

CHIA



Berkeley
Architecture
Research

<https://chialoops.ai>

CHIA: An open-source framework for principled, agentic AI-driven hardware/software co-design research

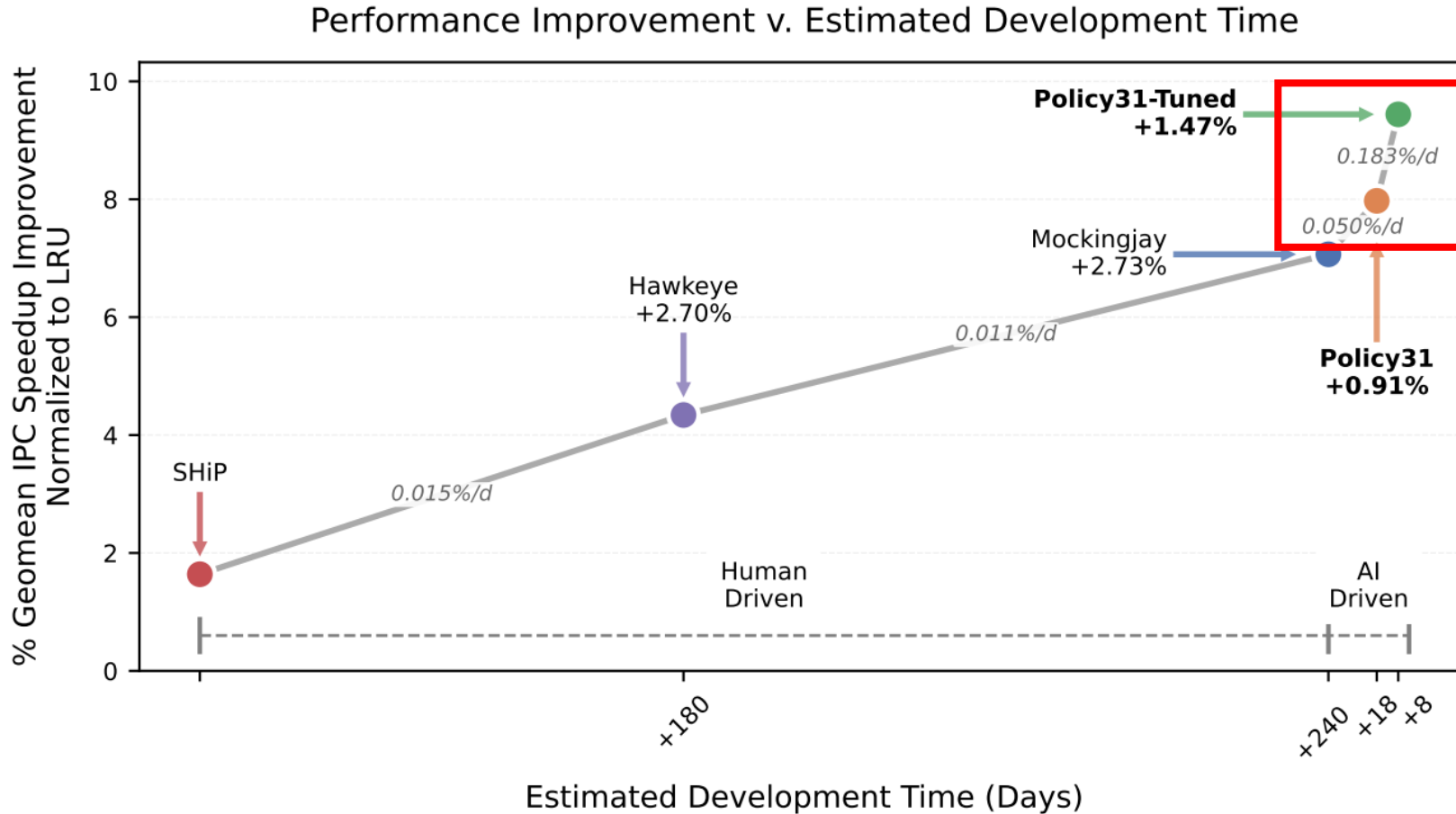
MLArchSys, ISCA 2026

Angela Cui*, **Ferran Hermida-Rivera***, **Jack Toubes***,
Raghav Gupta, Jim Fang, Chengyi Lux Zhang, Ella Schwarz, Junha Kim,
Yakun Sophia Shao, Borivoje Nikolić, Christopher W. Fletcher, Sagar Karandikar

*Equal Contribution



AI-driven design brings huge opportunities



Use AlphaEvolve to agentically discover new cache replacement policies

Up to 12x speedup in (Perf improvement) / (Dev time) compared to human

Raghav Gupta, Akanksha Jain, Abraham Gonzalez, Alexander Novikov, Po-Sen Huang, Matej Balog, Marvin Eisenberger, Sergey Shirobokov, Ngân Vu, Martin Dixon, Borivoje Nikolić, Parthasarathy Ranganathan, and Sagar Karandikar. 2025. **ArchAgent: Agentic AI-driven Computer Architecture Discovery**. <https://openreview.net/forum?id=hcXN9I6zqZ>



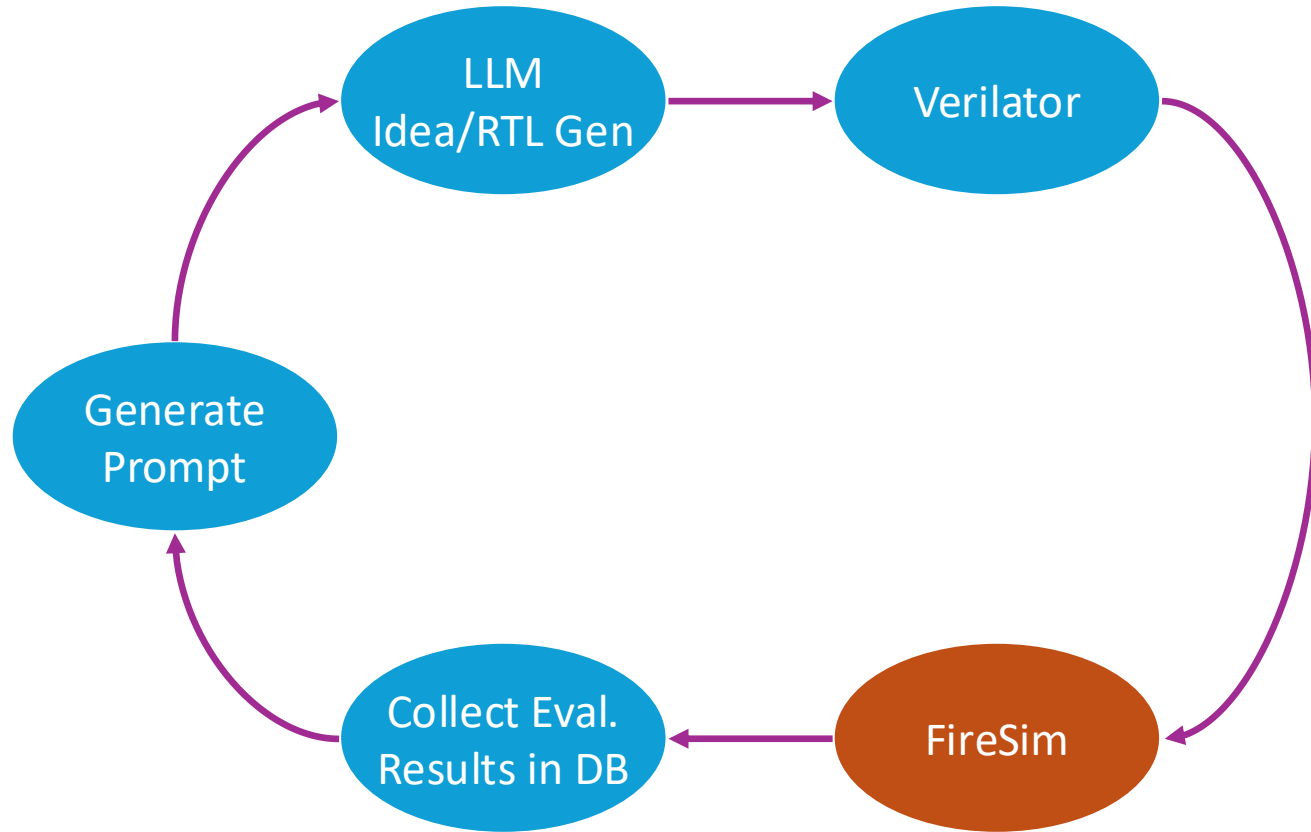
Challenges

- 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; designed for a specific use-case
- Immense engineering/"good science" challenges for long-running/highly parallel evals (i.e., all of HW/SW co-design)
 - Agentic flows work best with evaluators that are verified



A basic specification -> RTL agentic loop

- FPGA
- CPU

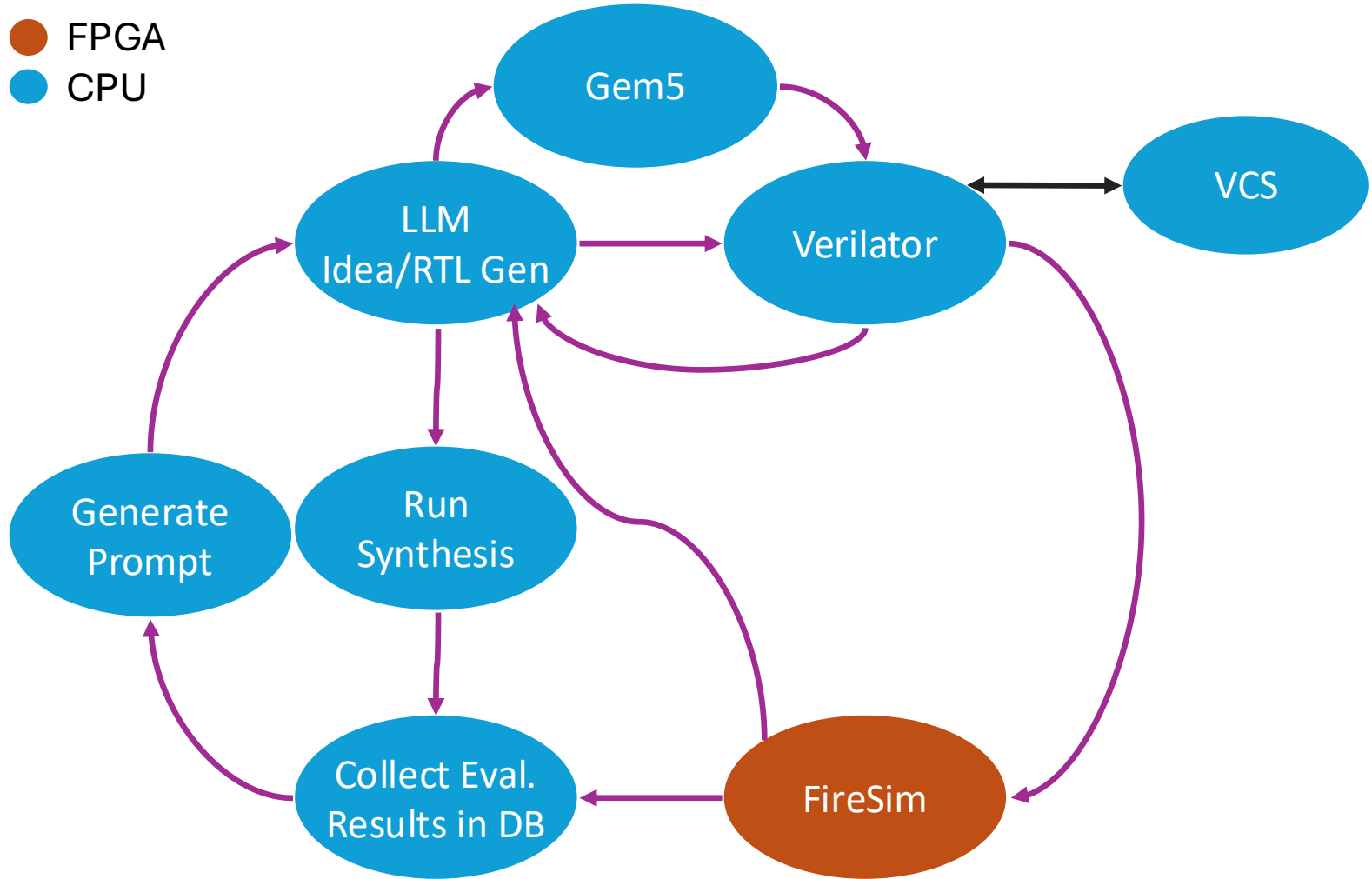


Maybe a script automates the steps of this loop...



What if we want to tune the flow?

- FPGA
- CPU



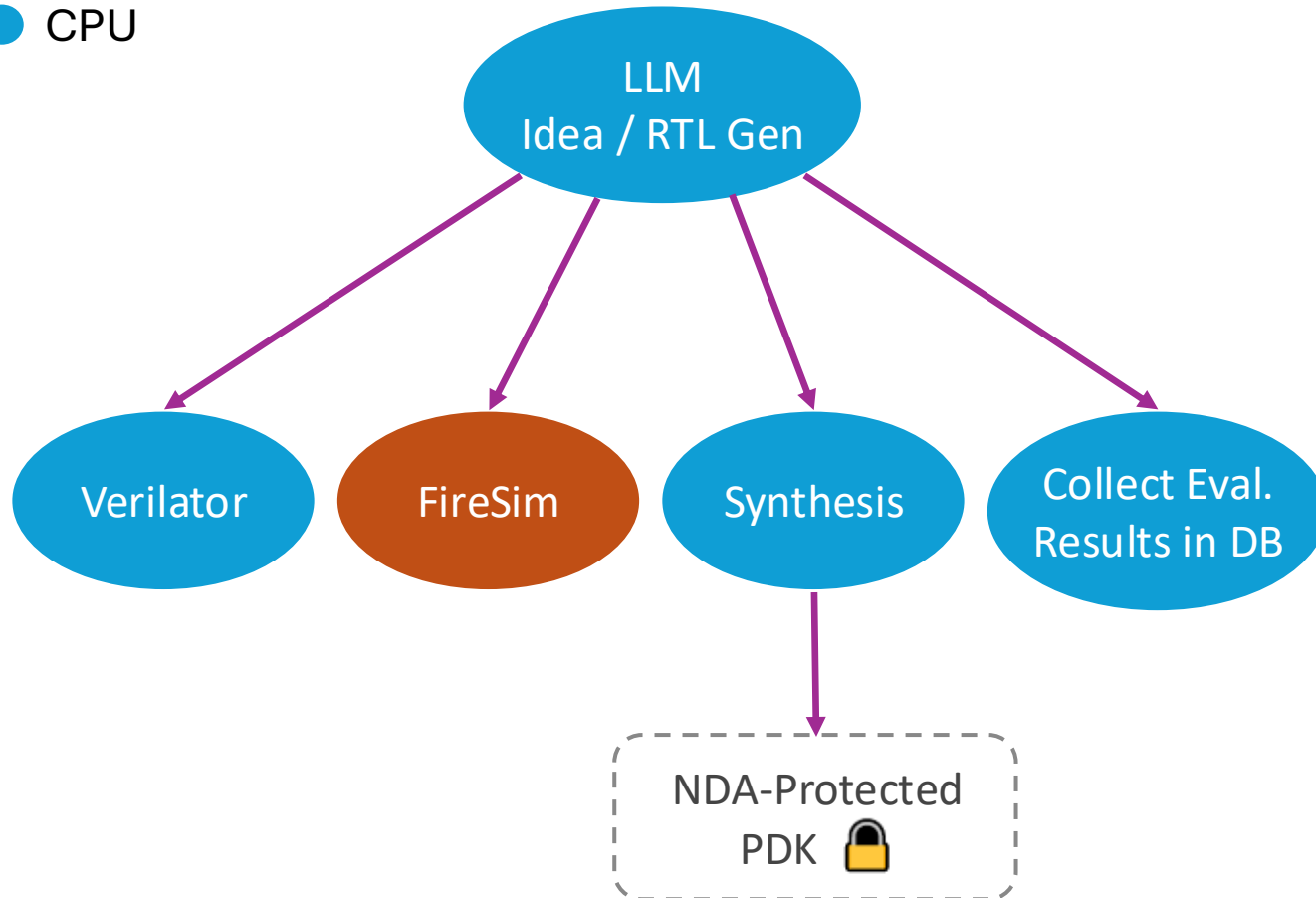
What the framework needs

- ▶ Enables agile experimentation



Don't let the agent run wild

- FPGA
- CPU

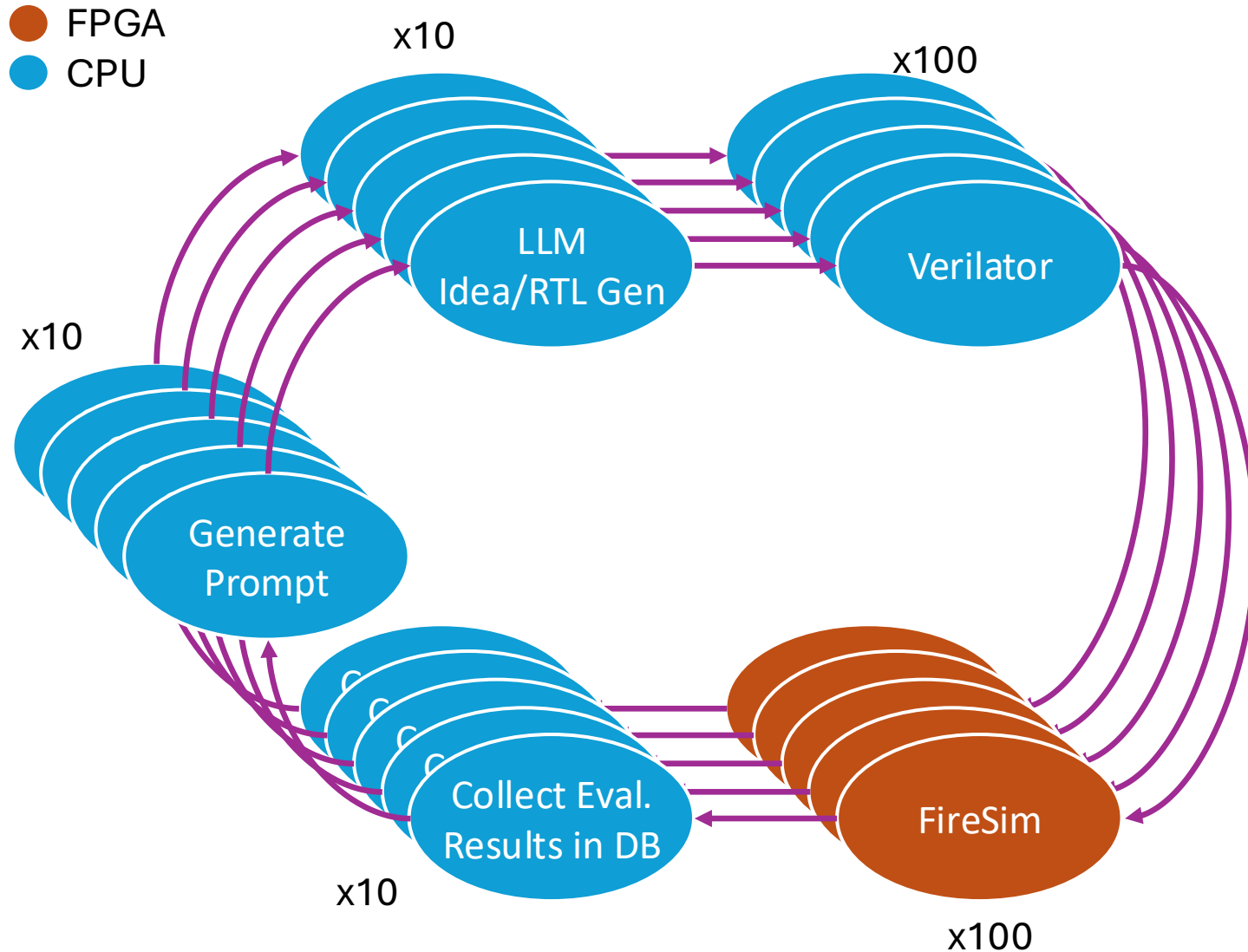


What the framework needs

- ✓ Enables agile experimentation
- ▶ **Control flow flexibility + expressiveness**



LLMs can do 10 things at once



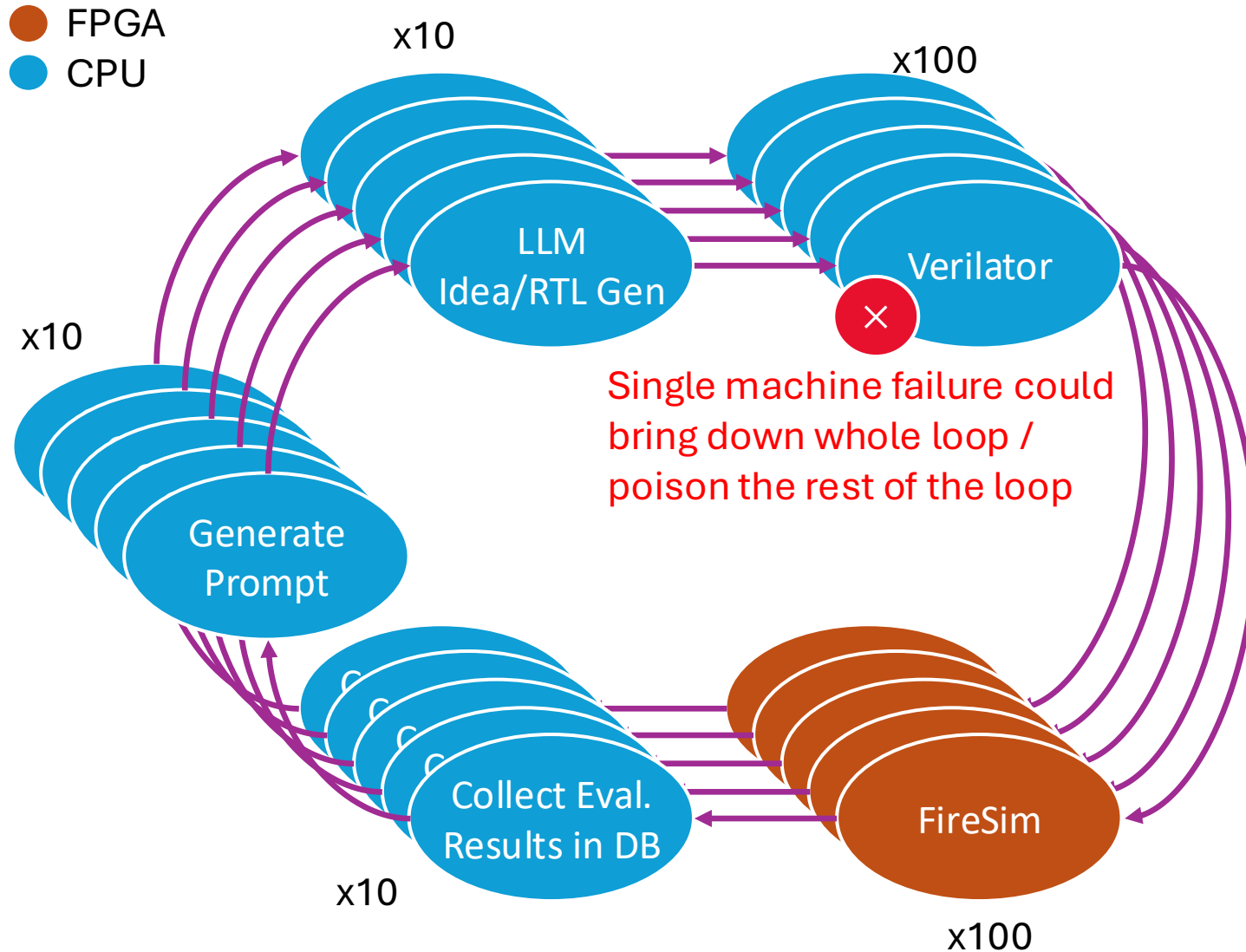
What the framework needs

- ✓ Enables agile experimentation
- ✓ Control flow flexibility + expressiveness

▶ **Parallelization -> Distribution**



Failures become a significant concern at scale

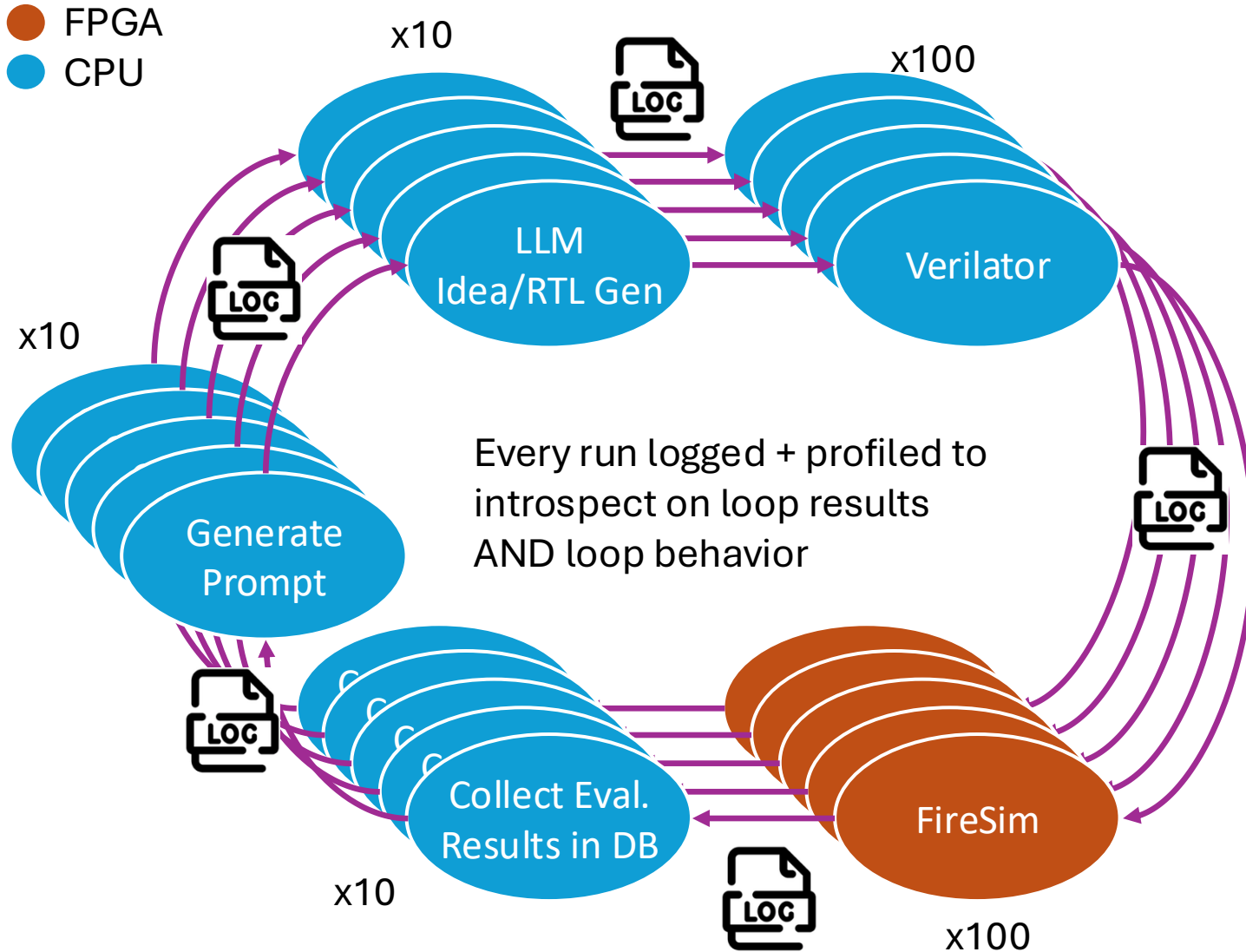


What the framework needs

- ✓ Enables agile experimentation
- ✓ Control flow flexibility + expressiveness
- ✓ Parallelization -> Distribution
- ▶ **Fault tolerance**



Do good science with data collection



What the framework needs

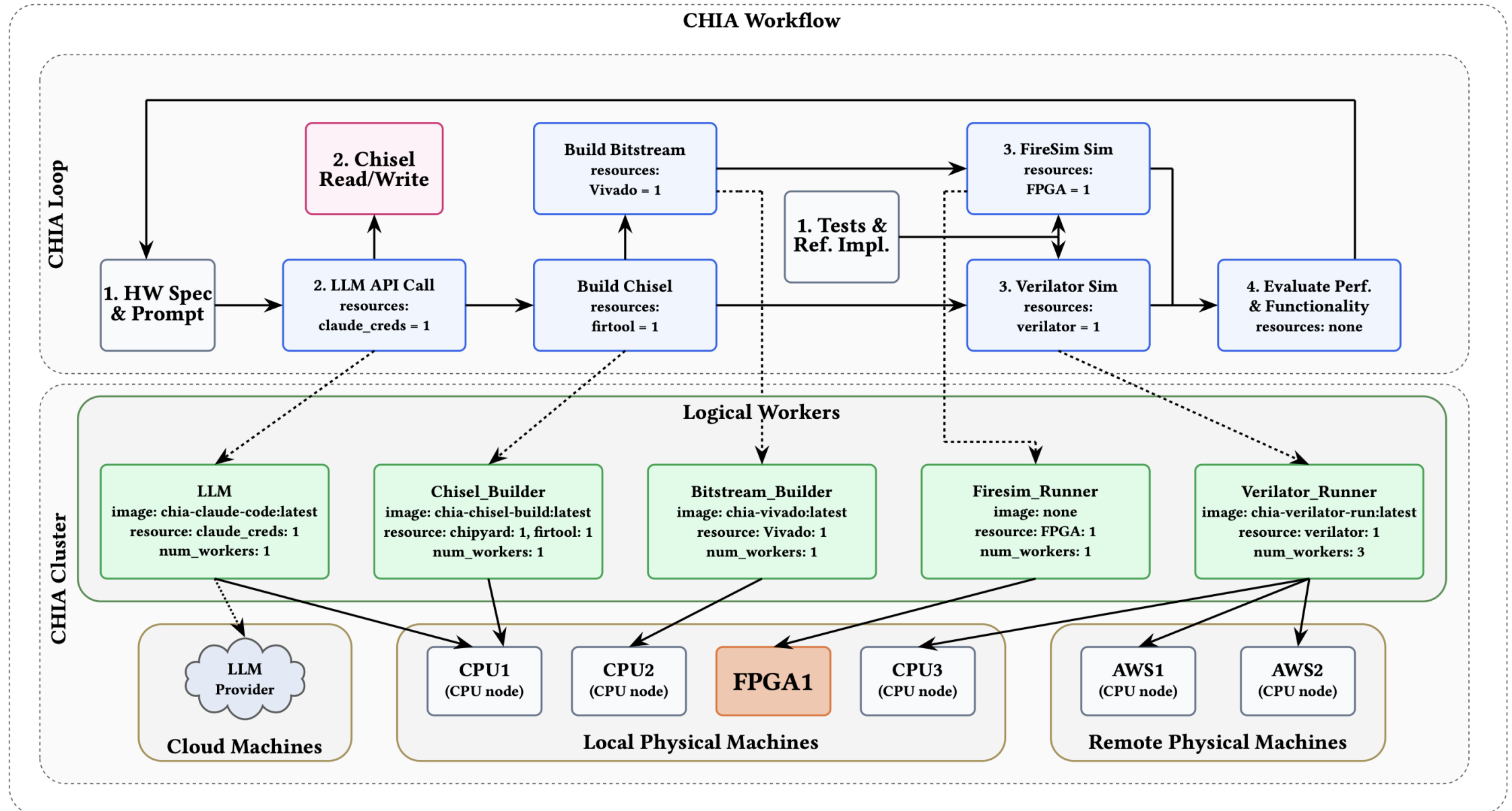
- ✓ Enables agile experimentation
- ✓ Control flow flexibility + expressiveness
- ✓ Parallelization -> Distribution
- ✓ Fault tolerance
- ▶ **Data collection, profiling, introspection**



Chia Design and Building a Chia Flow

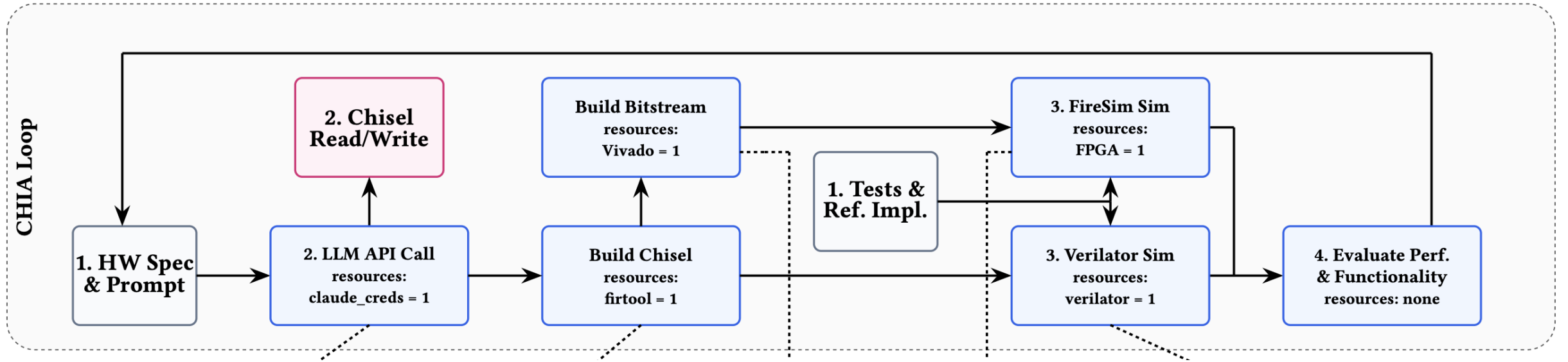


CHIA Abstractions: Workflows, Clusters, and Loops



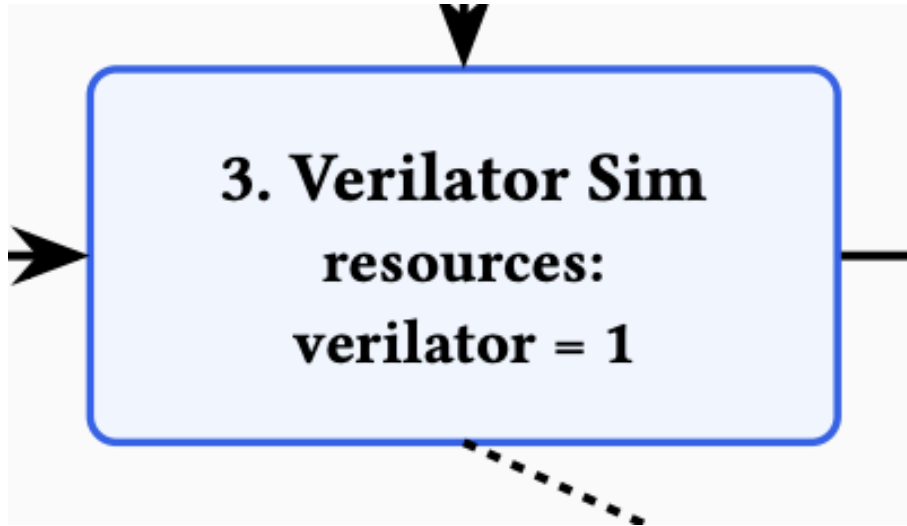


CHIA Abstractions: Loops





CHIA Abstractions: Defining and Running Nodes



The Power of ChiaFunction:



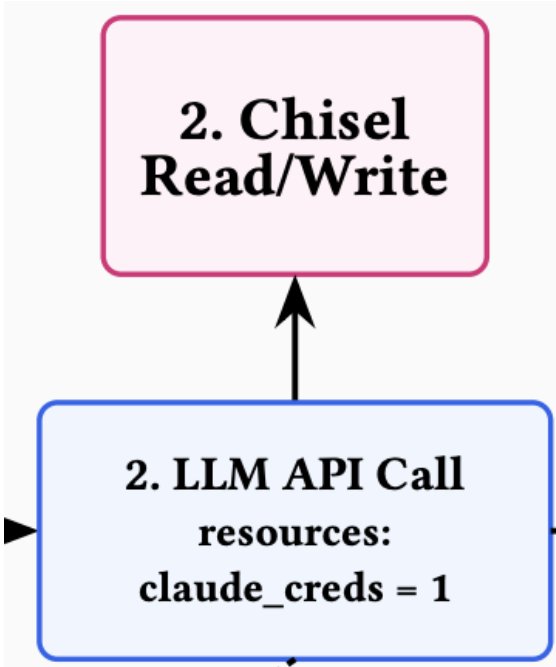
- 1) Explicit **Resource** Demands
- 2) Simple **Asynchronous/Parallel** Execution
 - 1) On **heterogeneous** substrate
 - 2) At **scale**
- 3) Built-in **Profiling**
- 4) Automatic **retry**

```
@ChiaFunction(resources={"verilator": 1.0})
def verilate(sim_binary, test_binaries):
    .../

test_results_future =
    verilate.chia_remote(sim_bin, test_bins)
```



CHIA Abstractions: Agentic Orchestration



The Power of ChiaTool:

You can give the agents exactly as much **control** of the flow as you want to give them.

Let them orchestrate everything, or just let them reason, or anything in between.

```
@ChiaFunction()
def read_src(filename) -> str: ...

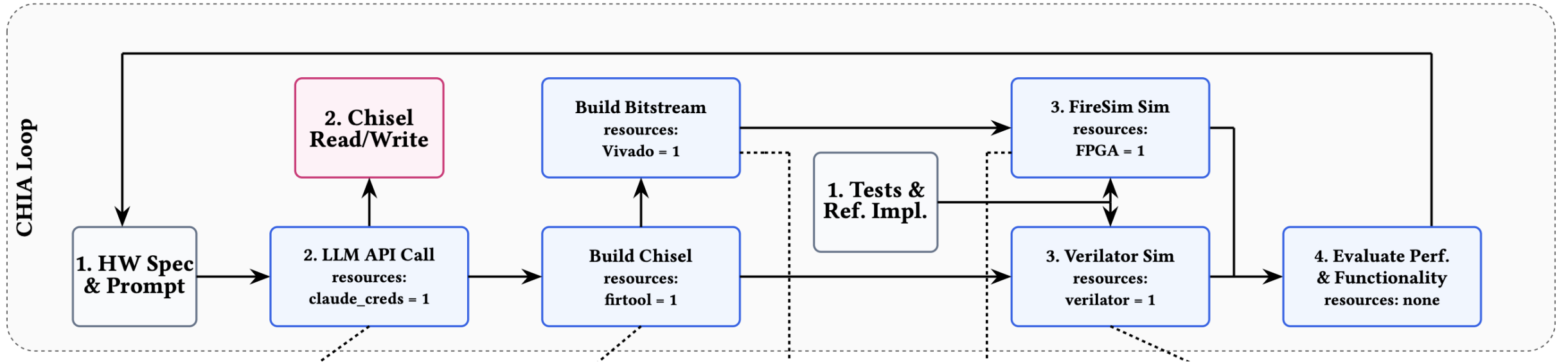
@ChiaFunction()
def write_src(filename, contents): ...

class rw_src_tool(ChiaTool):

def setup():
    self.mcp.add_tool(read_src, name="read_src_tool")
    self.mcp.add_tool(write_src, name="write_src_tool")
```

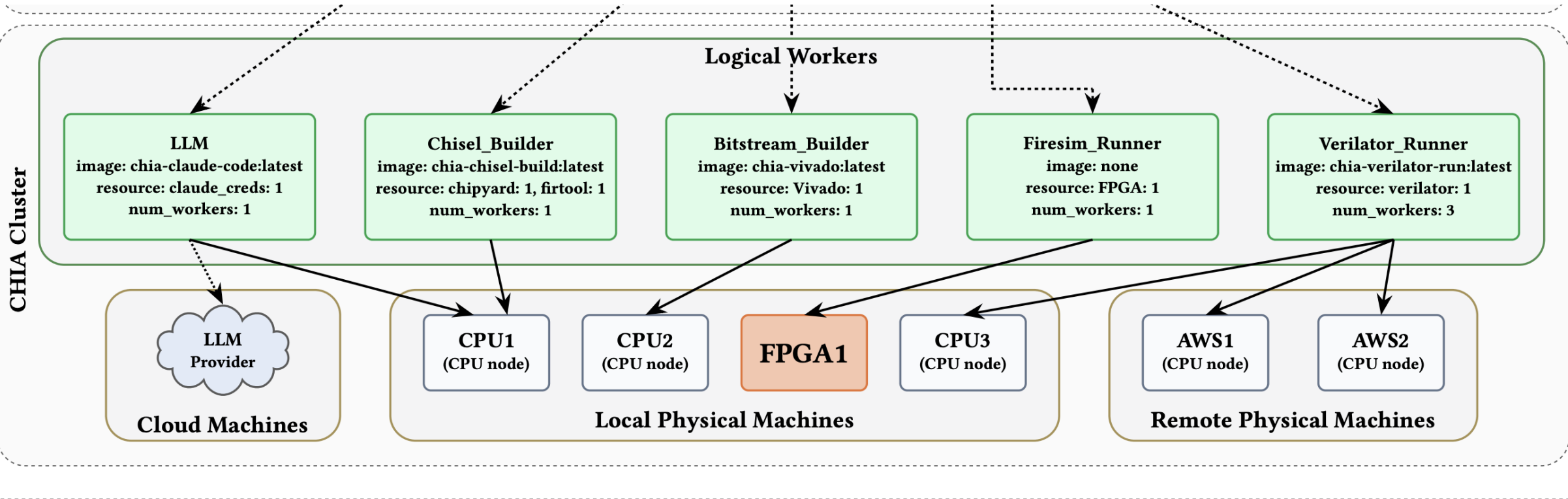


CHIA Abstractions: Loops





CHIA Abstractions: Clusters



The Power of CHIA:

Leverage on-premises and **public cloud compute** together in a single cluster.



CHIA Abstractions: Logical Workers

```
Verilator_Runner
image: chia-verilator-run:latest
resource: verilator: 1
num_workers: 3
```

```
# Spec2RTL.yaml
available_node_types:
  Verilator_Runner:
    docker:
      image: "chia-verilator-run:latest"
    resources: {"verilator": 1}
    num_workers: 3
    compatible_ips: [CPU3 \
                     @verilator_aws:0, \
                     @verilator_aws:1]
```

The Power of CHIA:

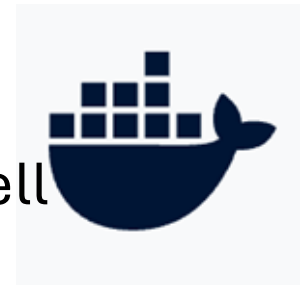


Logical Workers:

1. Clean SW **resource abstraction**
2. Abstracts heterogeneous HW **resources**

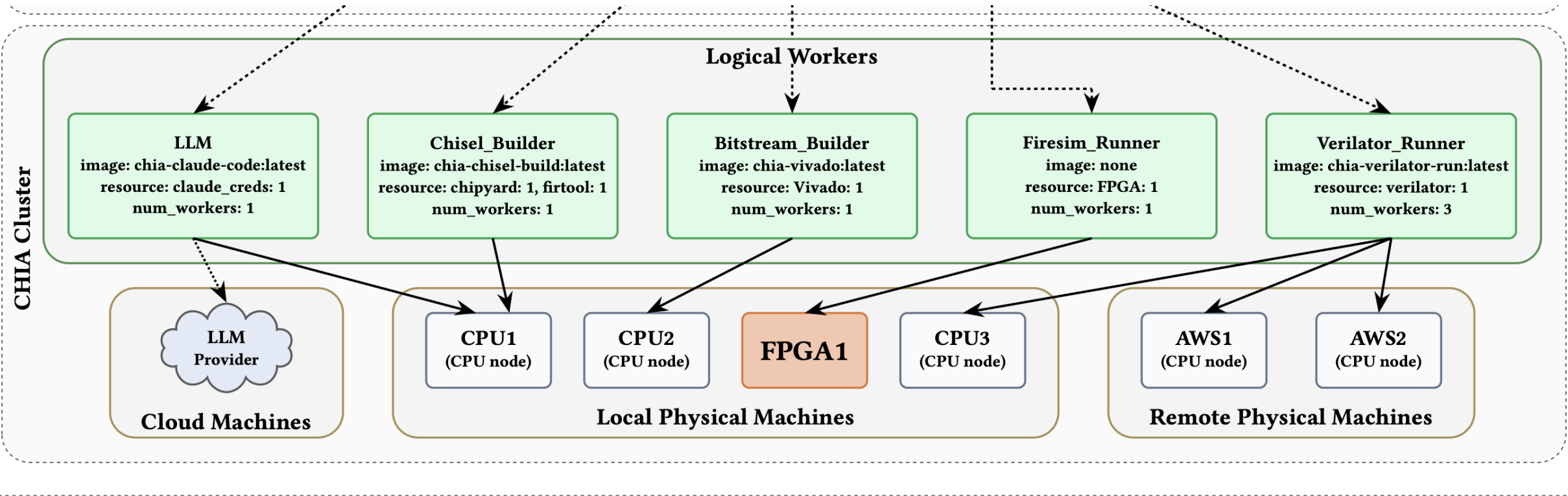
Containerization:

1. No **dependency management** hell
2. **Isolates** LLMs
3. Makes *Logical Workers* (more) **portable**



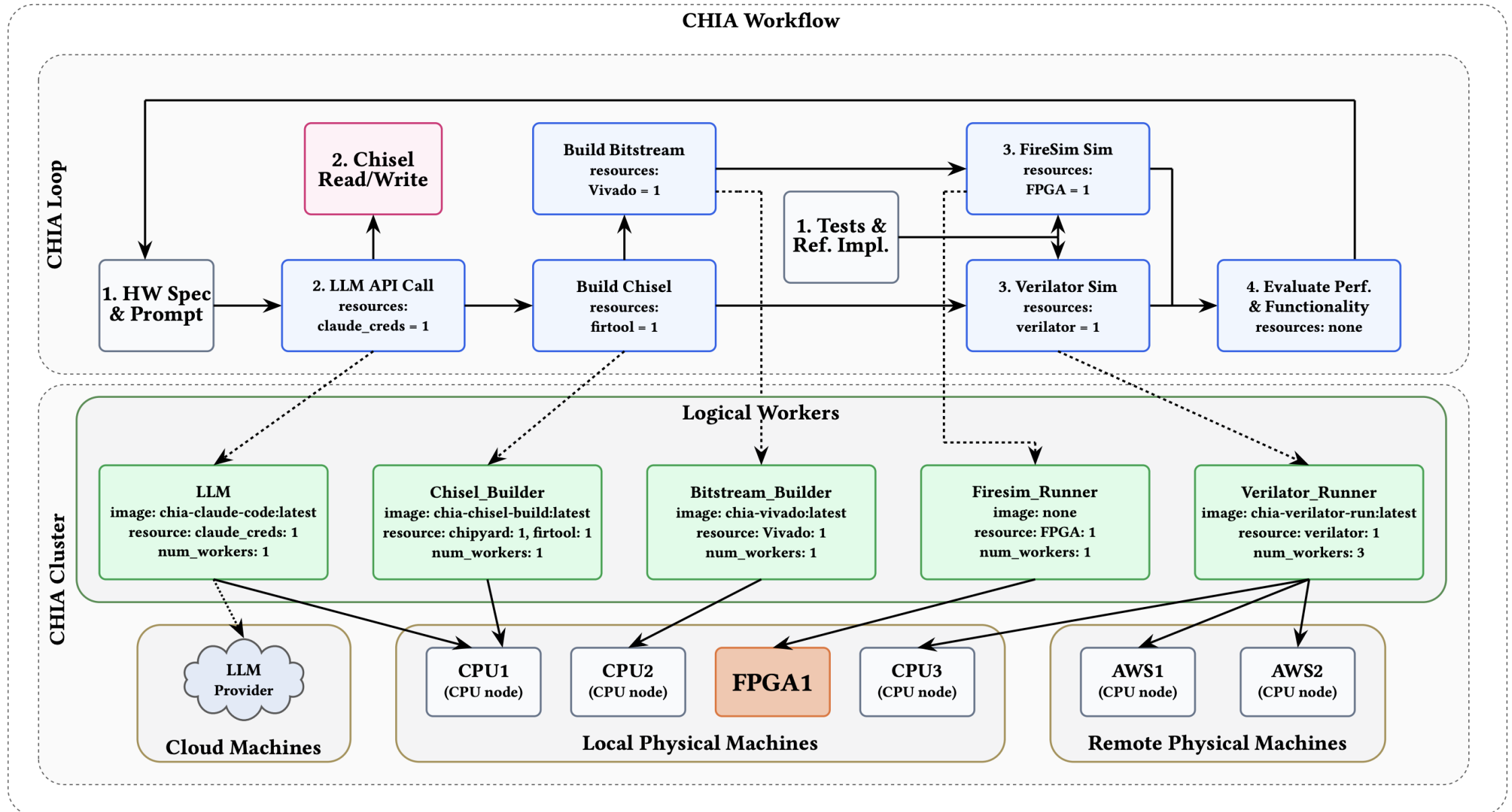


CHIA Abstractions: Clusters





CHIA Abstractions: Workflows, Clusters, and Loops





CHIA Library: CHIA works with the tools you already use!

Don't see your tool here? Consider contributing a node to CHIA!





Case Studies



Case Studies Overview (see paper for details!)

1



Agentic generation of a gem5 model from BOOM RTL

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

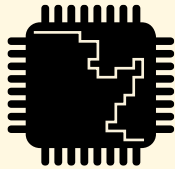
2



Agentic RISC-V ISA Extension RTL Implementation

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

3



Agentic IPC-aware Critical-Path Optimization in RTL

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

4



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



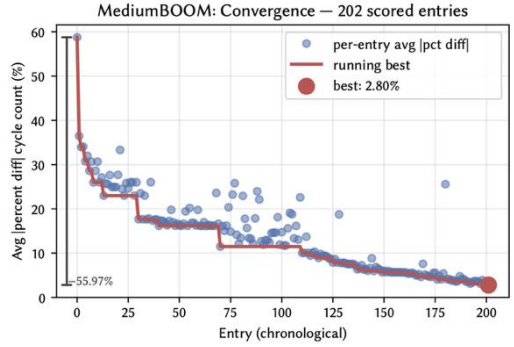
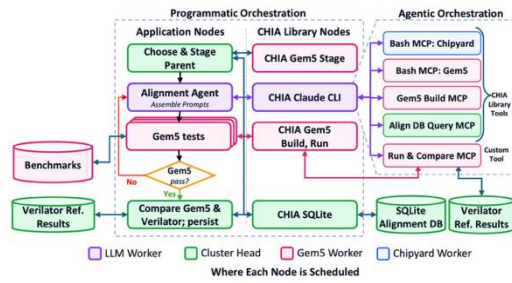
Maintainer-friendly Agentic Bug-fixing for CIRCT GitHub Issues

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

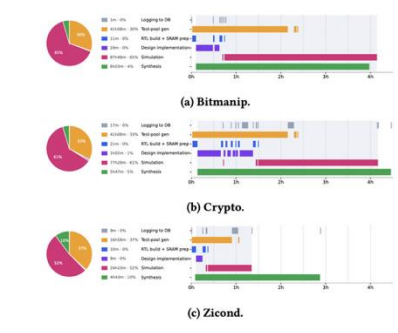
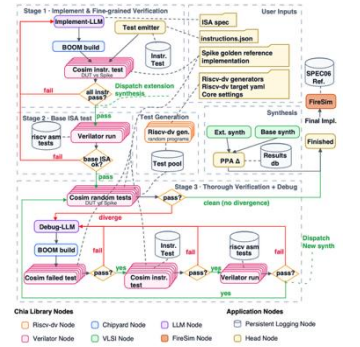


Case Studies Overview (see paper for details!)

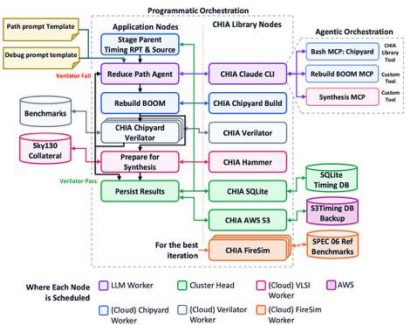
1 **gem5** **Agentic generation of a gem5 model from BOOM RTL**
 Improve gem5 core model's accuracy from 40% → 97.2% relative to ground-truth RTL simulation of BOOM over 10.5 days on 36 benchmarks



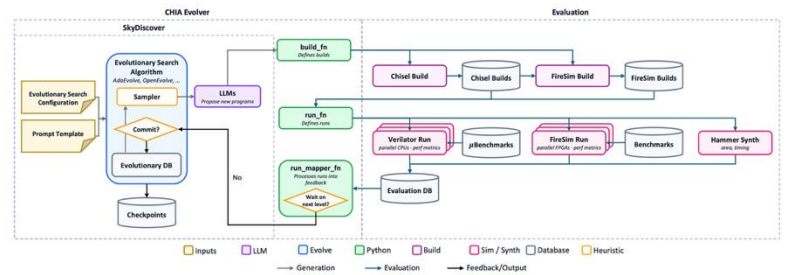
2 **RISC-V** **Agentic RISC-V ISA Extension RTL Implementation**
 5.6% (Bitmanip) and 3.5% (Zicnd) speedup on SPEC06 ref., 25+ T insts.; 10x speedup on OpenSSL (Crypto); < 5% area impact & no freq. change



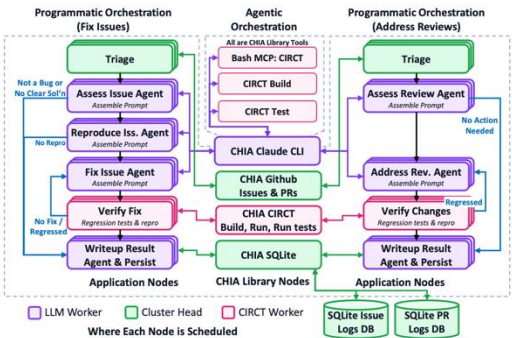
3 **Agentic IPC-aware Critical-Path Optimization in RTL**
 2.03x frequency increase with only 3.28% IPC loss on SPEC06 ref. by rewriting RTL; net 1.96x Iron Law speedup; negligible area change



4 **SkyDiscover** **Alpha Evolve** **Agentic Architecture Discovery in μArch Simulators and RTL**
 AI agent and evolutionary algorithm-directed exploration and implementation of novel microarchitectural ideas in ChampSim & BOOM




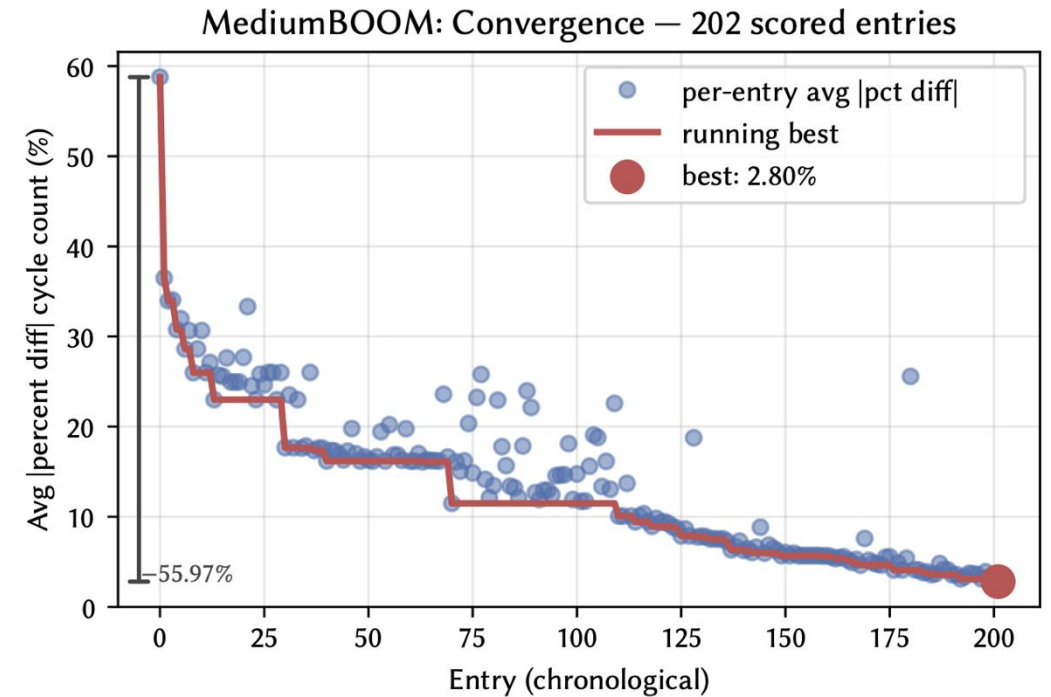
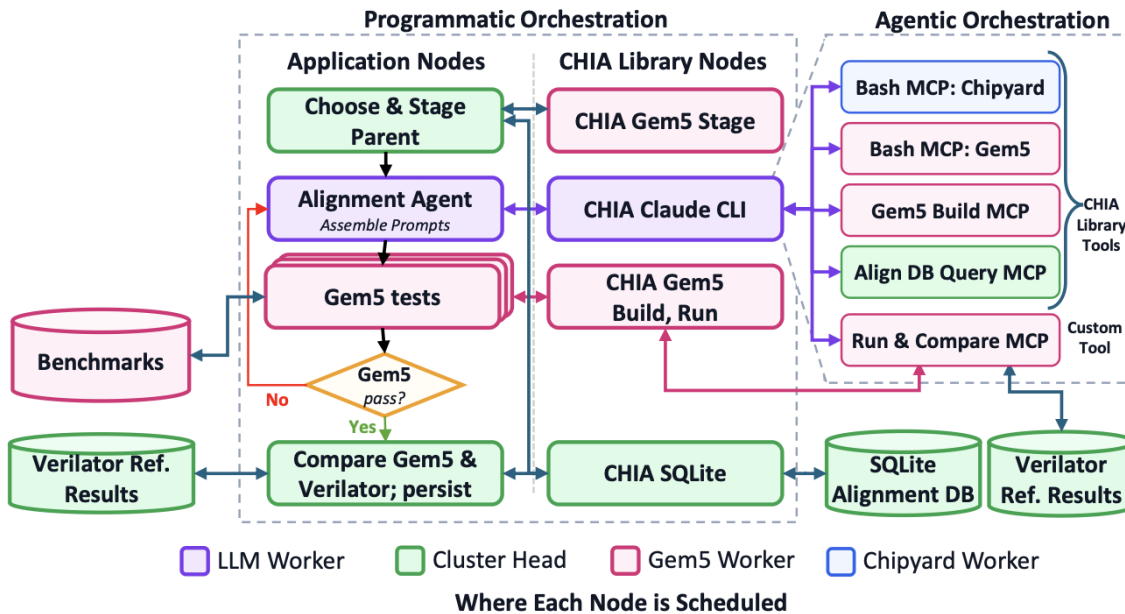
5 **Maintainer-friendly Agentic Bug-fixing for CIRCT GitHub Issues**
 Autonomously triaged CIRCT GitHub issues into actionable, high-quality, and now upstreamed PRs in ~45 minutes





Case Studies Overview (see paper for details!)

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





Case Studies Overview (see paper for details!)

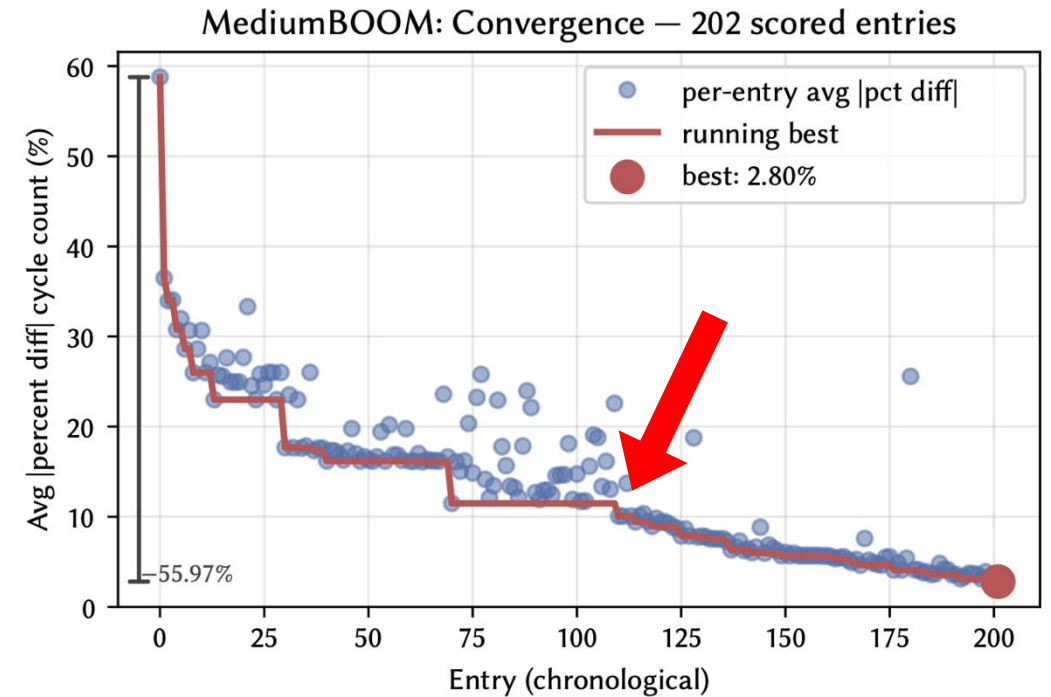
1



Agentic generation of a gem5 model from BOOM RTL

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

1. Flexibility and experimentation
2. Many loop designs tried in parallel --> easy to experiment
3. 1h to build!
4. Iteration 110:
The LLM can now test its own changes





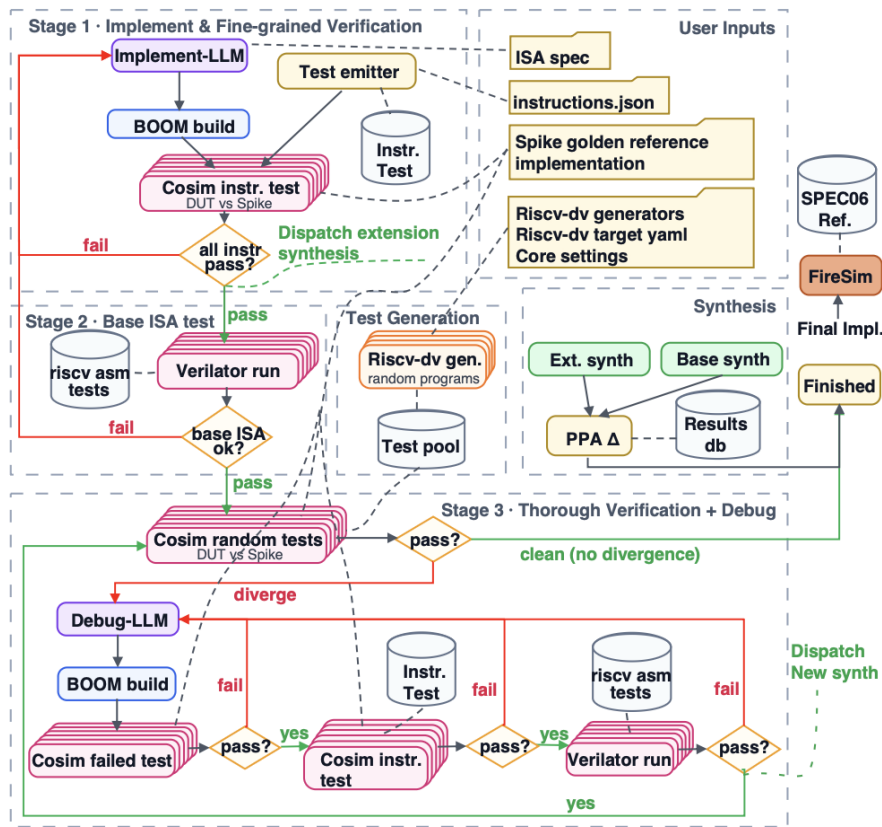
Case Studies Overview (see paper for details!)

2



Agentic RISC-V ISA Extension RTL Implementation

5.6% (Bitmanip) and 3.5% (Zicond) speedup on SPEC06 ref., 25+ T insts.; 10x 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

Application Nodes

- FireSim Node
- Head Node



Case Studies Overview (see paper for details!)

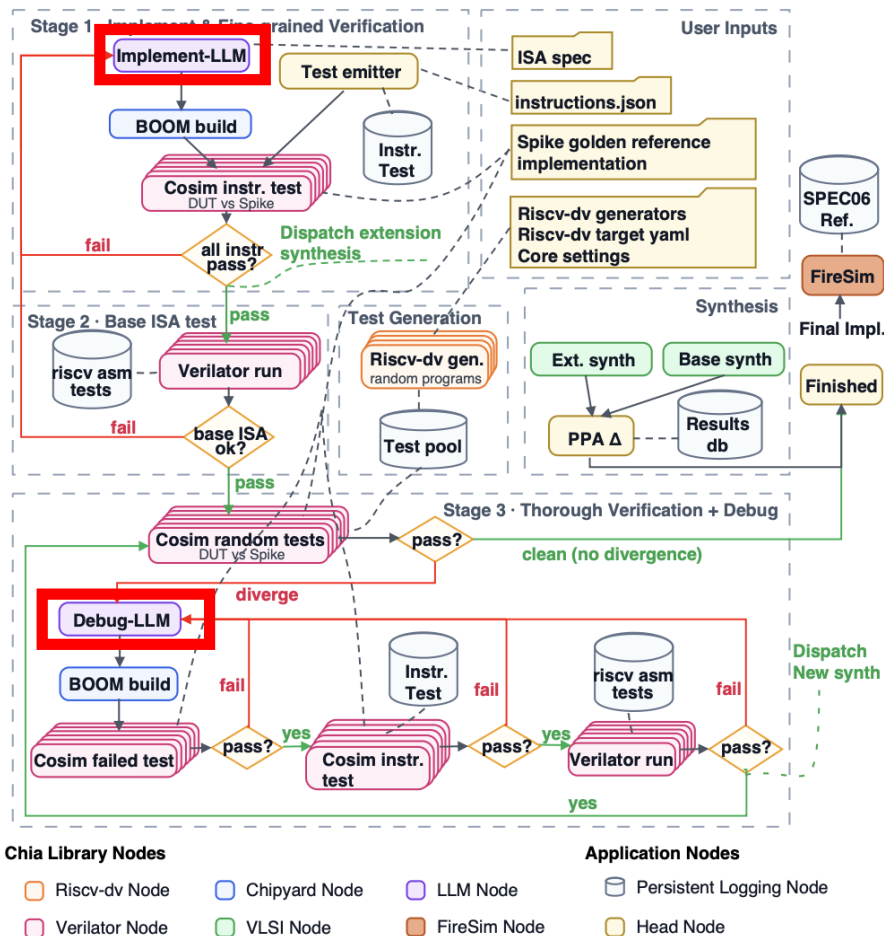
2



Agentic RISC-V ISA Extension RTL Implementation

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

1. CHIA + LLMs for specific tasks





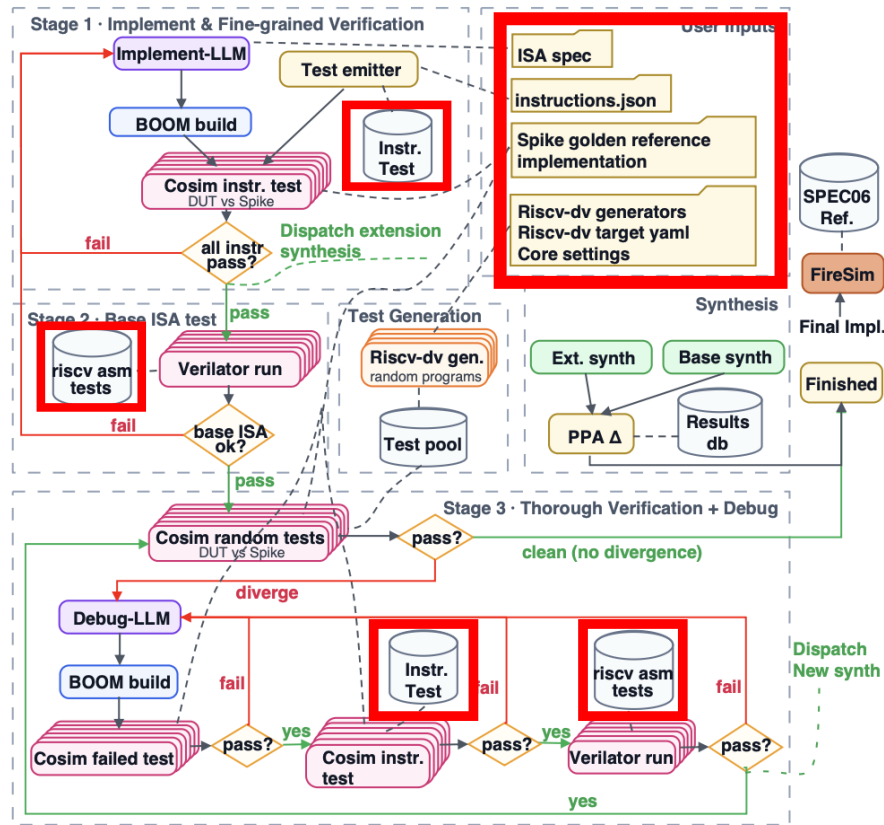
Case Studies Overview (see paper for details!)

2



Agentic RISC-V ISA Extension RTL Implementation

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



1. CHIA + LLMs for specific tasks
2. Control where it matters

Chia Library Nodes

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



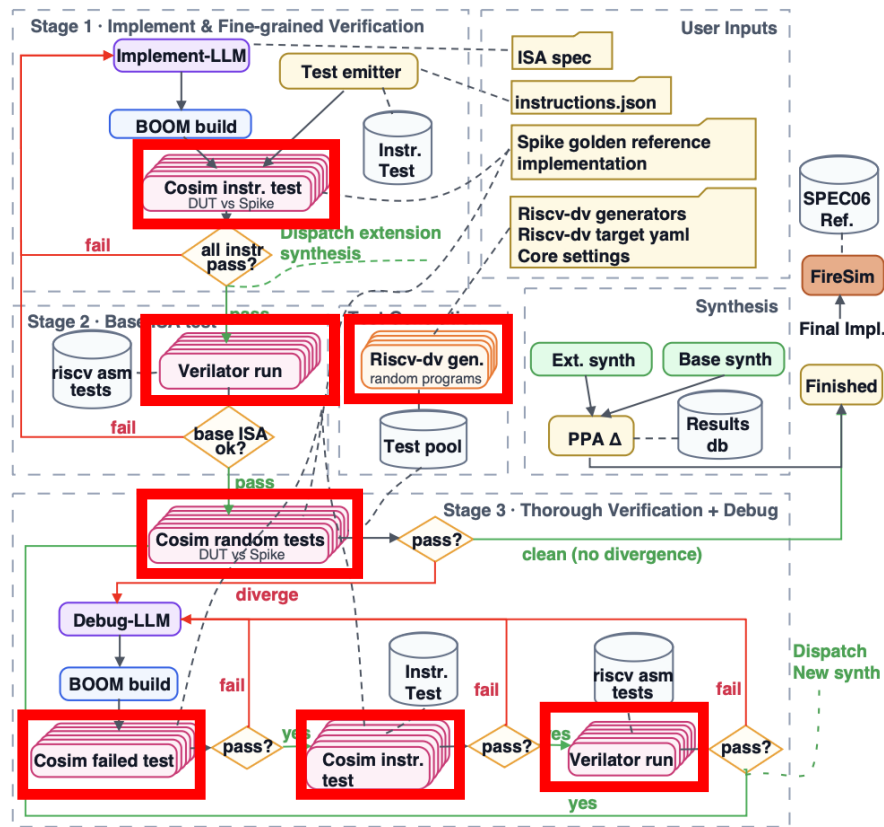
Case Studies Overview (see paper for details!)

2



Agentic RISC-V ISA Extension RTL Implementation

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



1. CHIA + LLMs for specific tasks
2. Control where it matters
3. Resource management

Chia Library Nodes

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

Application Nodes

-
-
-
-
-
-
-



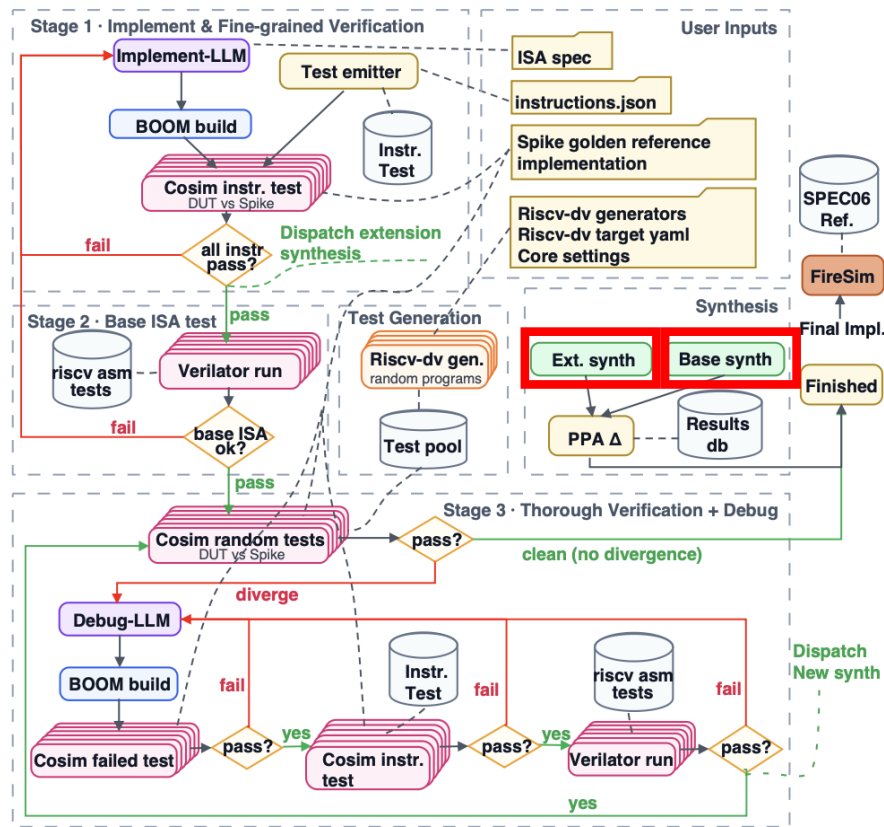
Case Studies Overview (see paper for details!)

2



Agentic RISC-V ISA Extension RTL Implementation

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



Chia Library Nodes

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

1. CHIA + LLMs for specific tasks
2. Control where it matters
3. Resource management
4. Licenses stay on-prem



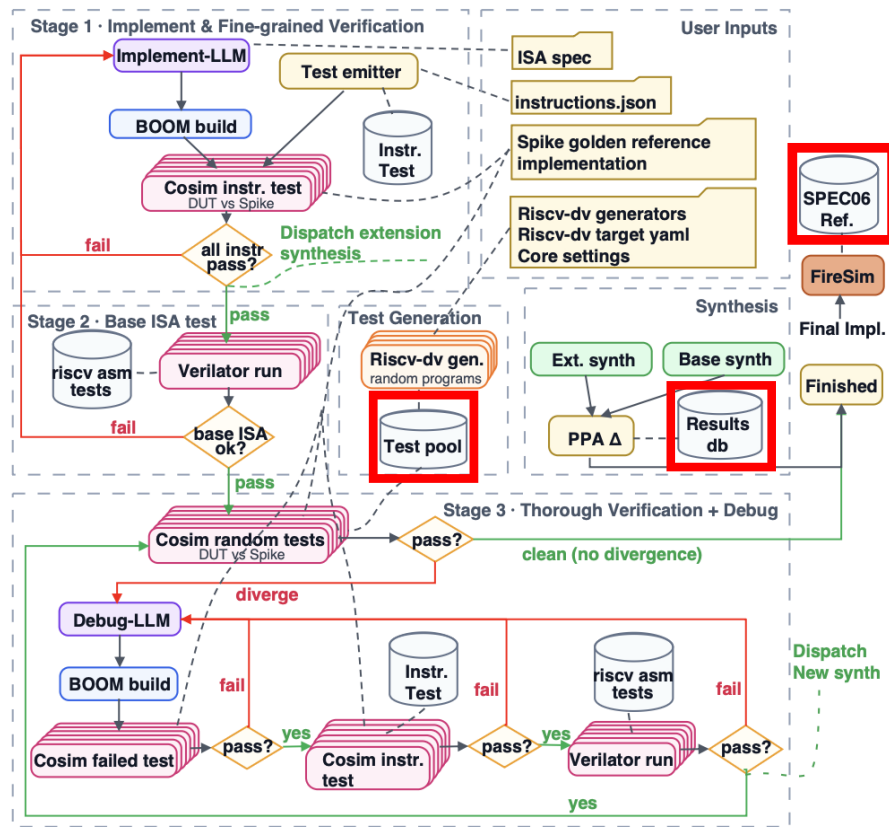
Case Studies Overview (see paper for details!)

2



Agentic RISC-V ISA Extension RTL Implementation

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



Chia Library Nodes

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

1. CHIA + LLMs for specific tasks
2. Control where it matters
3. Resource management
4. Licenses stay on-prem
5. Data logging



Case Studies Overview (see paper for details!)

2



Agentic RISC-V ISA Extension RTL Implementation

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

In less than 5h...

Bitmanip

5.6% on SPEC Ref.

Entire 25+ Trillion instr.

2% area increase, sky130

No max freq. regression

Zicond

3.5% on SPEC Ref.

Entire 25+ Trillion instr.

no area increase, sky130

No max freq. regression

Crypto

10x OpenSSL Speed Crypto

Entire 200B instructions

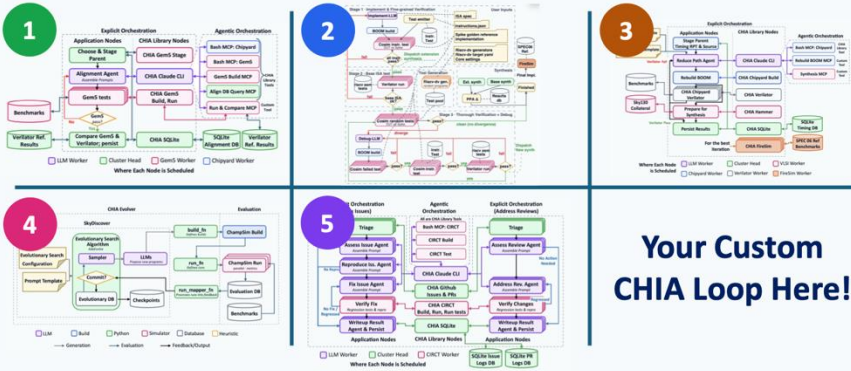
5% area increase, sky130

No max freq. regression



Summary

CHIA Loops



CHIA →


CHIA Cluster

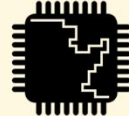
Hybrid On-Premises + Cloud Deployments



Case Studies

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.; 10x speedup on OpenSSL (Crypto); < 5% area impact & no freq. change

3  **Agentic IPC-aware Critical-Path Optimization in RTL**
2.03x frequency increase with only 3.28% IPC loss on SPEC06 ref. by rewriting RTL; net 1.96x Iron Law speedup; negligible area change

4  **Agentic Architecture Discovery in μArch 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 is hot off the presses, available to use!

<https://chialoops.ai>

<https://docs.chialoops.ai>

CHIA

About Pre-print Paper Blog Docs User Forum X Auto

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 experience](#) [Toolflow compatibility](#) [Case studies](#) [Learn more](#) [Latest 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.

```
~$ chia job submit -- bitmanip-flow.py
• LLM is implementing RISC-V Bitmanip extension in BOOM RTL.
```

Example CHIA Loop for RISC-V Extension RTL Implementation

```
graph TD
    A[Create RTL] --> B[BOOMChipyard SoC Site]
    B --> C[RISC-V (Verilator)]
```

MICRO 2026 tutorial!

CHIA

Welcome to CHIA's documentation! [View page source](#)

Welcome to CHIA's documentation!

CHIA provides a powerful and flexible orchestration of hardware design workflows — where all your hardware design tools all work together.

[Getting started](#), or go straight to [Installation](#) to set things up.

CHIA is currently in a beta phase. Many library nodes have not received thorough testing (but are available with Intelligent Agents).

[Getting started](#) (with Intelligent Agents)

[Installation](#) (with Intelligent Agents)

[Getting started](#) (Choosing an Acronym)

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

CLI REFERENCE

CLI Reference

API REFERENCE

Base Runtime
Chipyard Nodes
Model Backends
FireSim
VLSI

Concepts

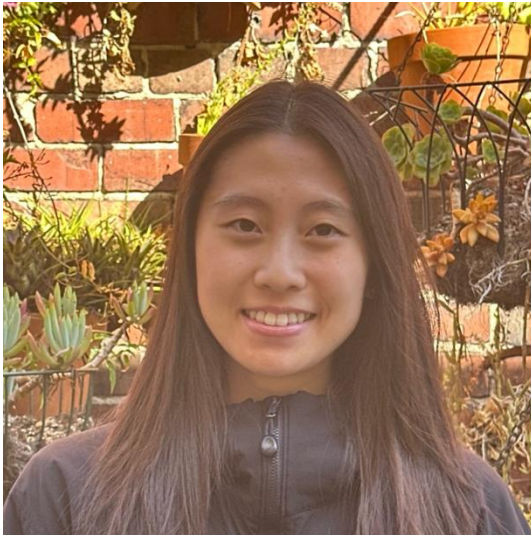
- Architecture Overview
 - CHIA flows
 - CHIA clusters



CHIA

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

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



Angela Cui



Ferran Hermida-Rivera



Jack Toubes