中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

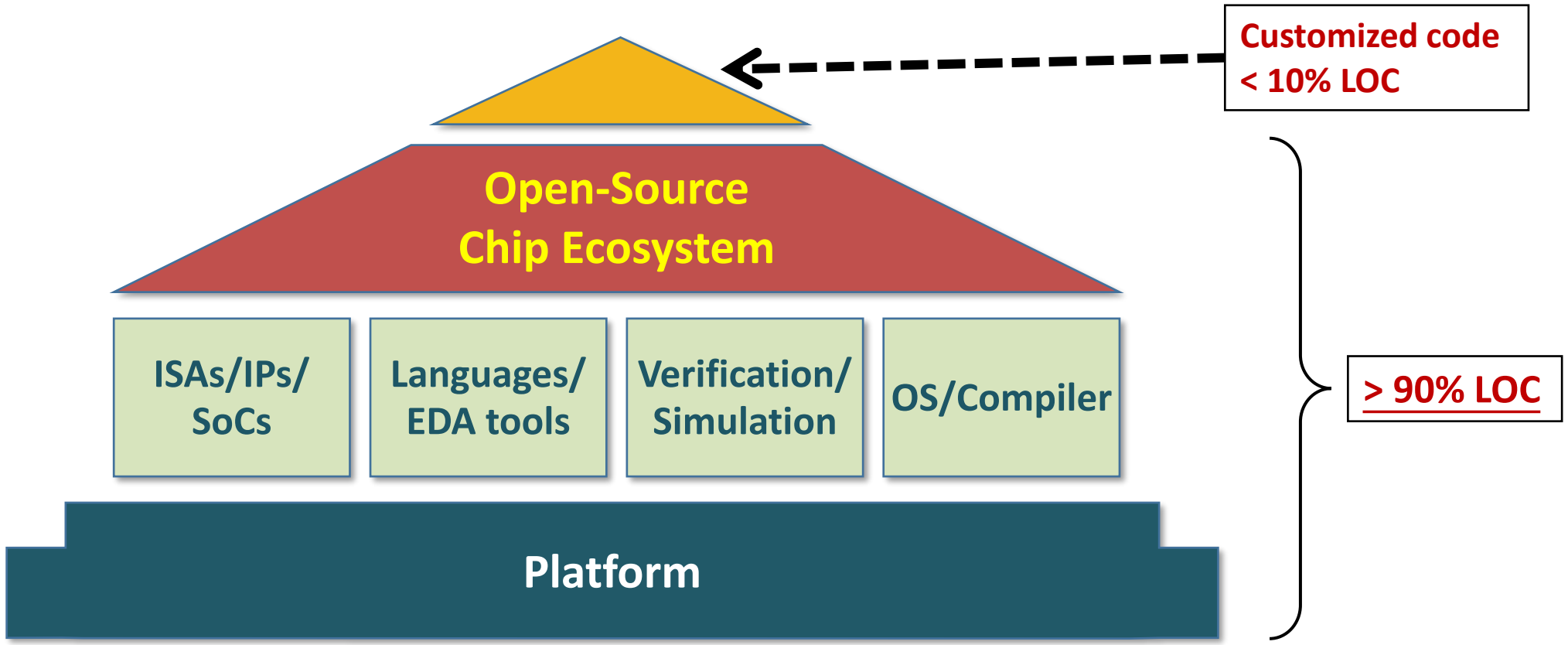
At the 55th IEEE/ACM International Symposium on Microarchitecture (MICRO), October 1–5, 2022

Towards Developing High Performance RISC-V Processors Using Agile Methodology

Yinan Xu⁎†, Zihao Yu⁎, Dan Tang⁎‡, Guokai Chen⁎†, Lu Chen⁎†, Lingrui Gou⁎†, Yue Jin⁎†, Qianruo Li⁎†, Xin Li⁎†, Zuojun Li⁎†, Jiawei Lin⁎†, Tong Liu⁎, Zhigang Liu⁎, Jiazhan Tan⁎, Huaqiang Wang⁎†, Huizhe Wang⁎†, Kaifan Wang⁎†, Chuanqi Zhang⁎†, Fawang Zhang ǁ, Linjuan Zhang⁎†, Zifei Zhang⁎†, Yangyang Zhao⁎, Yaoyang Zhou⁎†, Yike Zhou⁎, Jiangrui Zou ǁ, Ye Cai ǁ, Dandan Huan¶, Zusong Li¶, Jiye Zhao¶, Zihao Chen§, Wei He§, Qiyuan Quan§, Xingwu Liu⁎⁎, Sa Wang⁎†, Kan Shi⁎, Ninghui Sun⁎† and Yungang Bao⁎†

⁎State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences
†University of Chinese Academy of Sciences
‡Beijing Institute of Open Source Chip
§Peng Cheng Laboratory
¶Beijing VCore Technology
ǁShenzhen University
⁎⁎Dalian University of Technology

The Era of Agile and Open-Source Hardware



To lower the barrier of chip development

by saving time-to-market and the cost of IPs, EDA tools, facilities and engineers etc.

Agile and Open-Source Chip Ecosystem

Customized code < 10% LOC

Open-Source Chip Ecosystem

Open-Source > 90%

ISAs/IPs/SoCs	Languages/EDA tools	Verification/Simulation	OS/Compiler
RISC-V	CHISEL	OVP imperas	Linux
XIANGSHAN	bluespec	CHIPKIT	RT-Thread
Rocket Chip Generator	SpinalHDL	VERILATOR	freeRTOS
OpenPiton	PyMTL	ESSENT: A High-Performance RTL Simulator	GCC
BlackParrot	OpenROAD	FireSim	

Platform

Agile Approaches Adopted in the Industry

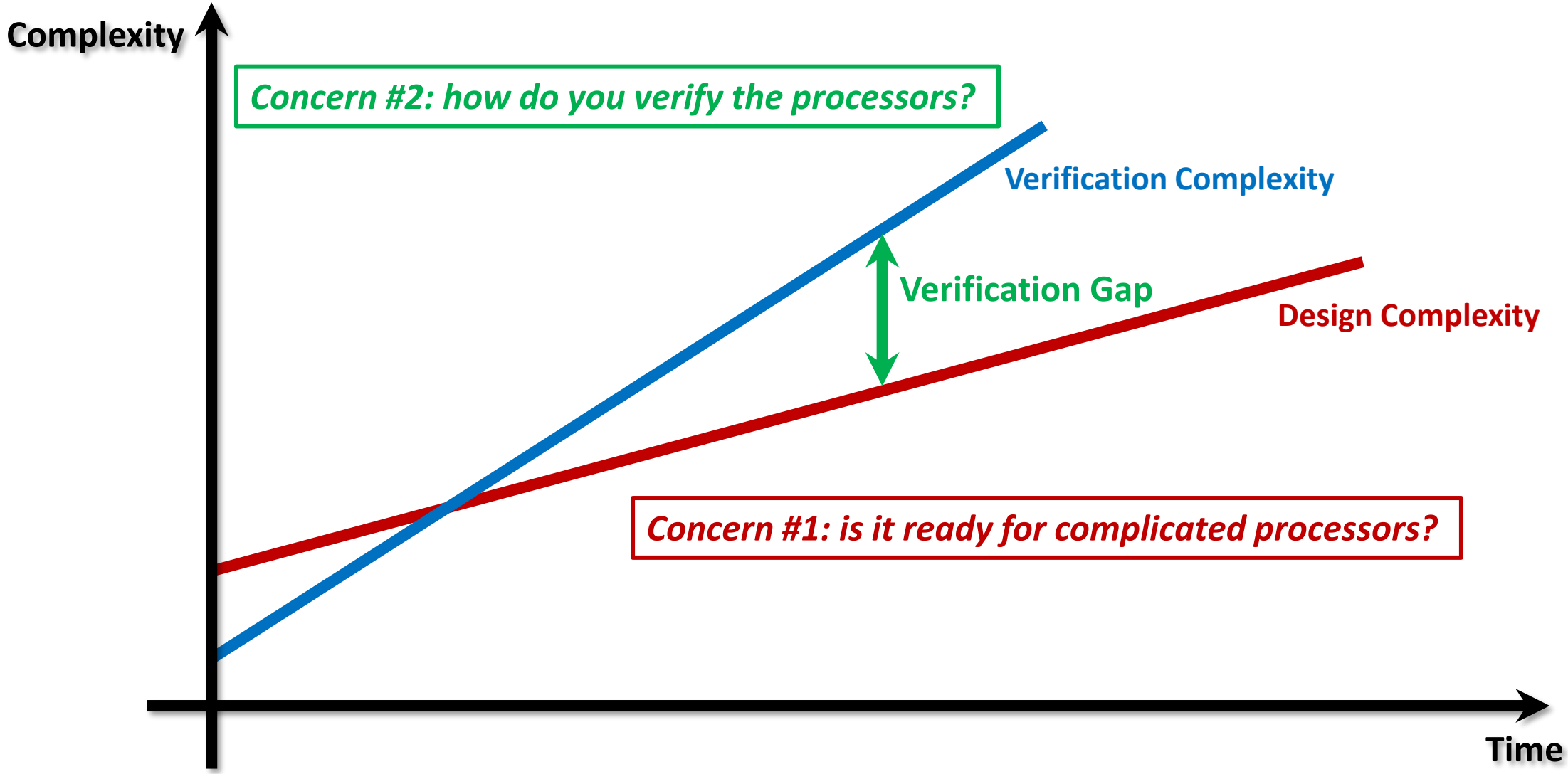
- **When we talked to some of the leading companies …**

We:

ISAs/IPs/ SoCs	Languages/ EDA tools	Verification/ Simulation	OS/Compiler

Big Ones:

Major Concerns Regarding the Agile Methodology



Source: Serge Leef, Reimagining Digital Simulation, DAC 2021.

This Work: Let's *Do It* and See What's Happening

We:

ISAs/IPs/ SoCs | Languages/ EDA tools | Verification/ Simulation | OS/Compiler

Big Ones:

Concern #1: it's not ready for complicated processors.

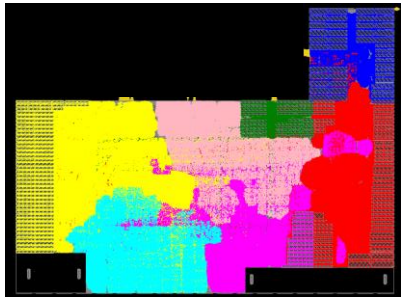
Concern #2: the verification process is still less agile.

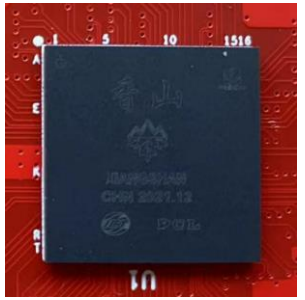
We:

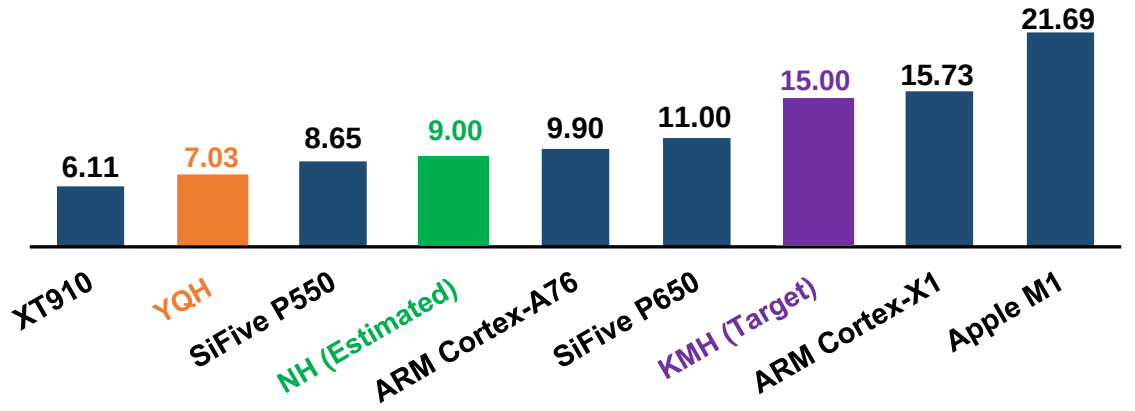
XiangShan: High Performance RISC-V Processors

XiangShan: Open-Source High Performance Processors

- 1st generation: YQH
  - RV64GC, single-core, superscalar OoO
  - **28nm tape-out, 1.3GHz, July 2021**
  - **SPEC CPU2006 7.01@1GHz, DDR4-1600**
- 2nd generation: NH
  - RV64GCBK, dual-core, superscalar OoO
  - **Scheduled 2GHz@14nm tape-out, Q4 2022**
  - **Estimated** SPEC CPU2006 19.45@2GHz**
- 3rd generation: KMH
  - RV64GCBKHV, quad-core, superscalar OoO
  - **Close collaboration with industrial partners**





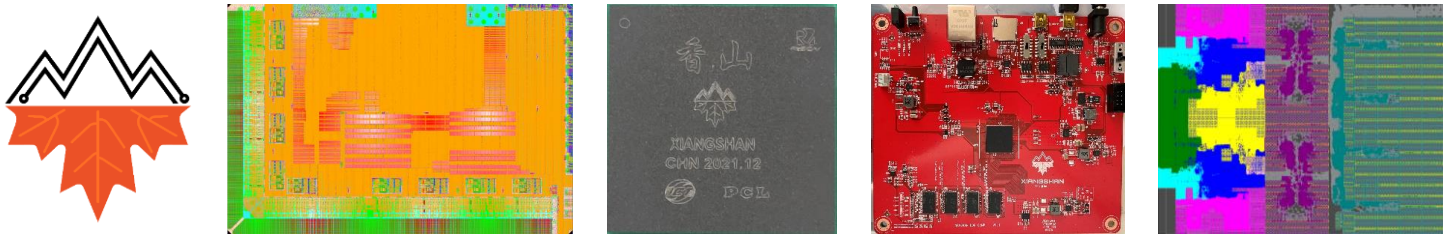


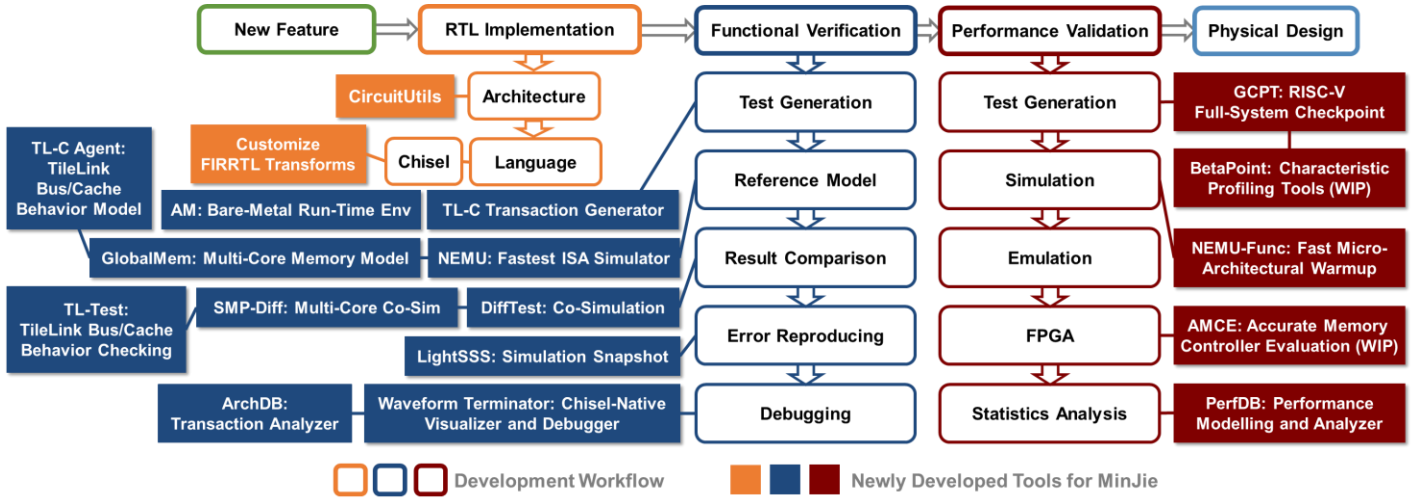
SPECint 2006/GHz (Proportional to IPC)*

* Source: XT910@ISCA'20, SiFive, AnandTech
** Updated September 14, 2022

How We Built XiangShan: MinJie

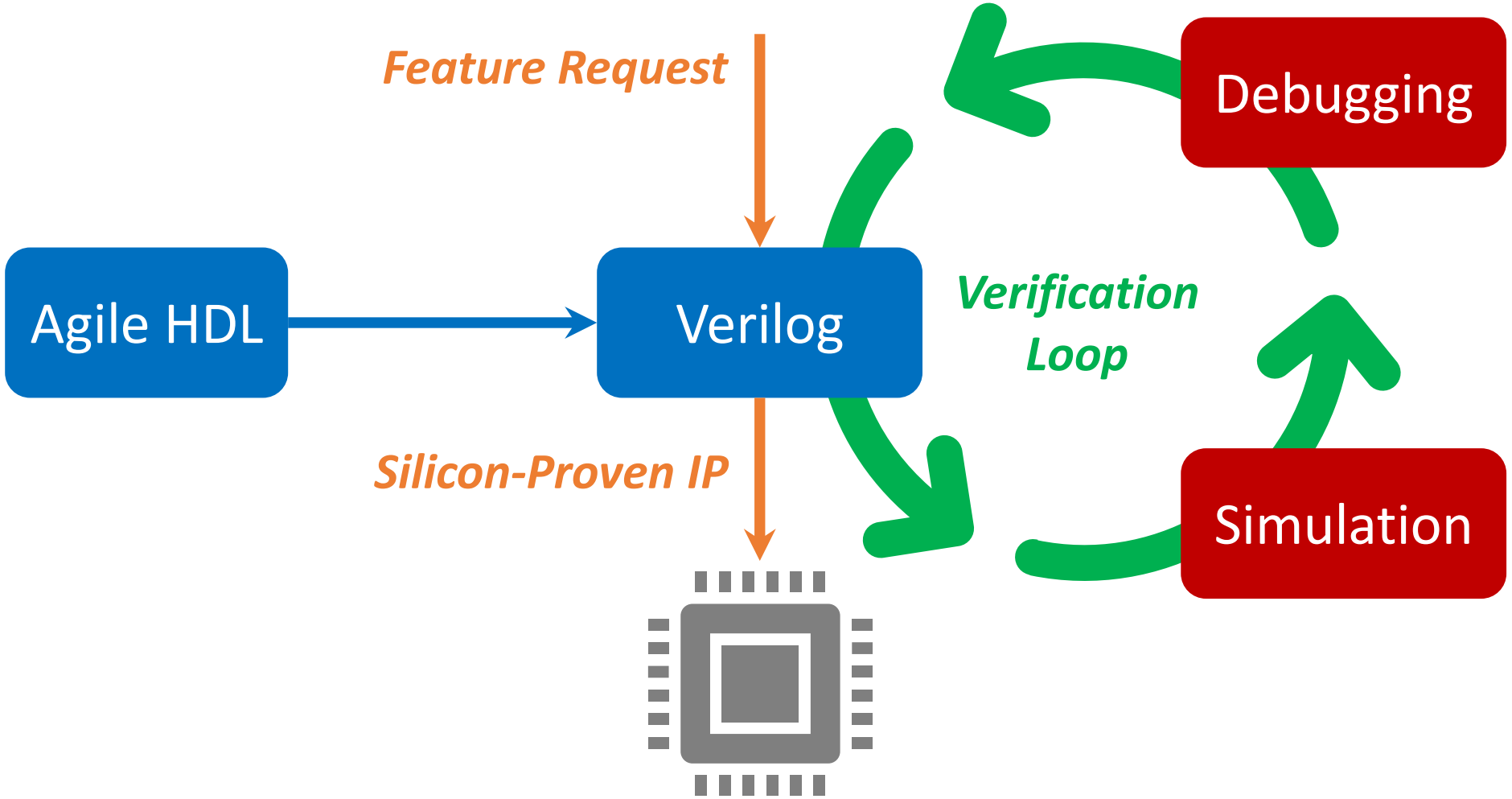
XiangShan: Open-Source High Performance RISC-V Processor



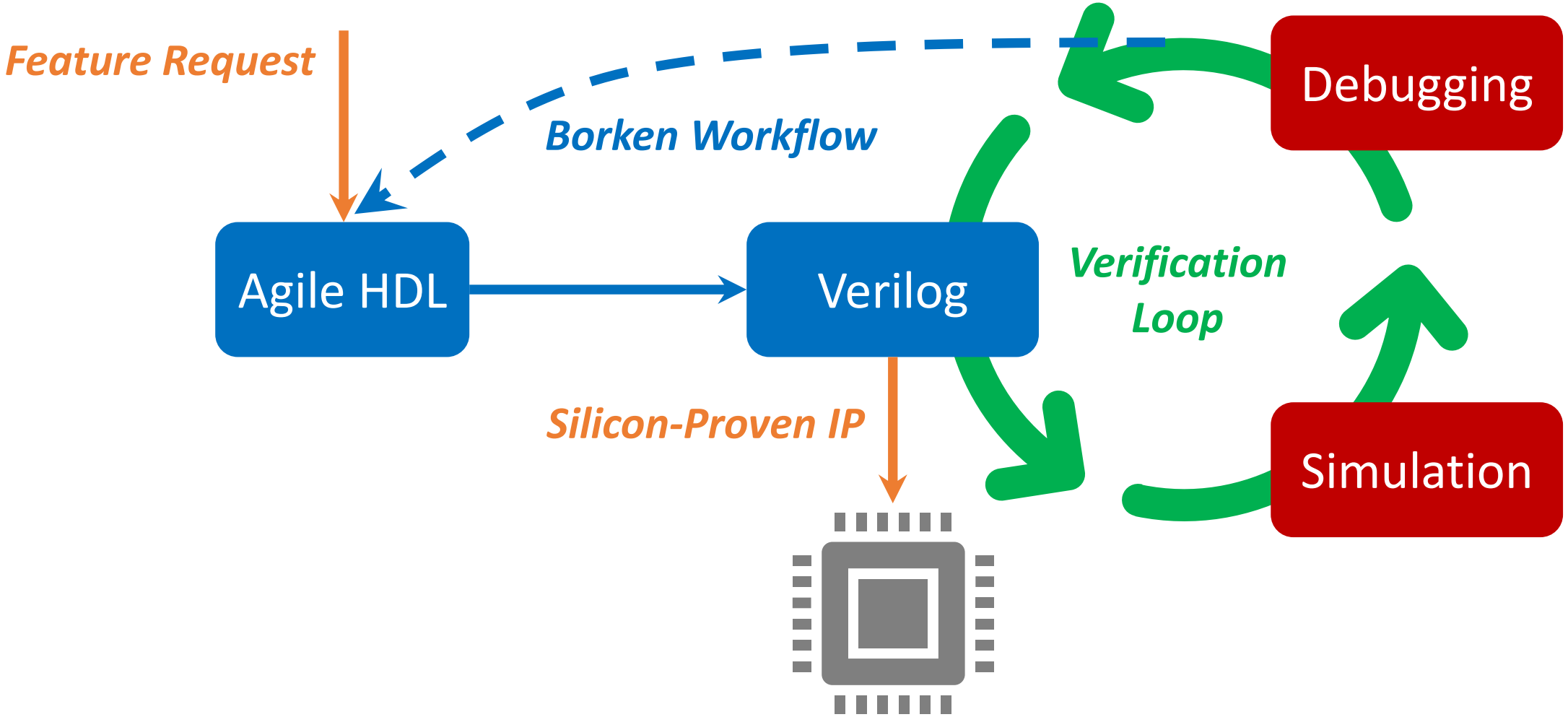


MinJie: Open-Source Platform with Agile Development Flows and Tools

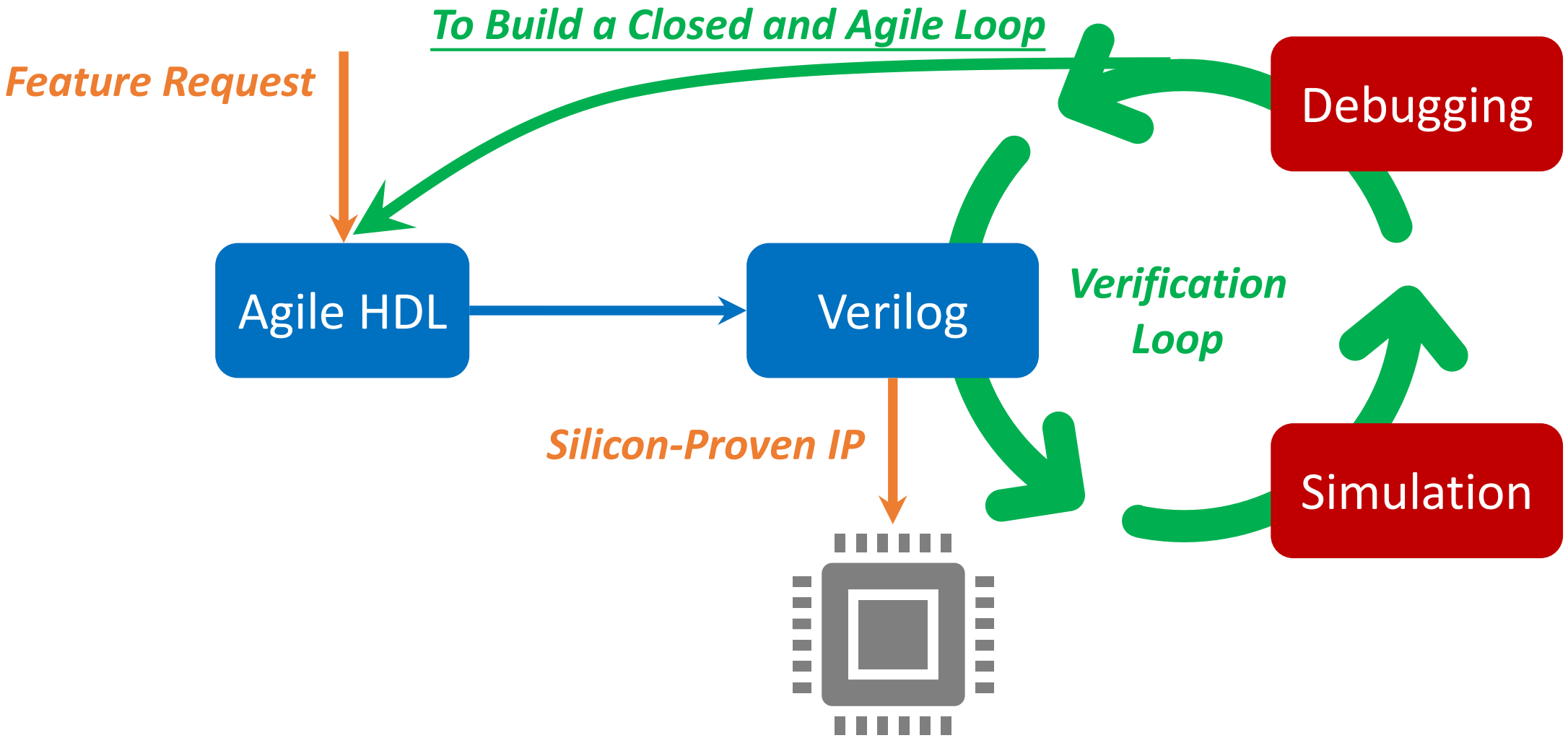
MinJie: Platform with Agile Development Flows and Tools



MinJie: Platform with Agile Development Flows and Tools



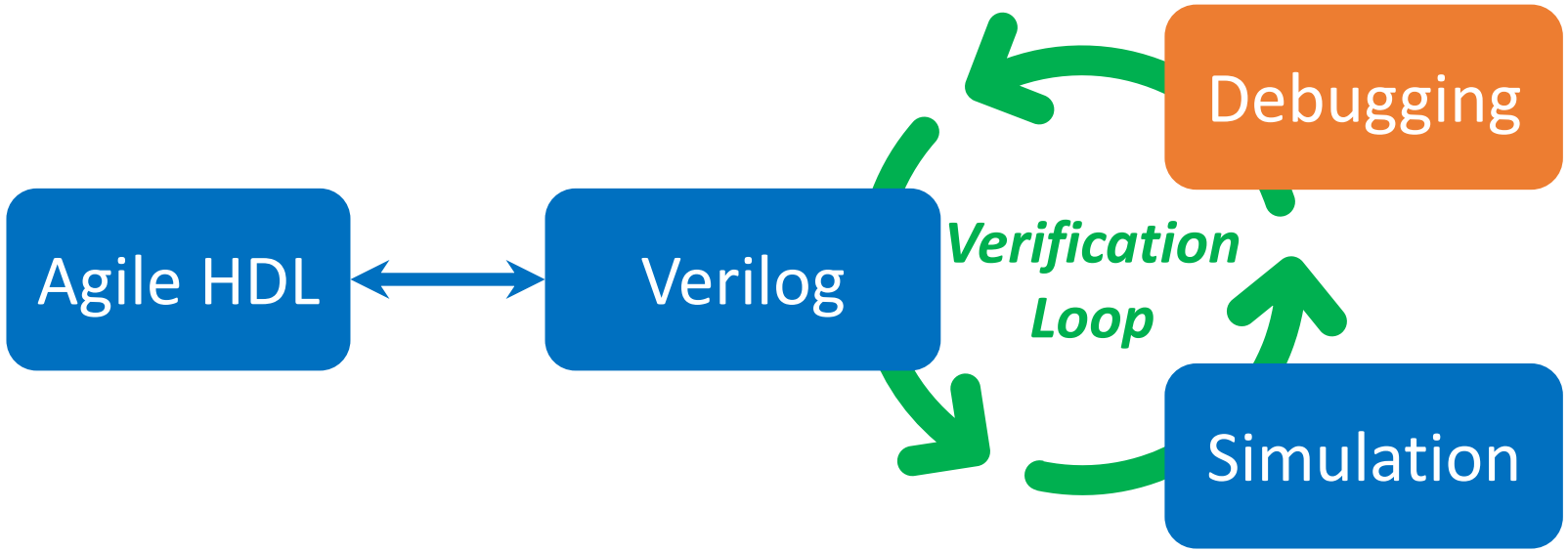
MinJie: Platform with Agile Development Flows and Tools



This Work

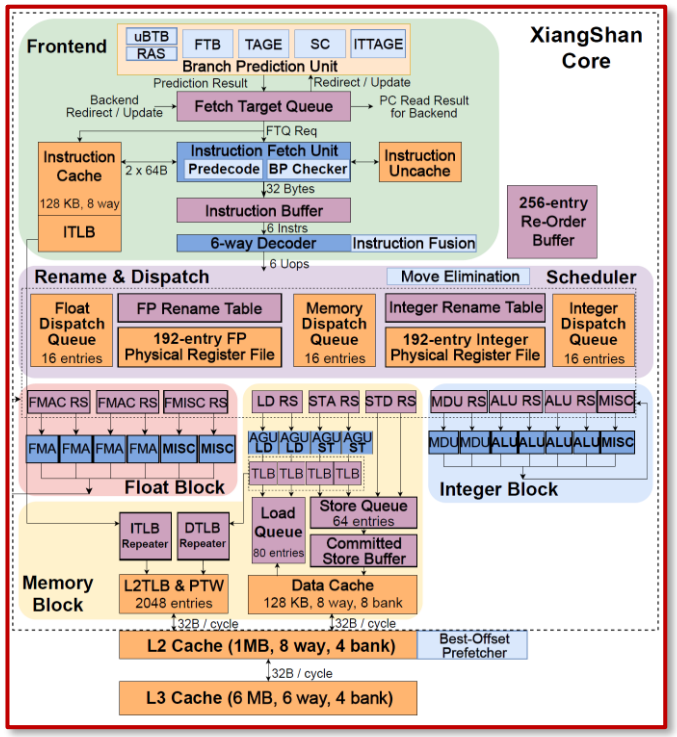
- ***XiangShan***: **High Performance RISC-V Processor**

- ***MinJie***: **Platform with Agile Development Flows and Tools**
  - *DRAV and DiffTest: Functional Verification*
  - *LightSSS: Debugging*

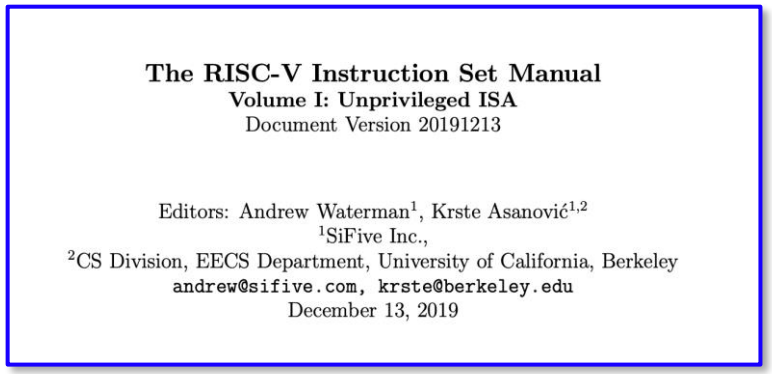


Processor Functional Verification

RISC-V Processor

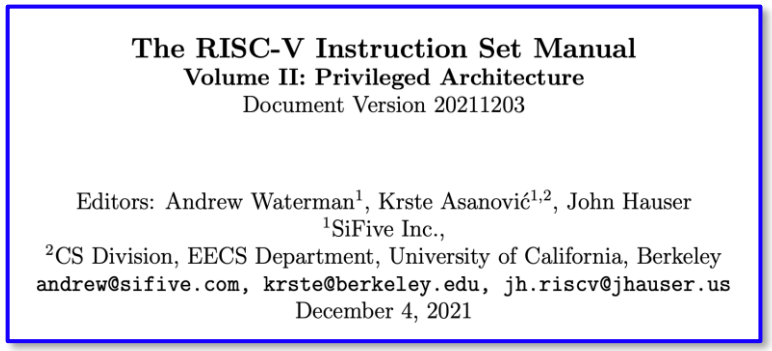


RISC-V Specifications



The RISC-V Instruction Set Manual
Volume I: Unprivileged ISA
Document Version 20191213

Editors: Andrew Waterman[1], Krste Asanović[1,2]
[1]SiFive Inc.,
[2]CS Division, EECS Department, University of California, Berkeley
andrew@sifive.com, krste@berkeley.edu
December 13, 2019

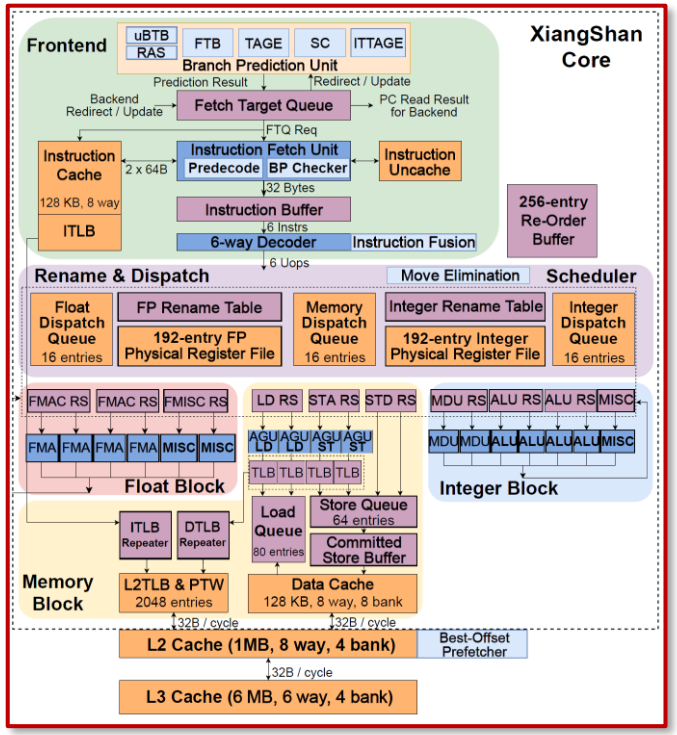


The RISC-V Instruction Set Manual
Volume II: Privileged Architecture
Document Version 20211203

Editors: Andrew Waterman[1], Krste Asanović[1,2], John Hauser
[1]SiFive Inc.,
[2]CS Division, EECS Department, University of California, Berkeley
andrew@sifive.com, krste@berkeley.edu, jh.riscv@jhauser.us
December 4, 2021

An ideal verification world:

Processor Functional Verification: Reality



RISC-V Processor

?

Spike RISC-V ISA Simulator

QEMU

imperas OVP Open Virtual Platforms

NEMU

RISCV Sail Model

Dromajo

Yet Another RISC-V Processor

=

The RISC-V Instruction Set Manual
Volume I: Unprivileged ISA
Document Version 20191213

Editors: Andrew Waterman[1], Krste Asanović[1,2]
[1]SiFive Inc.,
[2]CS Division, EECS Department, University of California, Berkeley
andrew@sifive.com, krste@berkeley.edu
December 13, 2019

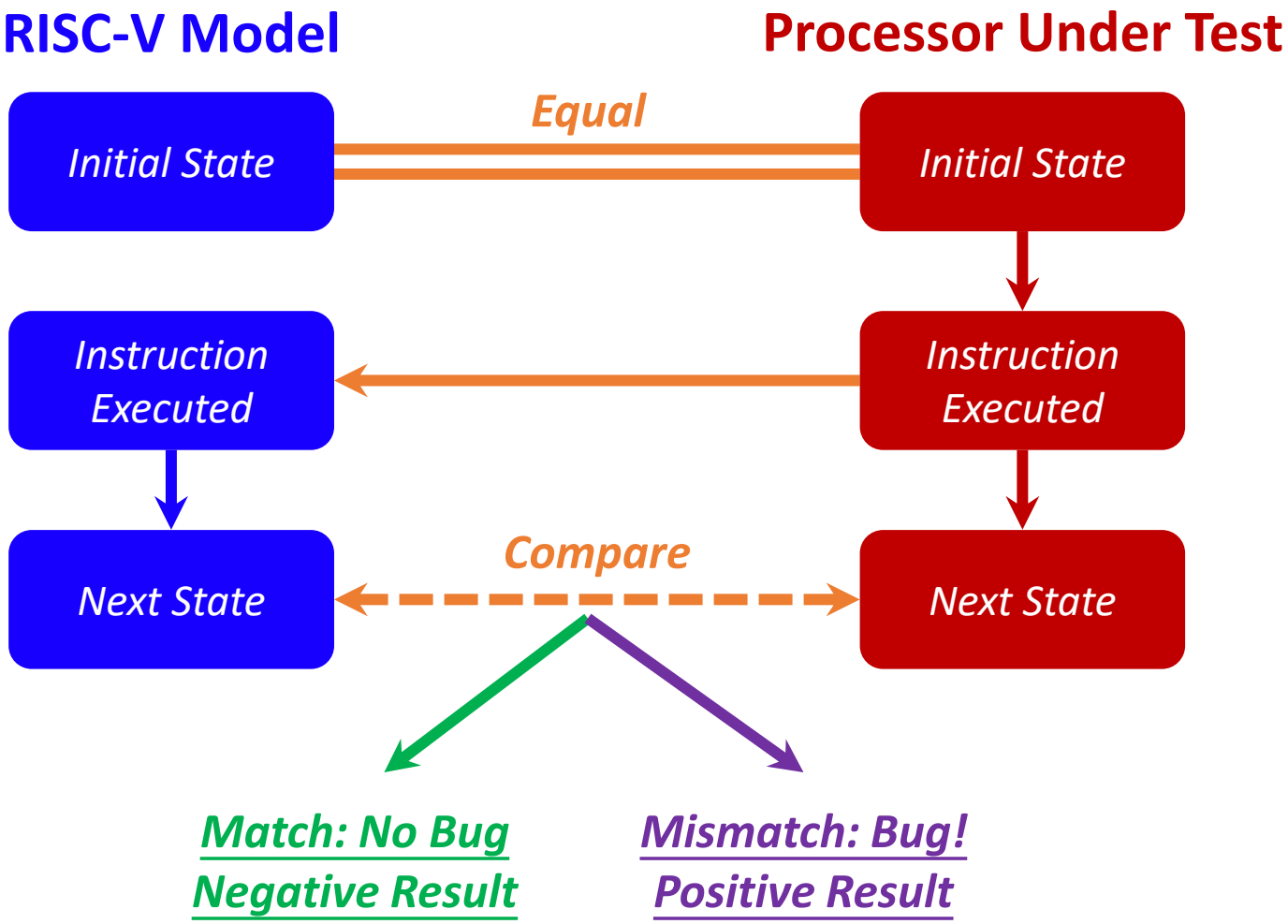
The RISC-V Instruction Set Manual
Volume II: Privileged Architecture
Document Version 20211203

Editors: Andrew Waterman[1], Krste Asanović[1,2], John Hauser
[1]SiFive Inc.,
[2]CS Division, EECS Department, University of California, Berkeley
andrew@sifive.com, krste@berkeley.edu, jh.riscv@jhauser.us
December 4, 2021

Non-Executable **RISC-V Specifications**

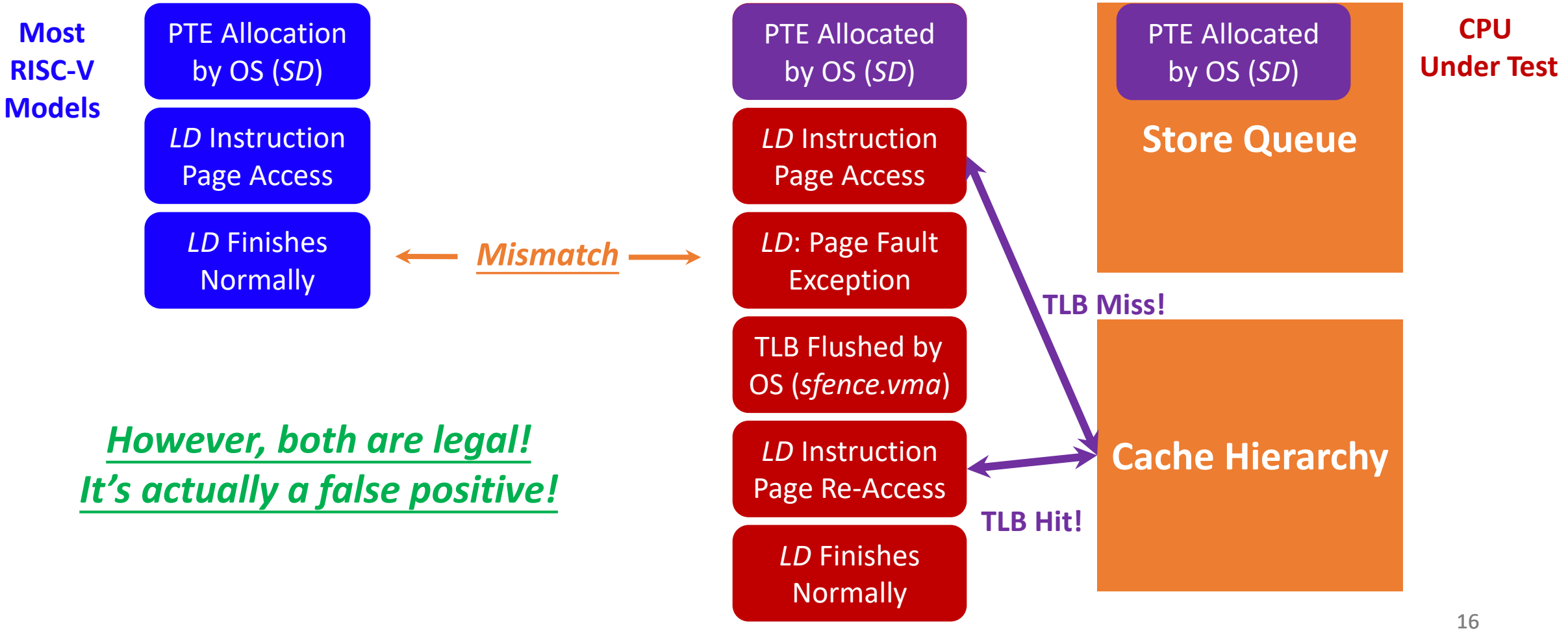
SOTA: Let's *co-simulate them* and *compare the results*!

Processor Co-simulation to Find Bugs



Challenge: False Positives

- **Example: Linux allocates valid PTEs and lazily executes memory-barrier instructions**
  - To avoid frequent TLB flushes for better performance

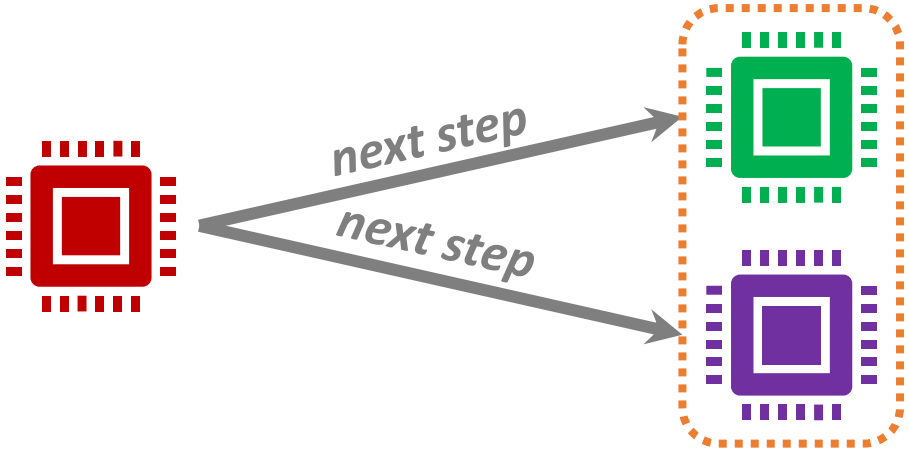


Observation: False Positives and Non-Determinism

- ***Designs are deterministic – as RTL/C++ code is fixed.***

$$R_{P_i}: \quad S_{P_i} \quad \times \quad E \quad \rightarrow \quad S_{P_i} \quad \text{for Processor } P_i$$

- ***Verification should be non-deterministic – as specifications allow diverse designs.***



$$\mathcal{R}: \quad S_{\mathcal{P}} \quad \times \quad E \quad \rightarrow \quad 2^{S_{\mathcal{P}}} \quad \text{for Processors } \mathcal{P} = \{P_i, i \in \mathbb{N}\}$$

This Work: *Diff-Rule based Agile Verification (DRAV)*

Probes: bridge hardware design and verification

Design: $R_{P_i}: S_{P_i} \times E \rightarrow S_{P_i}$ for Processor P_i

$\downarrow f$

Reference: $\mathcal{R}: S_{\mathcal{P}} \times E \rightarrow 2^{S_{\mathcal{P}}}$ for Processors $\mathcal{P} = \{P_i, i \in \mathbb{N}\}$

Diff-Rules: abstract legal behaviors defined in specifications

DRAV: To Check Whether $f\left(R_{P_i}(s_{P_i}, e)\right) \in \mathcal{R}(f(s_{P_i}), e)$

Towards 1-to-*N* Correspondence Between REF and Designs

One REF for One Processor Design

Previously: $R_{P_i}: S_{P_i} \times E \rightarrow S_{P_i}$ **for Processor** P_i

Now: $\mathcal{R}: S_{\mathcal{P}} \times E \rightarrow 2^{S_{\mathcal{P}}}$ **for Processors** $\mathcal{P} = \{P_i, i \in \mathbb{N}\}$

One REF for N Processor Designs

This Work: *DiffTest for RISC-V Processors*

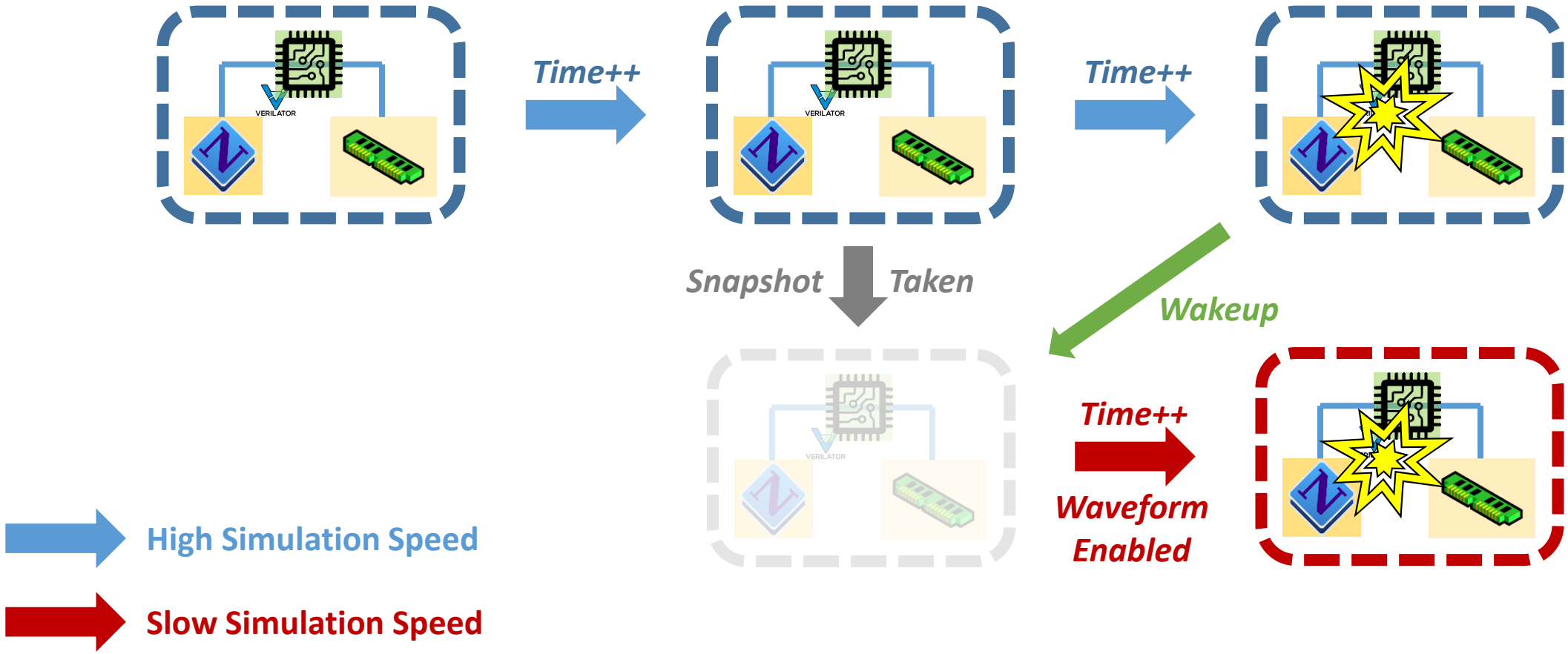
- ***Idea: acknowledge the non-deterministic nature of ISA – the ultimate golden model***
- ***DiffTest*: the state-of-the-art co-simulation framework for RISC-V processors**
  - ***This Work: Identify and Specify Sources of Behavioral Non-Determinism in ISA using Diff-Rules***
  - Dromajo[1]: to avoid non-deterministic sources such as the Debug Transport DTM (MMIO)
  - Imperas[2]: to extract asynchronous information from micro-architecture RTL pipeline

Categories	Sub-Categories	Examples	Addressed Before
Static	Impl. Dependent Registers	CSR, MMIO Devices	[1][2]
Dynamic	Impl. Dependent Registers	Timer, Counters	[2]
	Asynchronous Events	External/Timer Interrupts	[1][2]
	Speculative Execution	Instr./Load/Store Page Fault	***This Work Only***
	Memory Model	Memory Accesses in Multi-core	***This Work Only***
	Hardware Timing	LR/SC, Instruction Fusion, Caches	***This Work Only***

[1] Nursultan et al., Effective Processor Verification with Logic Fuzzer Enhanced Co-simulation. MICRO-54, 2021.
[2] Kevin McDermott. Brief Introduction to the 5 Levels of RISC-V Processor Verification. RISC-V Summit, 2021.

Accelerating Simulation Debugging with Snapshots

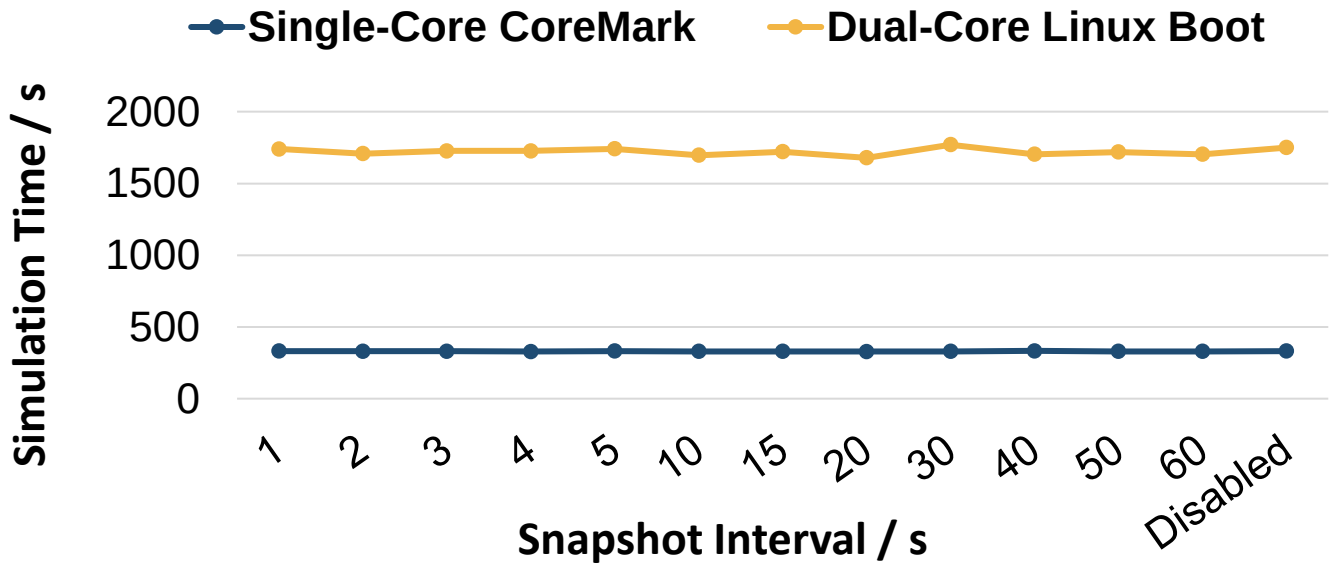
- ***Idea: to re-construct the last cycles of simulation after abort***



This Work: *Light*weight *S*imulation *S*napShot (*LightSSS*)

- ***Challenge***: **Minimize the Performance Overhead of Snapshots**

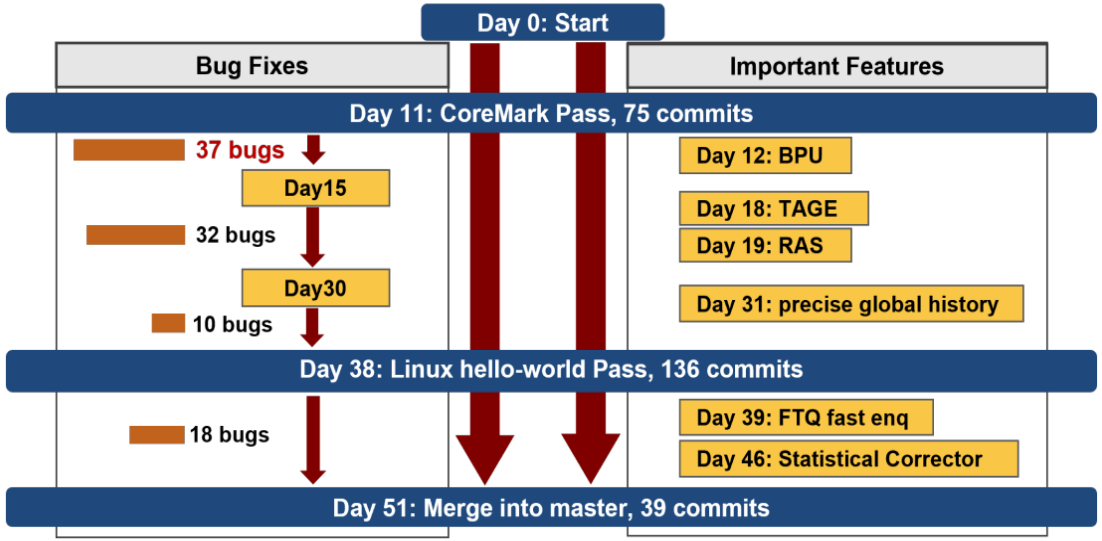
- ***Key Insight: using fork from Linux to take snapshots of the simulation process***
  - **Differential snapshot**: Copy On Write mechanism provided by the OS
  - **Portability**: transparent to the circuits, RTL simulators, external C/C++ models
  - **Efficiency**: in-memory snapshots without disk I/O



Minor performance overhead:
As snapshot interval size increases, simulation speed remains stable

Practice of Agile Development on XiangShan

① Design Optimization



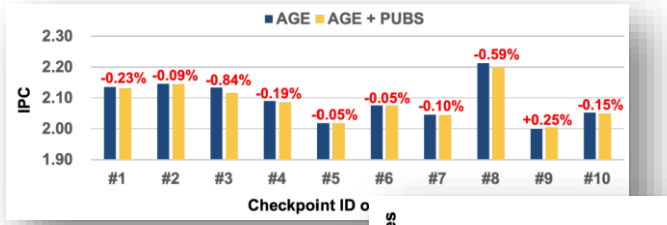
- 3 graduate students
- 11 days for a functionally correct prototype
- 37 bugs in 5 days
- 38 days to boot Linux with BPU
- 51 days for the overall frontend architecture

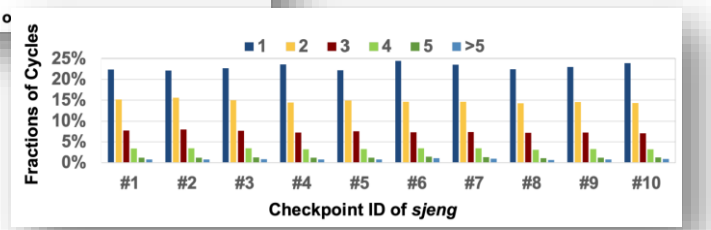
② Paper Reproduction

2018 51st Annual IEEE/ACM International Symposium on Microarchitecture

Performance Improvement by Prioritizing the Issue of the Instructions in Unconfident Branch Slices

One third-year PhD student, 200 minutes on XiangShan





More details in our paper

Conclusion

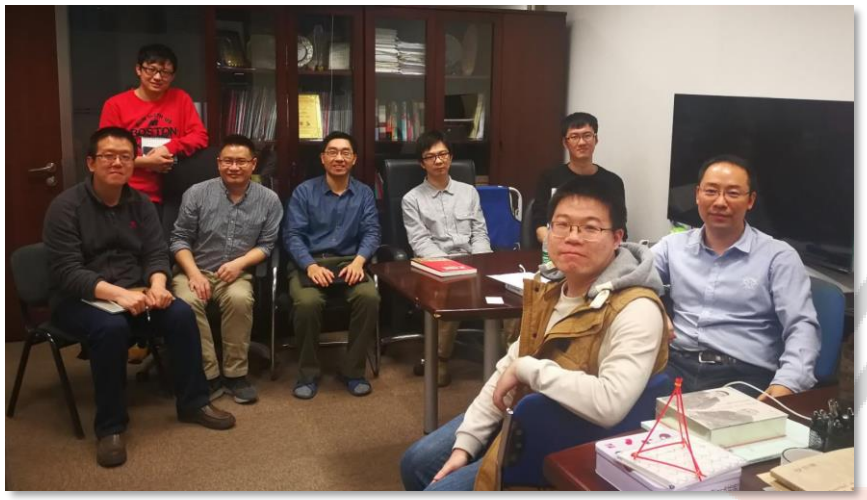
- ***XiangShan*: High Performance RISC-V Processor**

- ***MinJie*: Platform with Agile Development Flows and Tools**
  - DRAV and DiffTest: Functional Verification
  - LightSSS: Debugging

- **Both XiangShan and MinJie are *Open-Source***
  - Follow us at github/OpenXiangShan *with 3K stars ♥ from the lovely community*
  - Artifacts Available, Functional, and Reproduced

Thanks!

Open-Source Chip Working Group



中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

中国科学院大学
University of Chinese Academy of Sciences

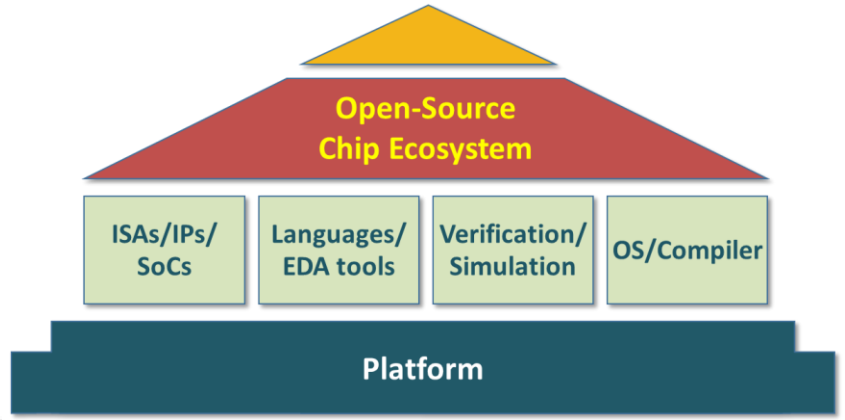
北京开源芯片研究院
BEIJING INSTITUTE OF OPEN SOURCE CHIP

PENGCHENG LABORATORY

vcore

SHENZHEN UNIVERSITY

DALIAN UNIVERSITY OF TECHNOLOGY



Together for a Shared Future

To Lower the Barrier of Chip Development

XiangShan Team

Open-Source Collaborators