Not always the end of the world if a flow step takes 8 hours
- Most of the time, the human doesn't want to edit design representation at arbitrary levels of abstraction
- The human can only interpret so much data from the system at once...

But they are a pretty good start!

Challenges Revisited

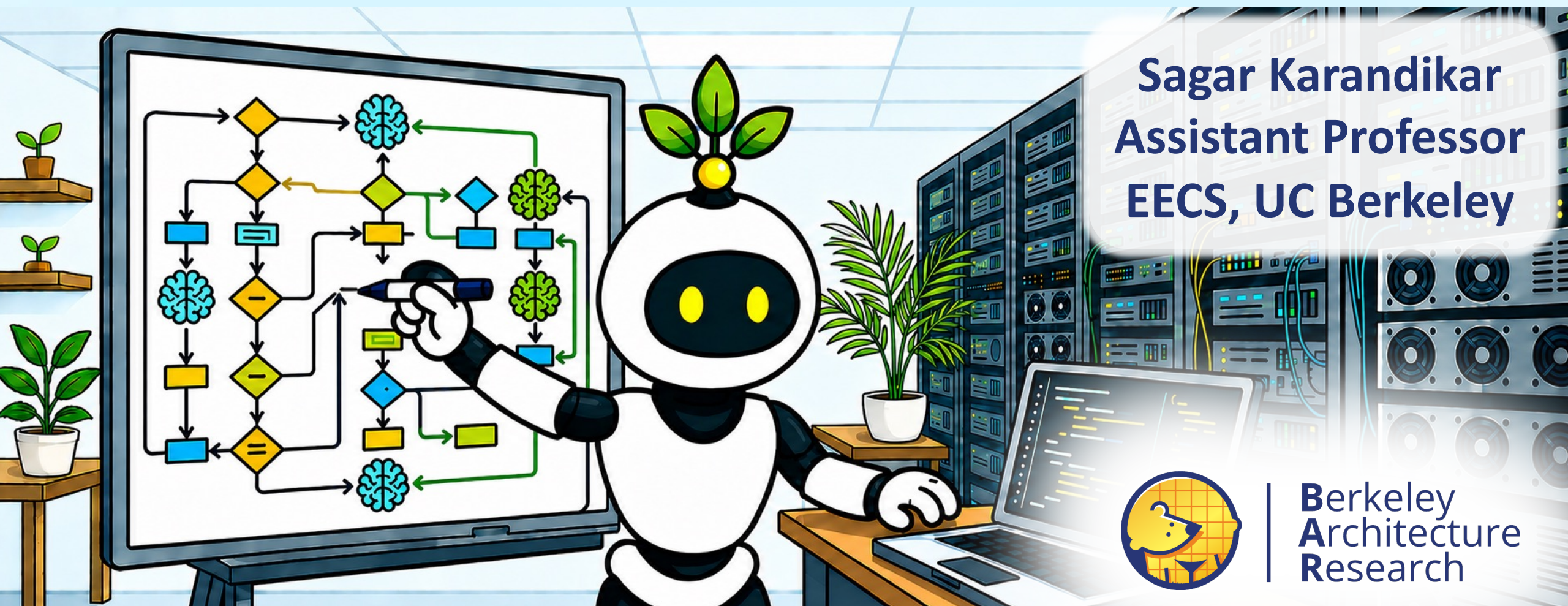
- The whole point of applying AI: We want the computers to be doing the work for us
 - High-level uarch sim won't cut it as the evaluation endpoint
- Many proposed evolutionary flows/algorithms
 - Algorithmic differences are interesting
 - But the flows are static; most expect one contained SW binary to eval
- Immense engineering/"good science" challenges for long-running evals (i.e., all of HW/SW co-design)
 - Other areas have IMMENSE work on validating the sims they are using in these areas



CHIA

<https://chialoops.ai>

An open-source framework for
principled, agentic AI-driven
HW/SW co-design research



Sagar Karandikar
Assistant Professor
EECS, UC Berkeley

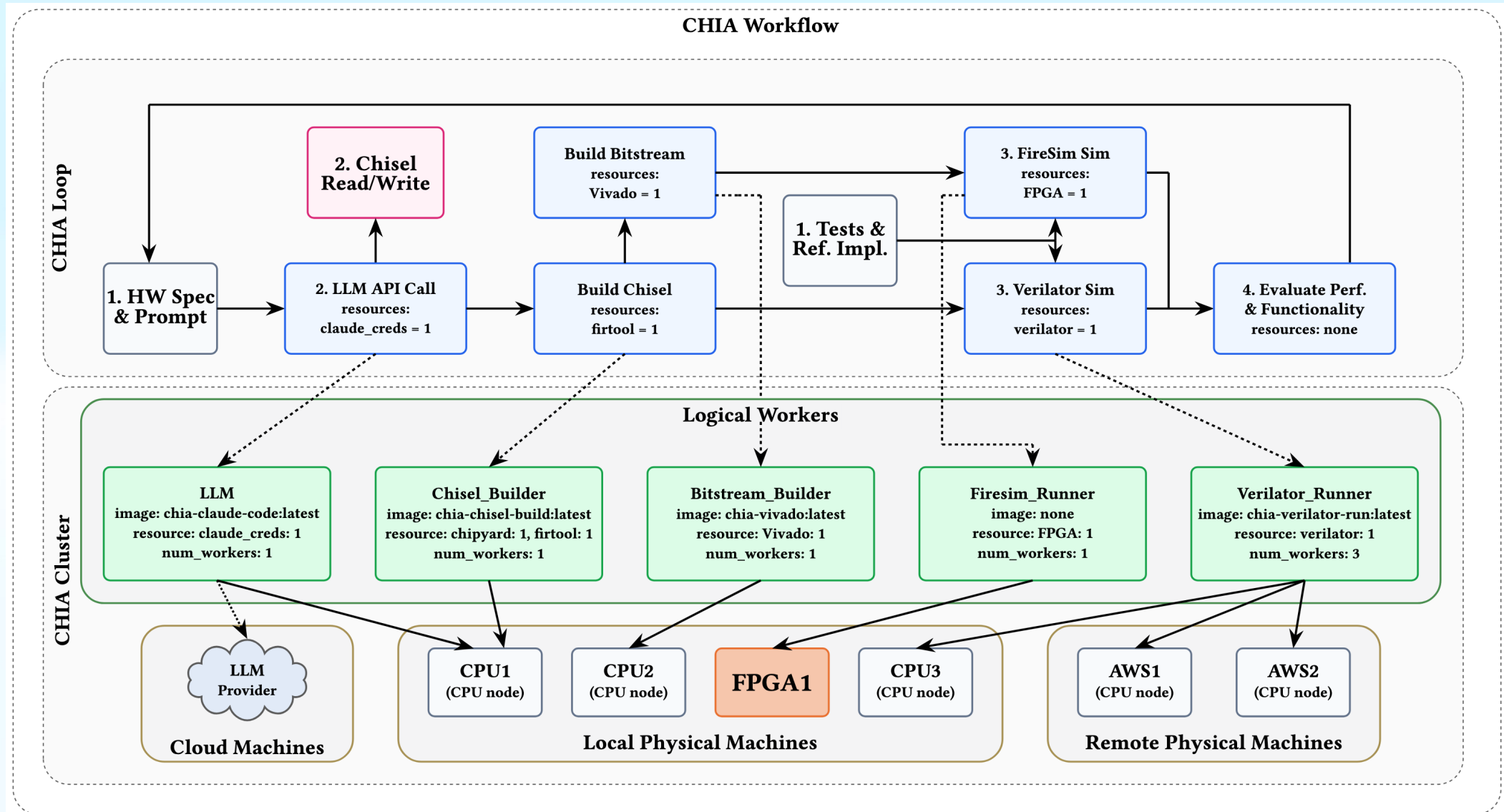


Berkeley
Architecture
Research

An open-source framework for principled, agentic AI-driven HW/SW co-design research

- Graph-based CHIA Loop abstraction to make it easy to design massive, parameterized AI-infused design pipelines
- Maximize modularity and re-use: CHIA provides plug-and-play support for lots of tools
- Express both *programmatic* and *agentic* decision making
- Automatically map the CHIA Loop onto dynamic and heterogeneous substrate, flexibly and with fault-tolerance
- Enable “good science”: built in data collection and flow profiling, plus isolation of LLMs from, e.g., tests and golden models by construction

A user should worry about only one thing...



CHIA works with the tools you already use!

Don't see your tool here? Adding a new CHIA node is easy!



ChampSim

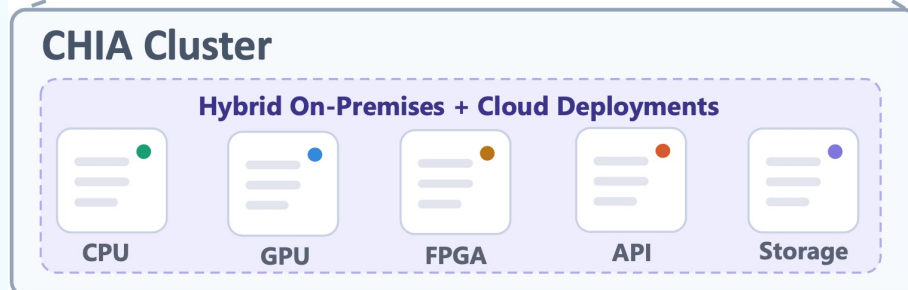
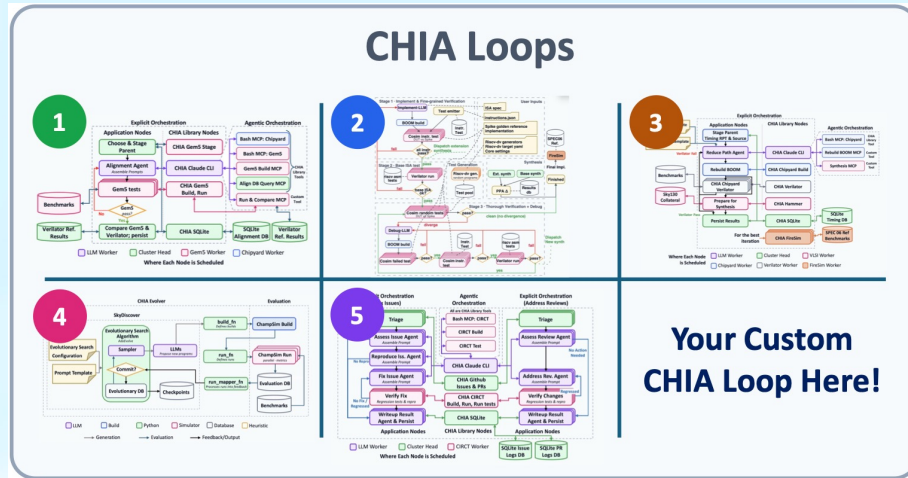


AlphaEvolve

OpenEvolve



Case studies



Case Studies (Section 5)

Agentic generation of a gem5 model from BOOM RTL

Improve gem5 core model's accuracy from 40% → 97.2% relative to ground-truth RTL simulation of BOOM over 10.5 days on 36 benchmarks

Agentic RISC-V ISA Extension RTL Implementation

5.6% (Bitmanip) and 3.5% (Zicond) speedup on SPEC06 ref., 25+ T insts.; 10× speedup on OpenSSL (Crypto); < 5% area impact & no freq. change

Agentic IPC-aware Critical-Path Optimization in RTL

2.03× frequency increase with only 3.28% IPC loss on SPEC06 ref. by rewriting RTL; net 1.96× Iron Law speedup; negligible area change

Agentic Architecture Discovery in μArch Simulators and RTL

AI agent and evolutionary algorithm-directed exploration and implementation of novel microarchitectural ideas in ChampSim & BOOM

Maintainer-friendly Agentic Bug-fixing for CIRCT GitHub Issues

Autonomously triaged CIRCT GitHub issues into actionable, high-quality, and now upstreamed PRs in ~45 minutes

Your Custom CHIA Loop Here!

Build your own custom AI-driven HW/SW co-design loops with CHIA!

Case studies



Agentic generation of a gem5 model from BOOM RTL

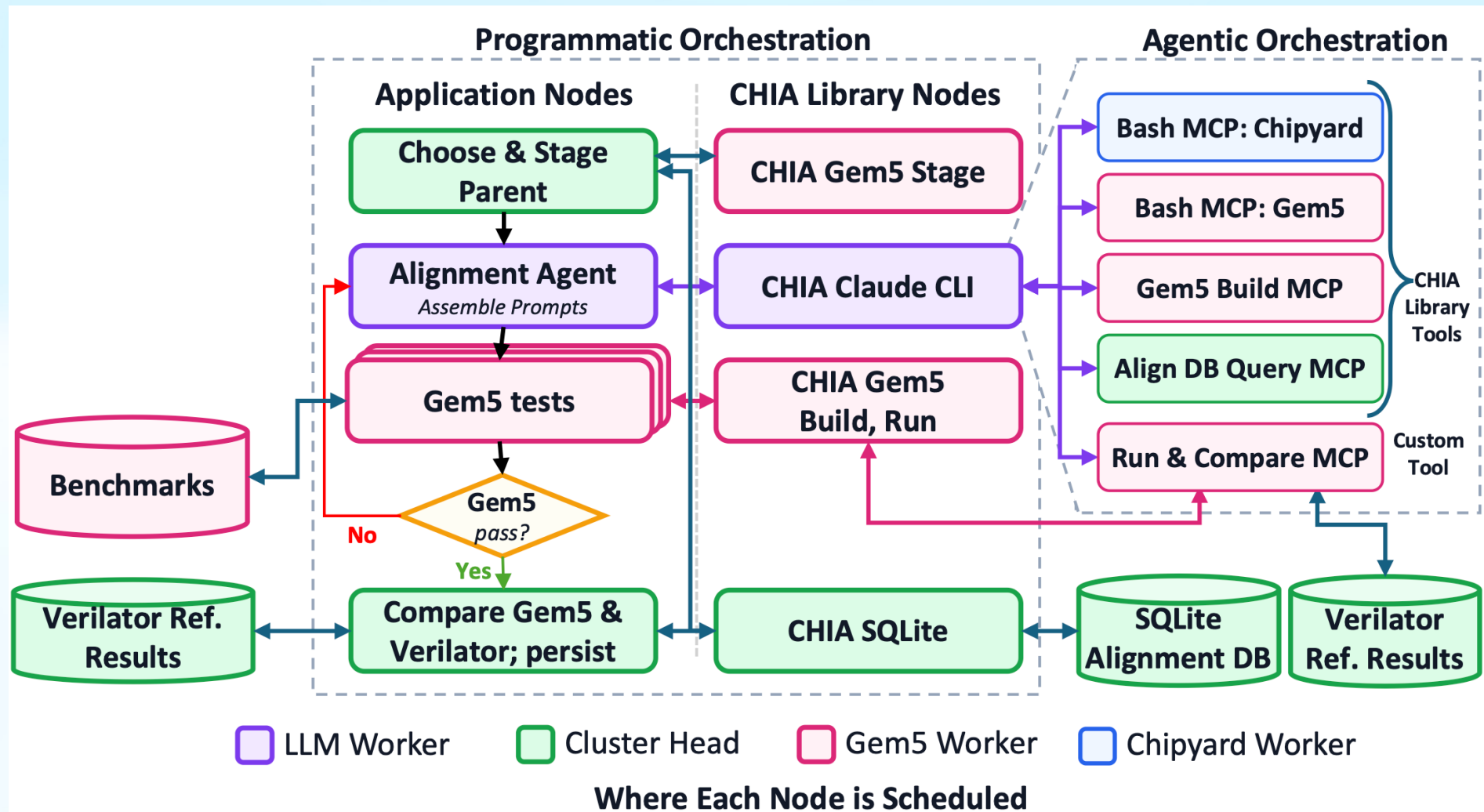
Improve gem5 core model's accuracy from 40% → 97.2% relative to ground-truth RTL simulation of BOOM over 10.5 days on 36 benchmarks

1



Agentic generation of a gem5 model from BOOM RTL

Improve gem5 core model's accuracy from 40% → 97.2% relative to ground-truth RTL simulation of BOOM over 10.5 days on 36 benchmarks

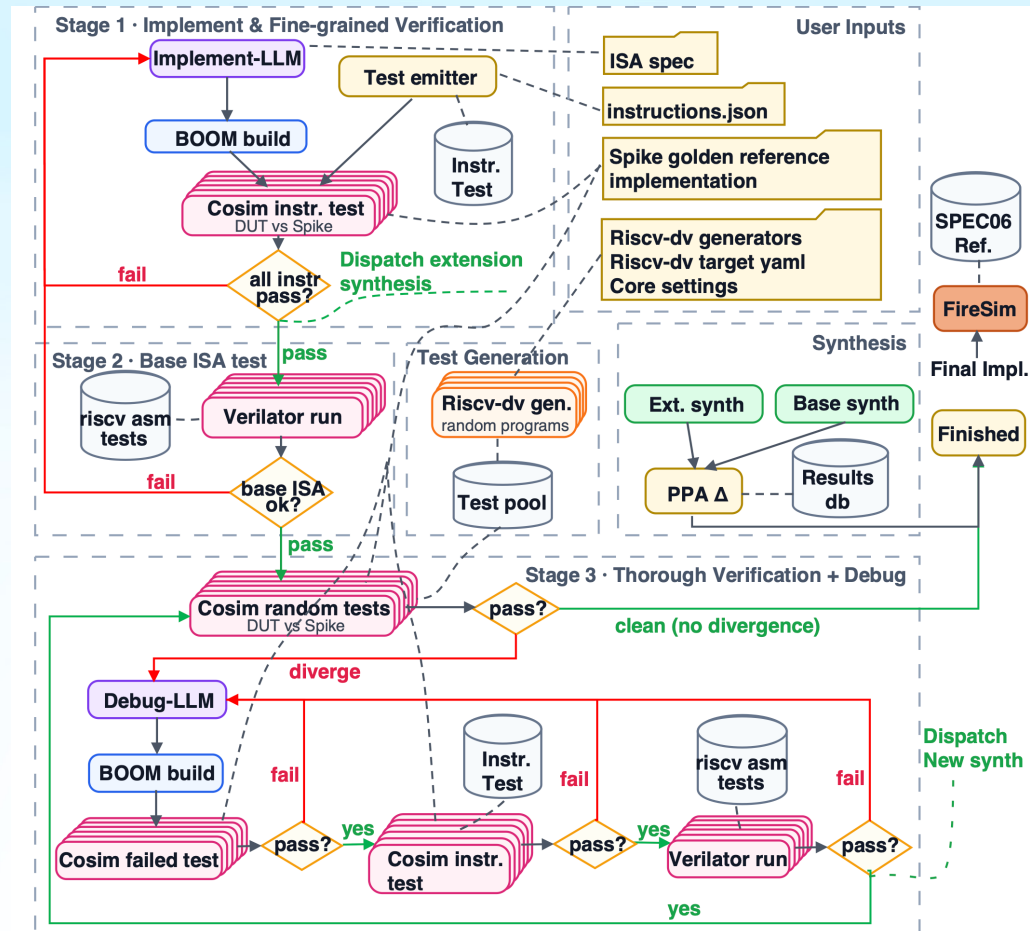


2



Agentic RISC-V ISA Extension RTL Implementation

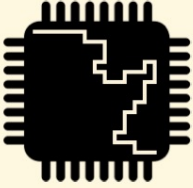
5.6% (Bitmanip) and 3.5% (Zicond) speedup on SPEC06 ref., 25+ T insts.;
10× speedup on OpenSSL (Crypto); < 5% area impact & no freq. change



Chia Library Nodes

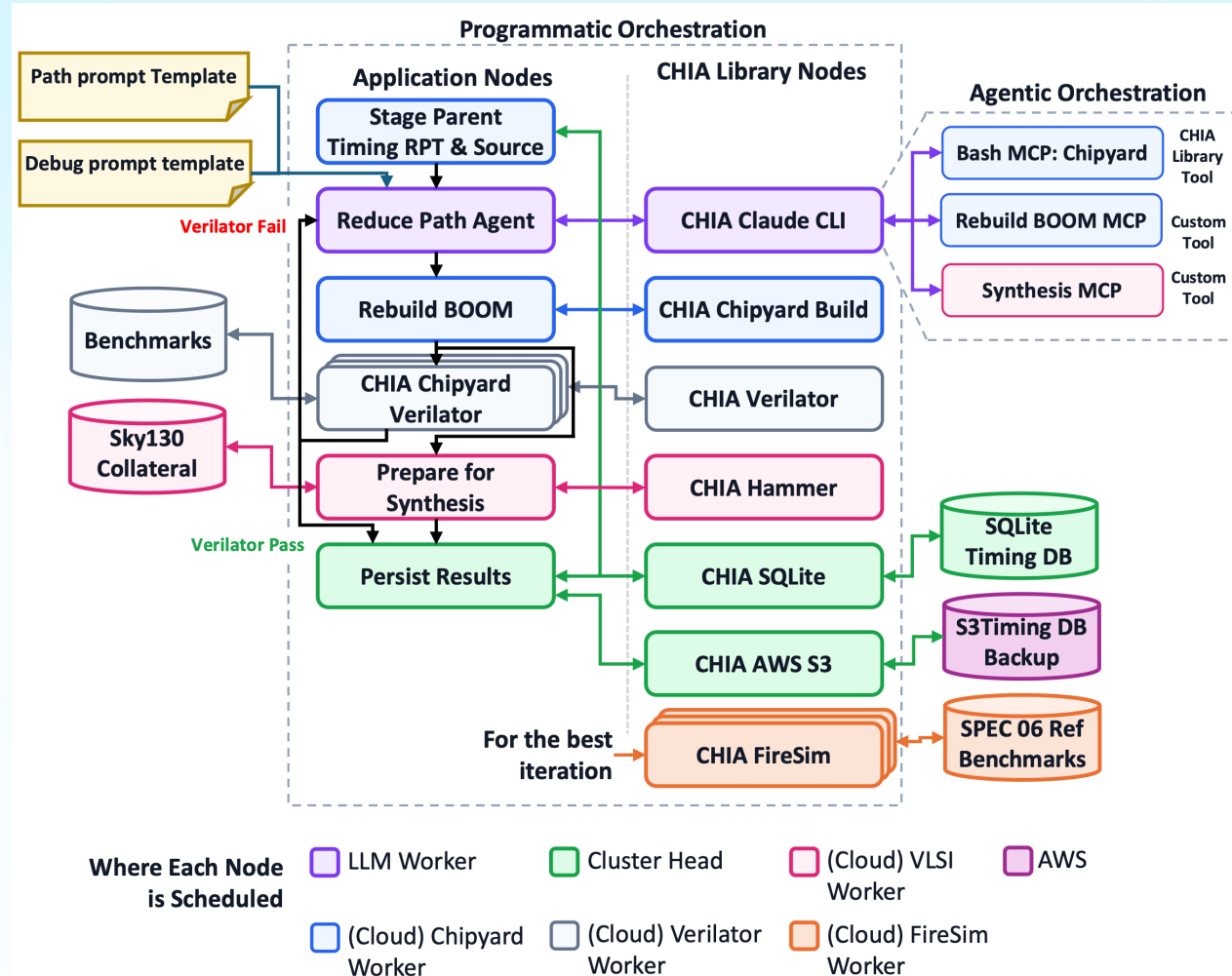
- Riscv-dv Node
- Chipyard Node
- LLM Node
- Persistent Logging Node
- Verilator Node
- VLSI Node
- FireSim Node
- Head Node

3

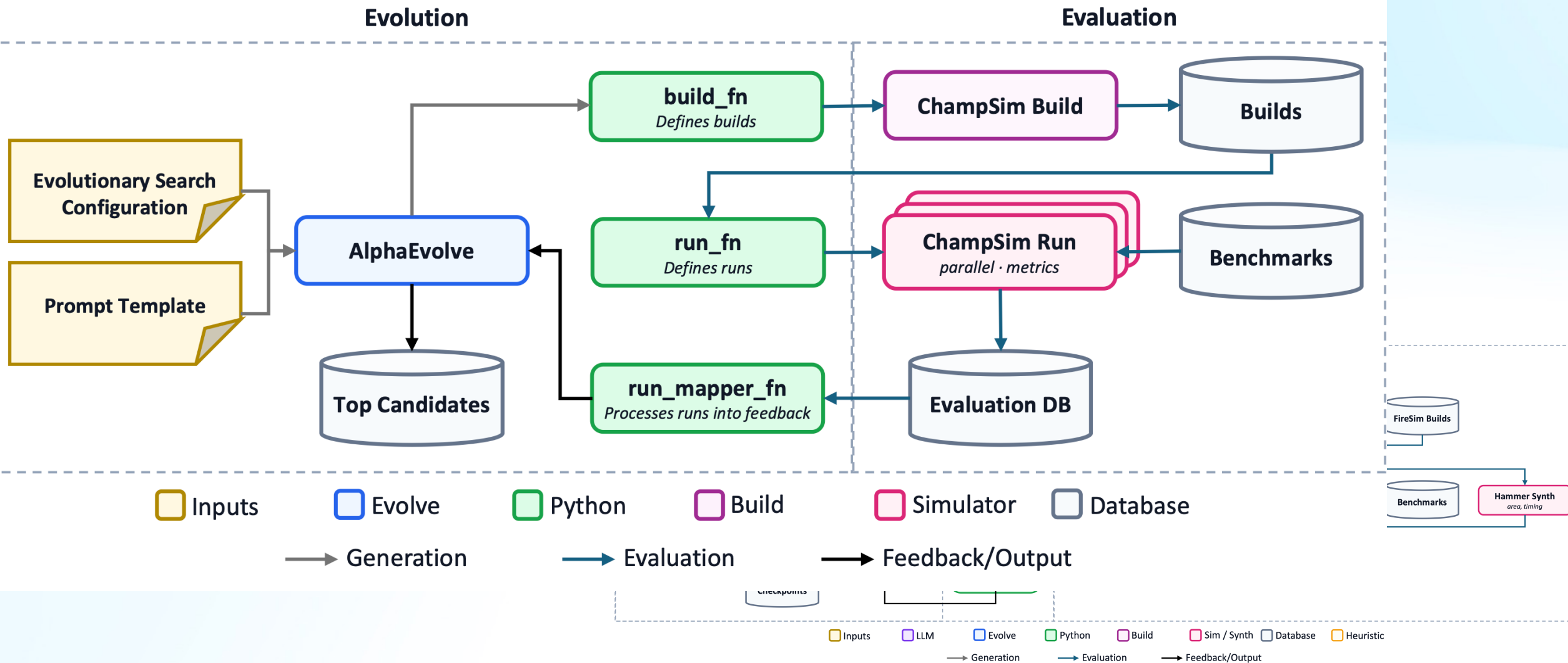


Agentic IPC-aware Critical-Path Optimization in RTL

2.03× frequency increase with only 3.28% IPC loss on SPEC06 ref. by rewriting RTL; net 1.96× Iron Law speedup; negligible area change



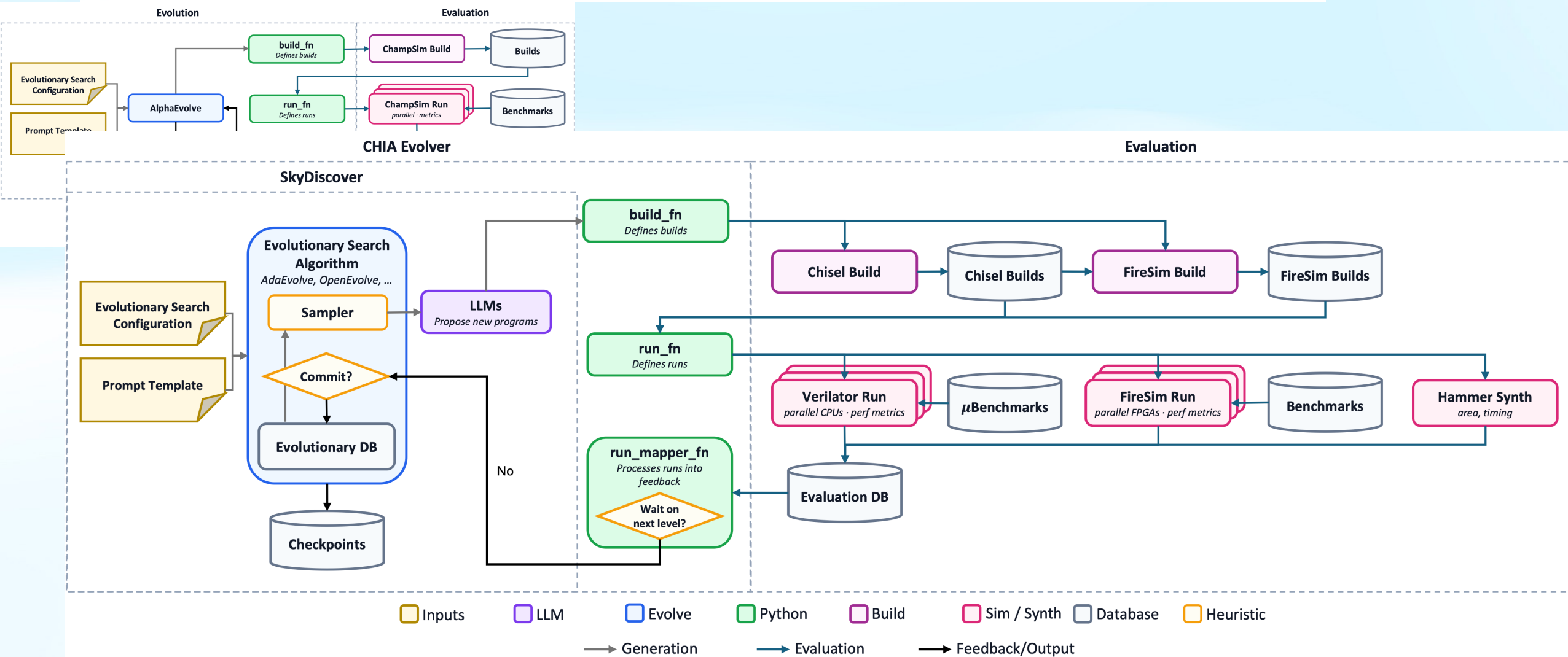
4



4

SkyDiscover AlphaEvolve

Agentic Architecture Discovery in μ Arch Simulators and RTL
AI agent and evolutionary algorithm-directed exploration and implementation of novel microarchitectural ideas in ChampSim & BOOM

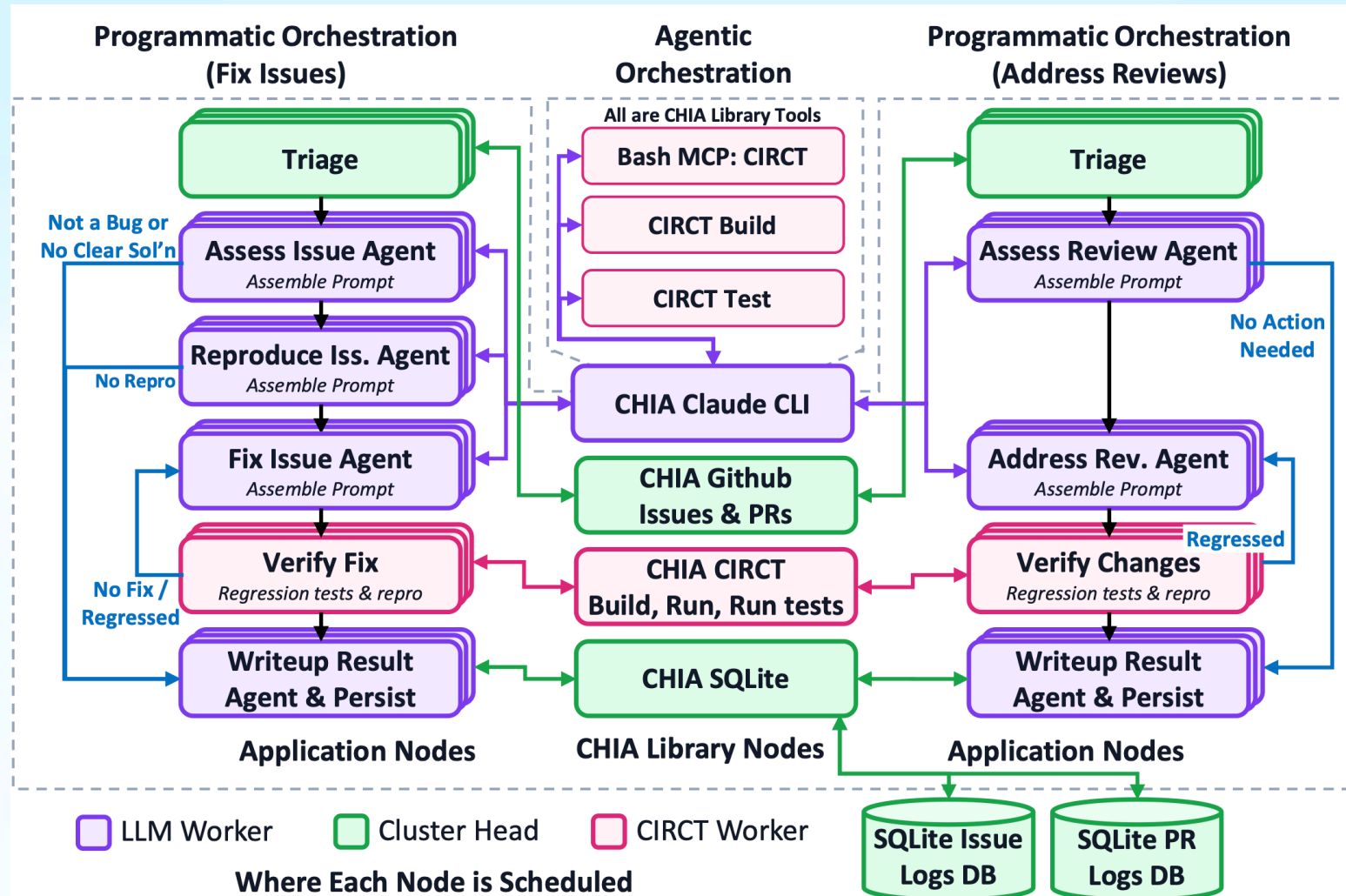


5



Maintainer-friendly Agentic Bug-fixing for CIRCT GitHub Issues

Autonomously triaged CIRCT GitHub issues into actionable, high-quality, and now upstreamed PRs in ~45 minutes





Your Custom CHIA Loop Here!

Build your own custom AI-driven HW/SW co-design loops with CHIA!

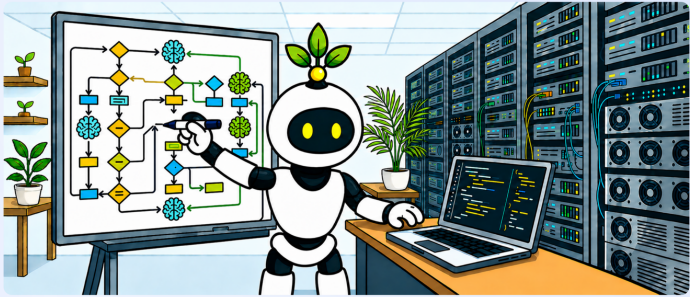
CHIA is hot off the presses, available to use!

<https://chialoops.ai>

<https://docs.chialoops.ai>

CHIA

AboutPre-print PaperBlogDocsUser ForumAuto



An open framework for designing and deploying custom AI-driven HW/SW co-design flows *fast*

CHIA puts AI agents to work across the hardware/software co-design stack, with use cases ranging from **agentic architectural discovery** to solving **practical chip engineering problems**. The *CHIA Loops* abstraction makes it easy to **compose, deploy, and share** complex, AI-in-the-loop flows with the tools you already use.

CHIA works with many common HW/SW co-design tools, including RTL system-on-chip development tools (e.g., [Chipyard](#)), microarchitectural simulators (e.g., [gem5](#), [Champsim](#)), RTL simulators (e.g., [Verilator](#), [FireSim](#)), compilers (e.g., [CIRCT](#)), FPGA and ASIC CAD tools (e.g., via [Hammer](#)), and more.

[Read the Pre-print Paper](#)

[Get started](#)

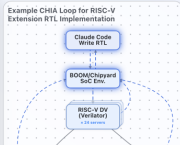
Jump to:

User experienceToolflow compatibilityCase studiesLearn moreLatest updates

One command, a full AI-driven co-design loop

Deploy AI agents inside a complete RISC-V SoC framework. A single `chia job submit` can drive agentic implementation, debugging, simulation, and PPA analysis end-to-end.

```
~$ cd chipyard (open or for illustration purposes)
$ chia job submit -- bitmanip-flow.py
• LLM is implementing RISC-V Bitmanip extension in BOOM RTL.
```



CHIA

latest

Search docs

GETTING STARTED

CHIA Basics

Installation

Quickstart: Say Hello (World) with CHIA

CONCEPTS

Architecture Overview

USER GUIDES

ChiaFunction

ChiaTool

Cluster Configuration Reference

Building a CHIA-compatible Docker image

Caching and Bypass

Profiling

Example: Agentic RoCC Accelerator (MemCpy)

gem5 ↔ BOOM Microarchitecture Alignment

Autonomous CIRCT Issue Solving

BOOM Critical-Path Timing Optimization

CLI REFERENCE

CLI Reference

API REFERENCE

Base Runtime

Chipyard Nodes

Model Backends

FireSim

VLSI

Welcome to CHIA's documentation!

[View page source](#)

Welcome to CHIA's documentation!

CHIA is an open-source framework for agile and principled hardware/software co-design research — where humans, agents, and existing hardware-design tools all work together.

New to CHIA? Start with the [CHIA Basics](#) page, or go straight to [Installation](#) to set things up.

The current release of CHIA is considered a beta. If you find any issues, please reach out to us via the [user forum](#) or [contact the lead developers](#) / [PI](#) (Angela Cui, Ferran Hermida-Rivera, Jack Toubes, and Sagar Karandikar).

CHIA stands for:

(Co-designing Hardware and software with Intelligent Agents)

Originally CHIA stood for:

(CHipyard's Intelligent Agents)

(CHipyard's Intelligence Agency)

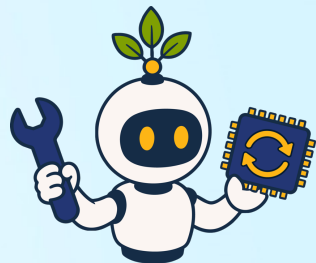
Getting Started

- CHIA Basics
- Installation
 - Optional extras
 - Core dependencies
 - Next steps
- Quickstart: Say Hello (World) with CHIA
 - What we will do in this tutorial
 - Requirements
 - 0. Project Setup
 - 1. `print()`ing Hello from a worker
 - 2. Print Hello in Verilator
 - 3. Asking an LLM to say Hello
 - 4. Editing RTL to say Hello

Concepts

[Architecture Overview](#)

latest



CHIA

<https://chialoops.ai>
(includes paper link)

Sagar Karandikar
Assistant Professor
EECS, UC Berkeley

Come say hi!
(P.S. We have stickers)

Tomorrow in MLArchSys, the lead students will give a talk at 3:15pm!



Angela Cui



Ferran Hermida-Rivera



Jack Toubes