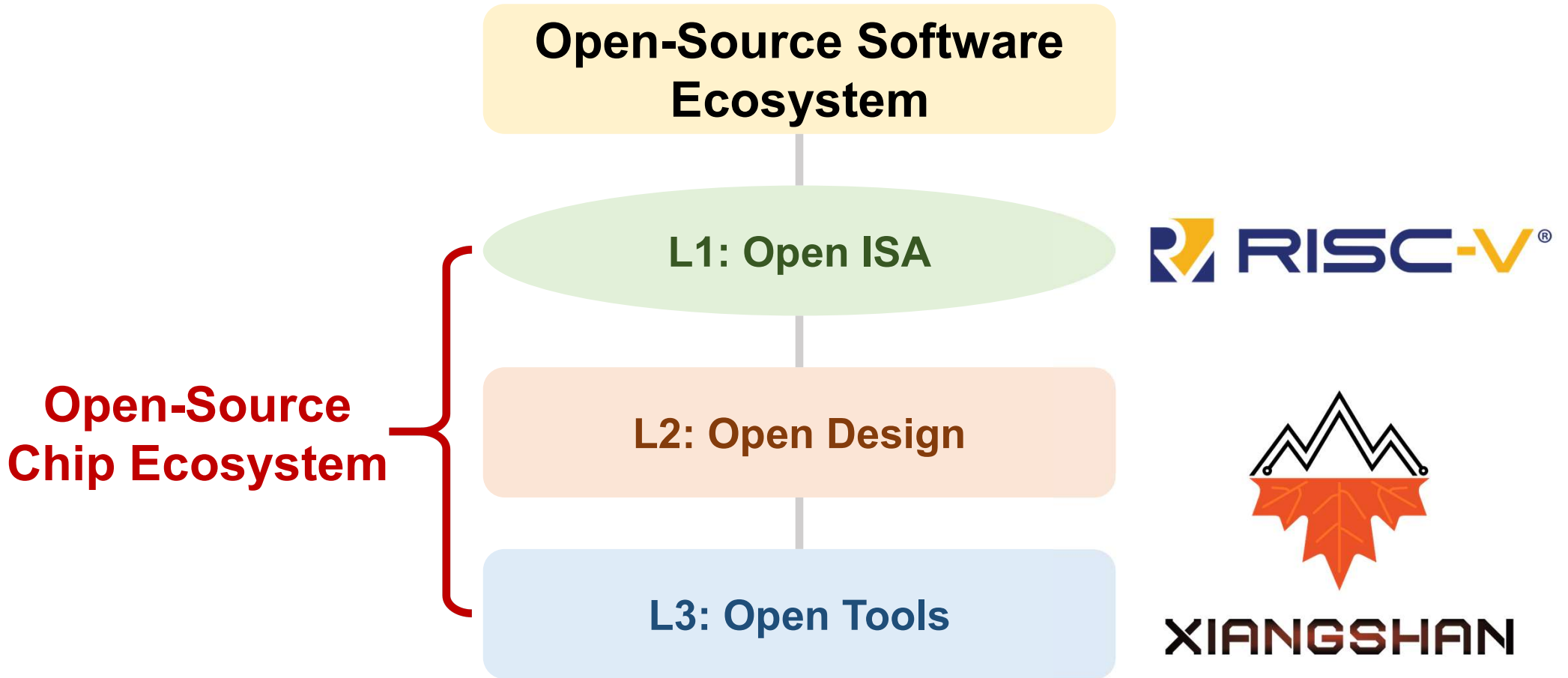


Open-Source Tools and Open Problems in Agile Chip Development Infrastructure

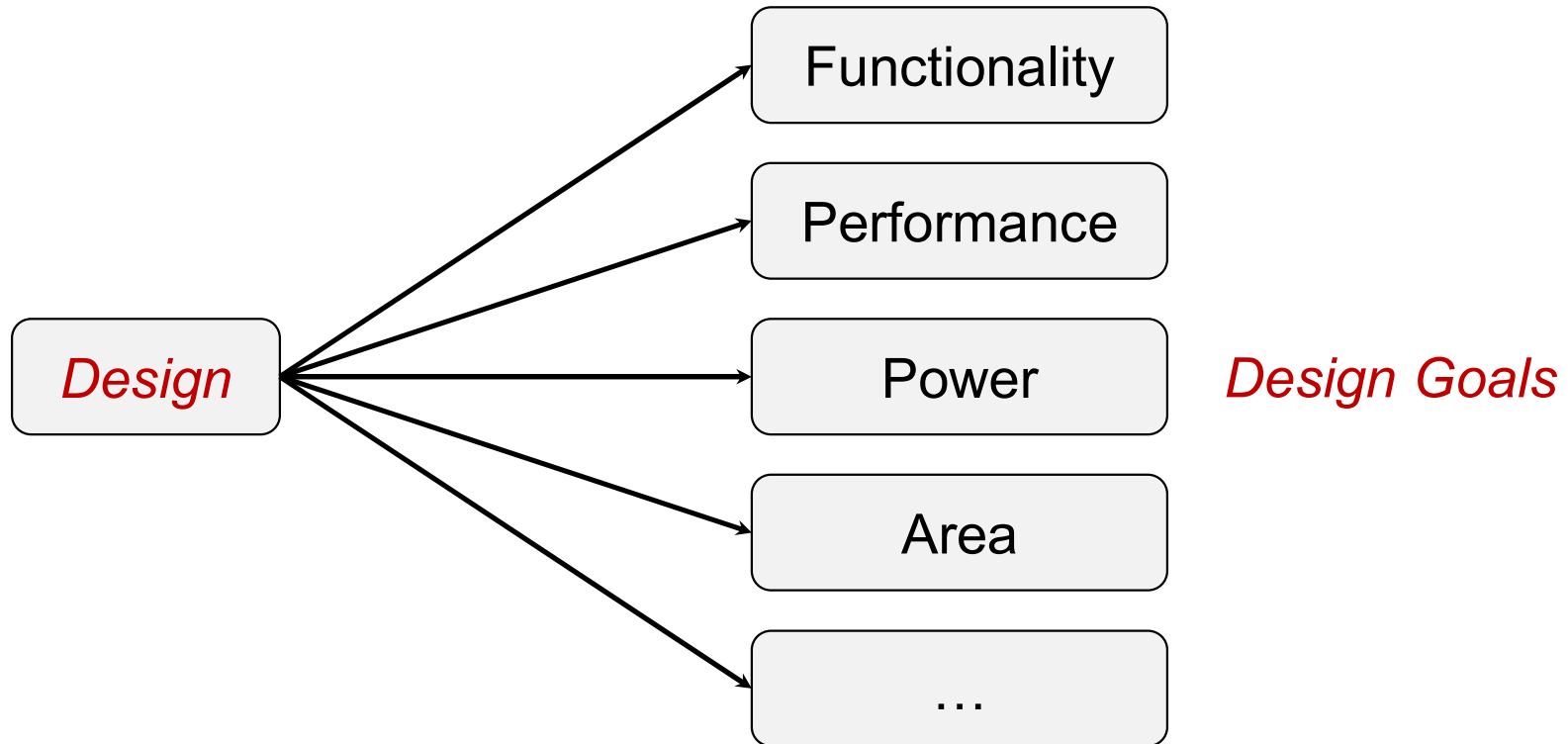
The XiangShan Team
HPCA 26 @ Sydney, Australia
January 31, 2026

The XiangShan Project



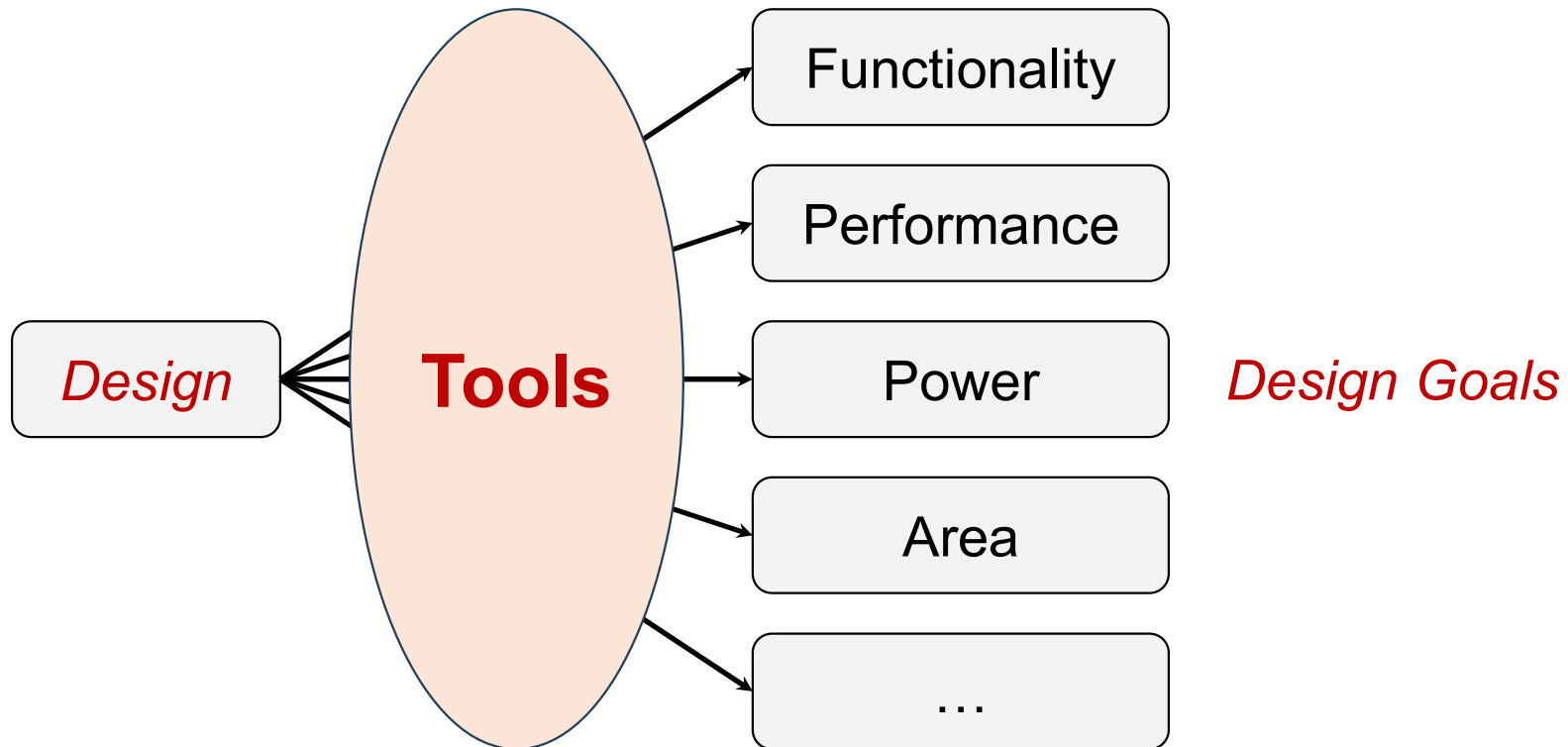


Why Infrastructure and Tools: Design Goals

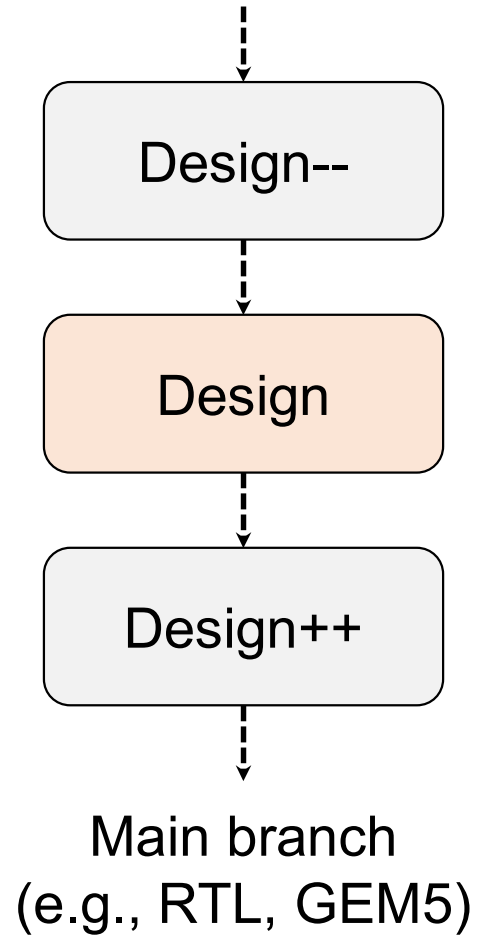


Why Infrastructure and Tools

- *To improve development efficiency, productivity, quality, ...*

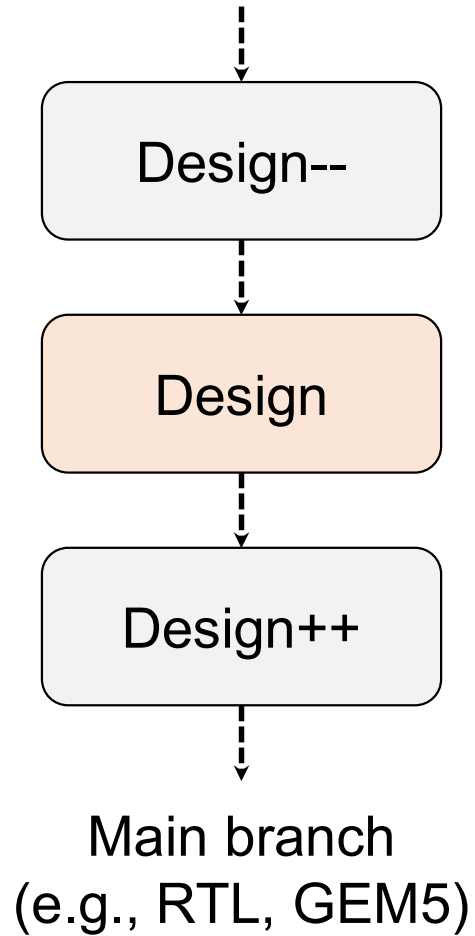


Chip Development Workflow





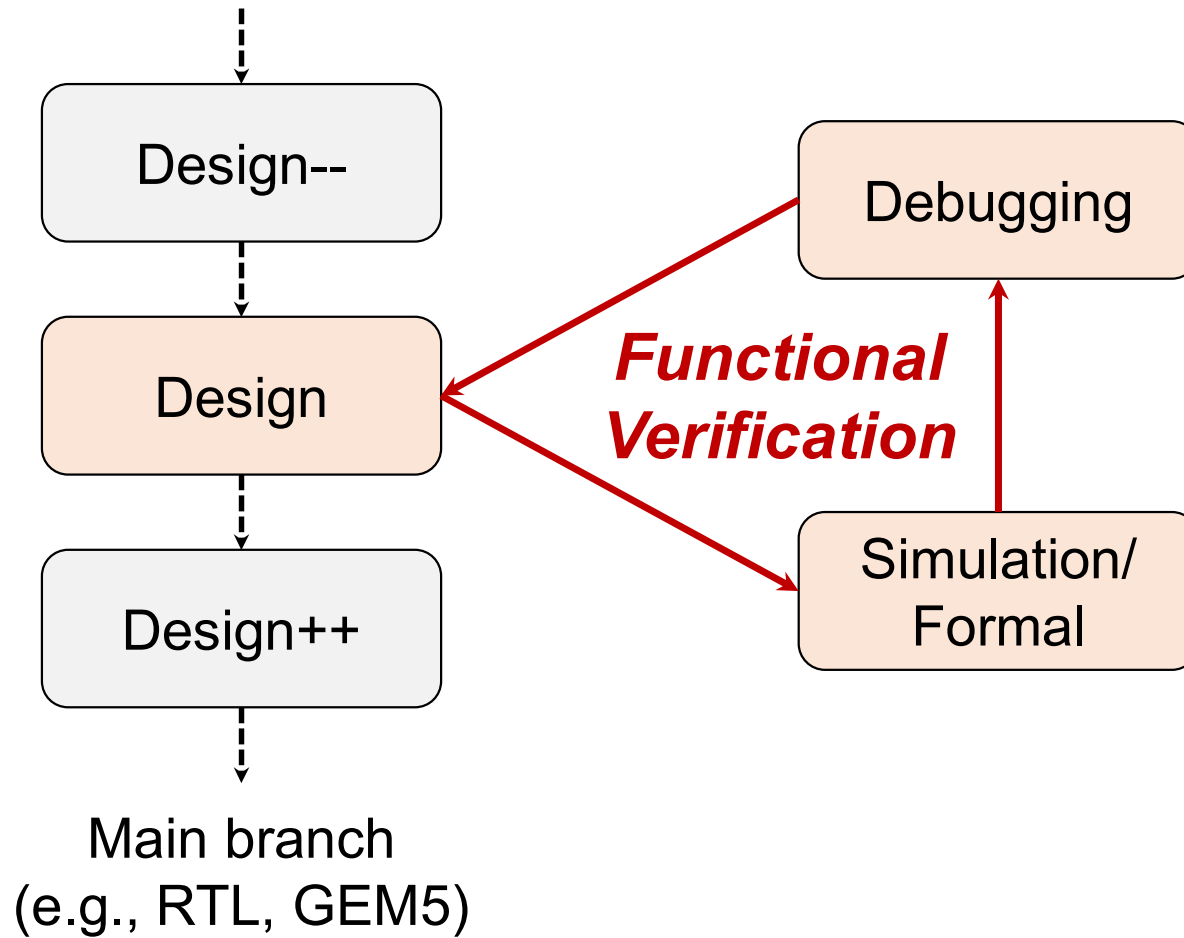
Chip Development Workflow: Description



Description Languages
(e.g., Chisel, Verilog, C++)

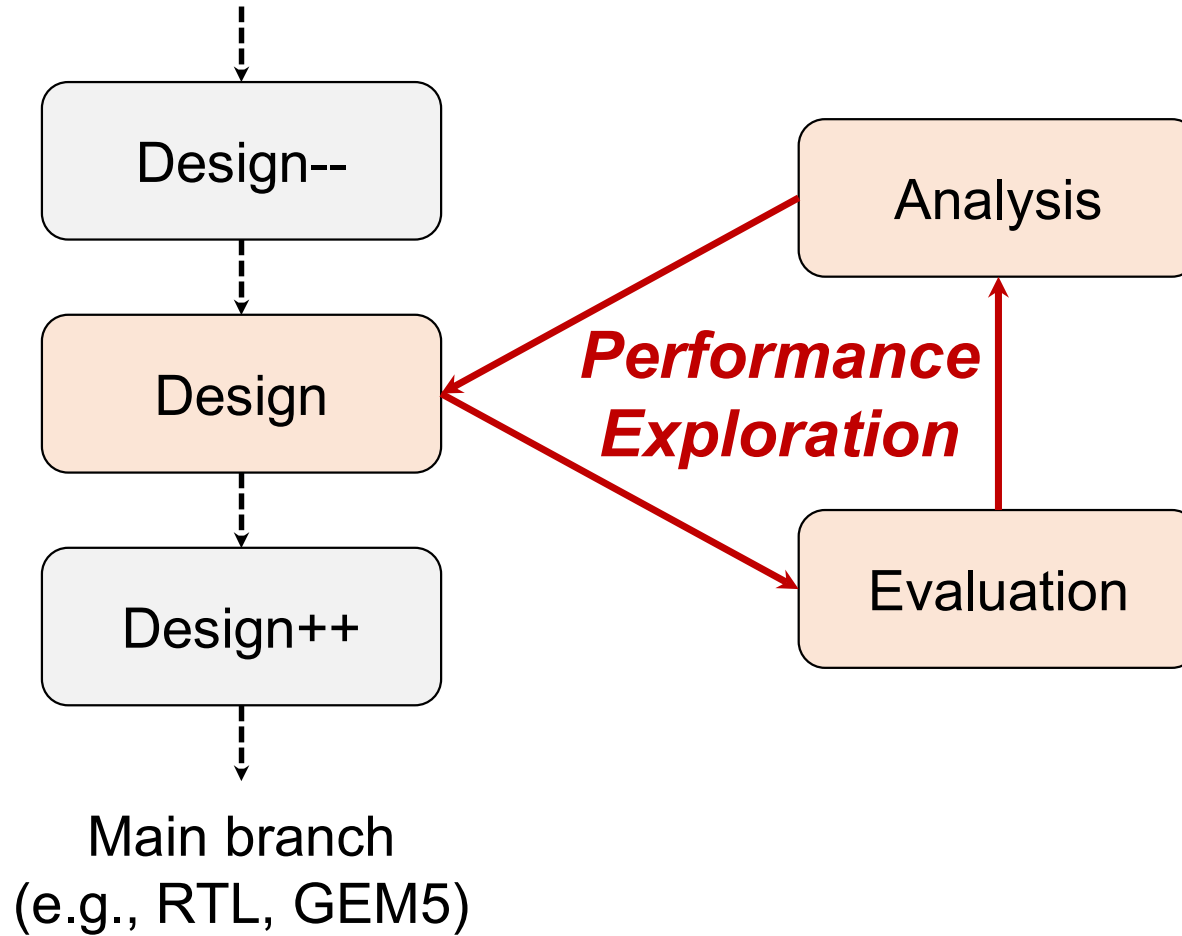


Chip Development Workflow: Functionality





Chip Development Workflow: Performance





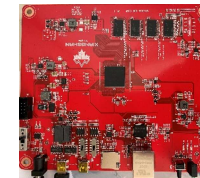
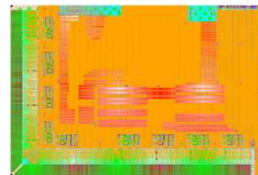
Chip Development Workflow and Problems

- **Design description languages**
 - Hardware description languages
 - High-level architectural simulators
- **Functional Verification**
 - RTL-simulation
 - Test generation
- **Performance Exploration**
 - Evaluation methodologies (simulation, sampling, ...)
 - Analysis methodologies

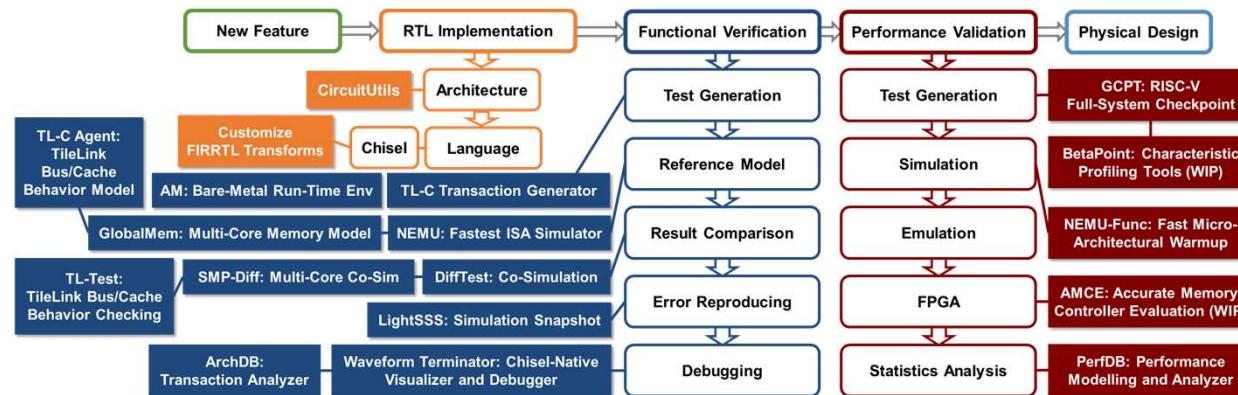


MinJie: Open & Agile Development Toolchain

- Infrastructure is the key outcome of the XiangShan Project
- Open source to benefit both academia and industry



XiangShan



MinJie

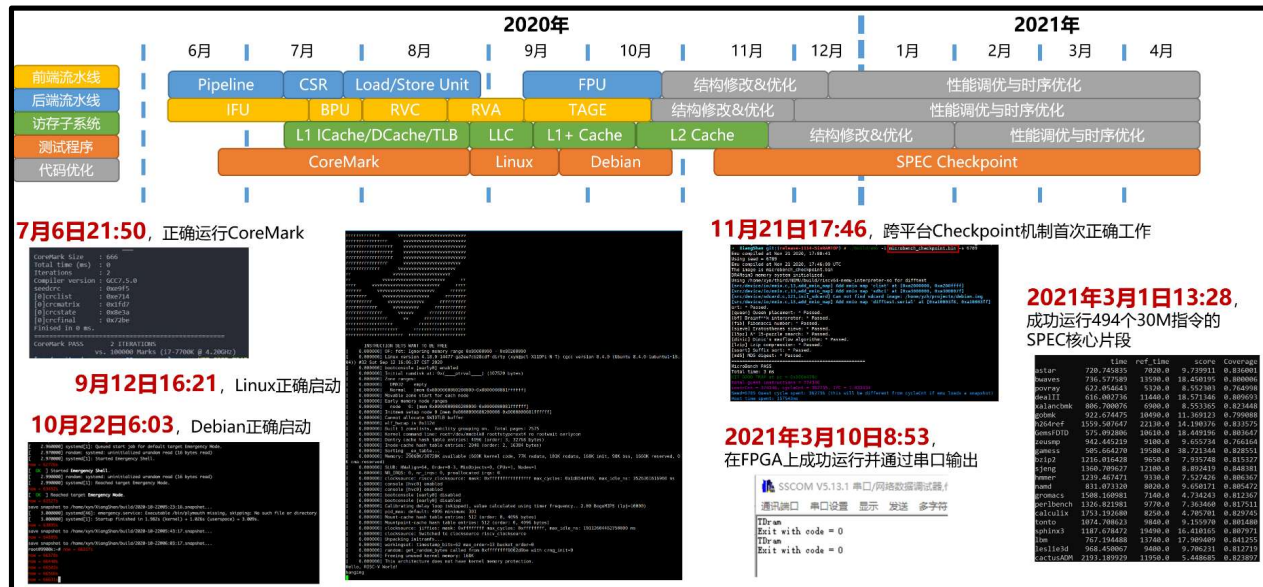
Chip Development Tools

- **Design description languages**
 - Hardware description languages – **Use Chisel**
 - High-level architectural simulators – **XS-GEM5 (calibrated GEM5 for XS)**
- **Functional Verification**
 - RTL-simulation
 - Test generation
- **Performance Exploration**
 - Evaluation methodologies (simulation, sampling, ...)
 - Analysis methodologies



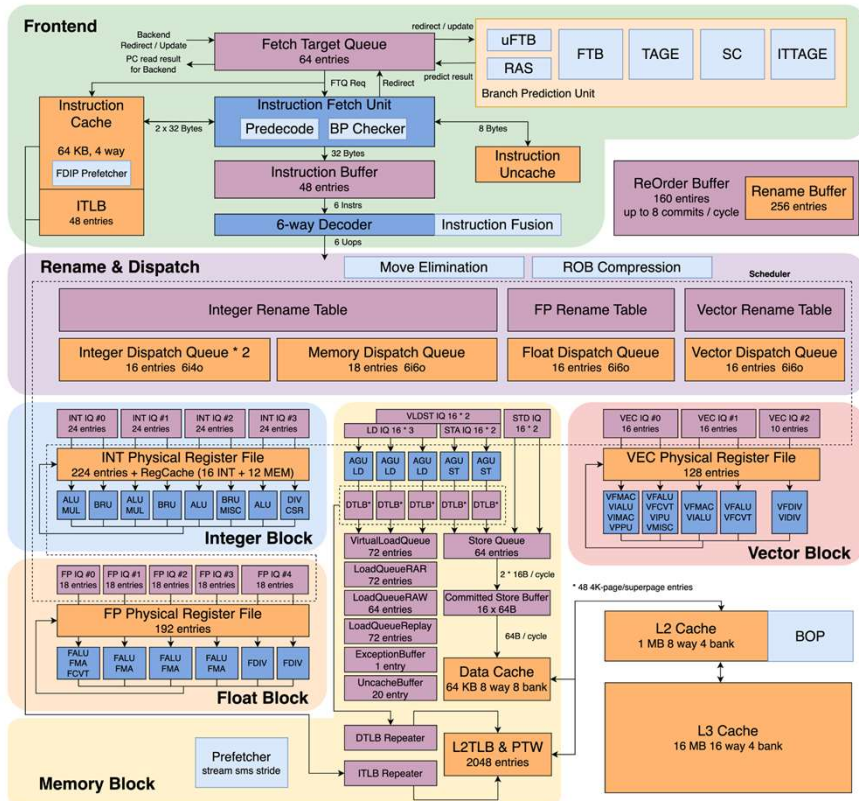
We Use Chisel for Hardware Description

- 2018: quantitatively comparing Chisel and Verilog, reducing code size by 80%
- 2020: completed the 1st gen of XiangShan, booting Linux within 3 months
- 2022: 67,000 lines of design code and 31,000 lines of verification code
- 2024: 214,000 lines of code in all code repositories

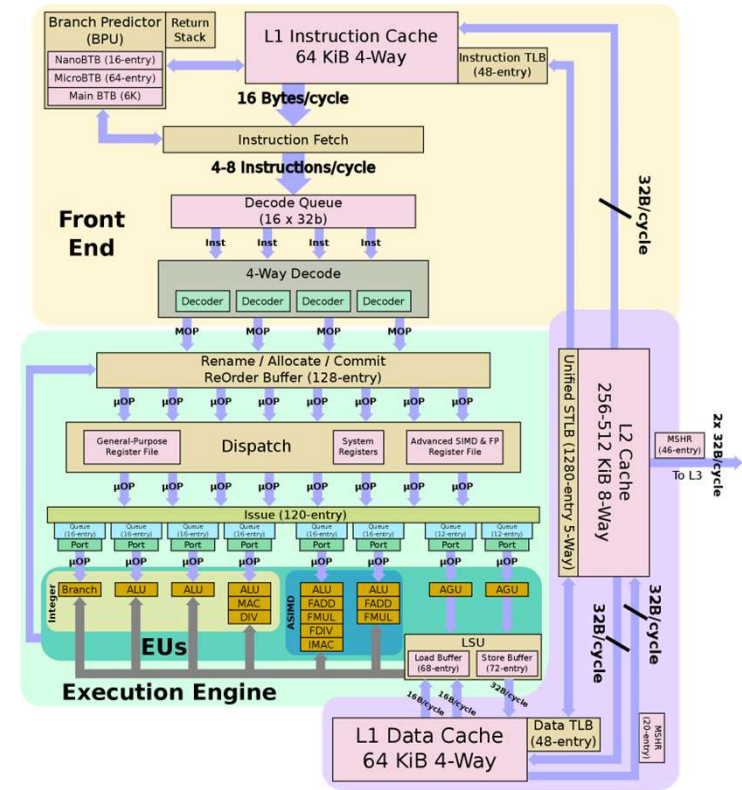


Better Design Flexibilities

Kunminghu has **212** configurable parameters, and its L2/L3 cache has **65** configurable parameters



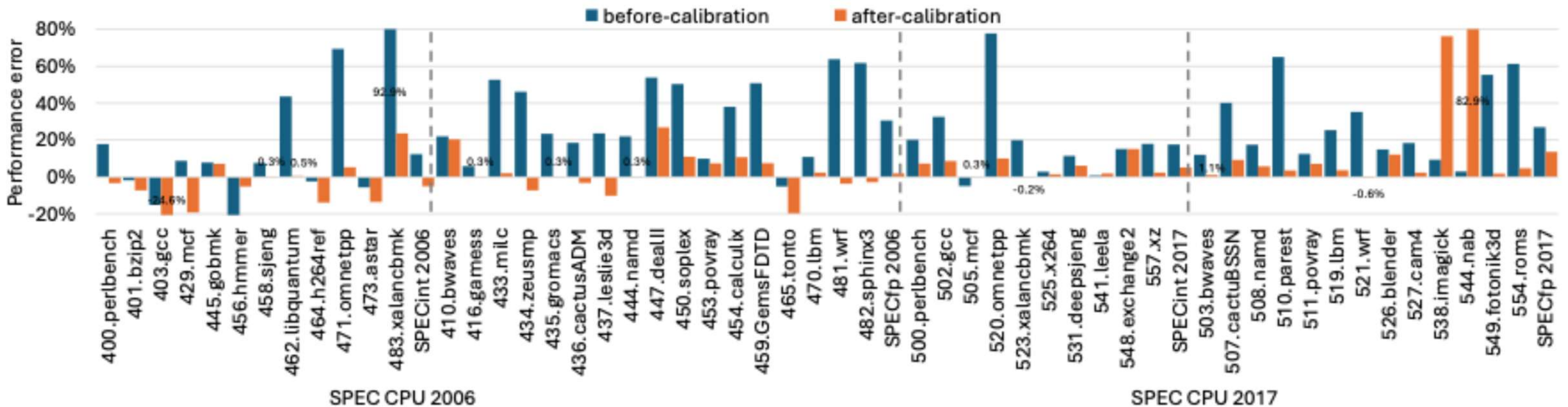
ARM A76 has **8** configurable parameters, and the DSU (L3) has **25** configurable parameters.





XS-GEM5: Calibrated GEM5 Simulator for XiangShan

- **<3% SPEC06 performance error** against the RTL (KMH-v2)
- Currently being used for architecture exploration in KMH-v3
- Will be introduced in the next session



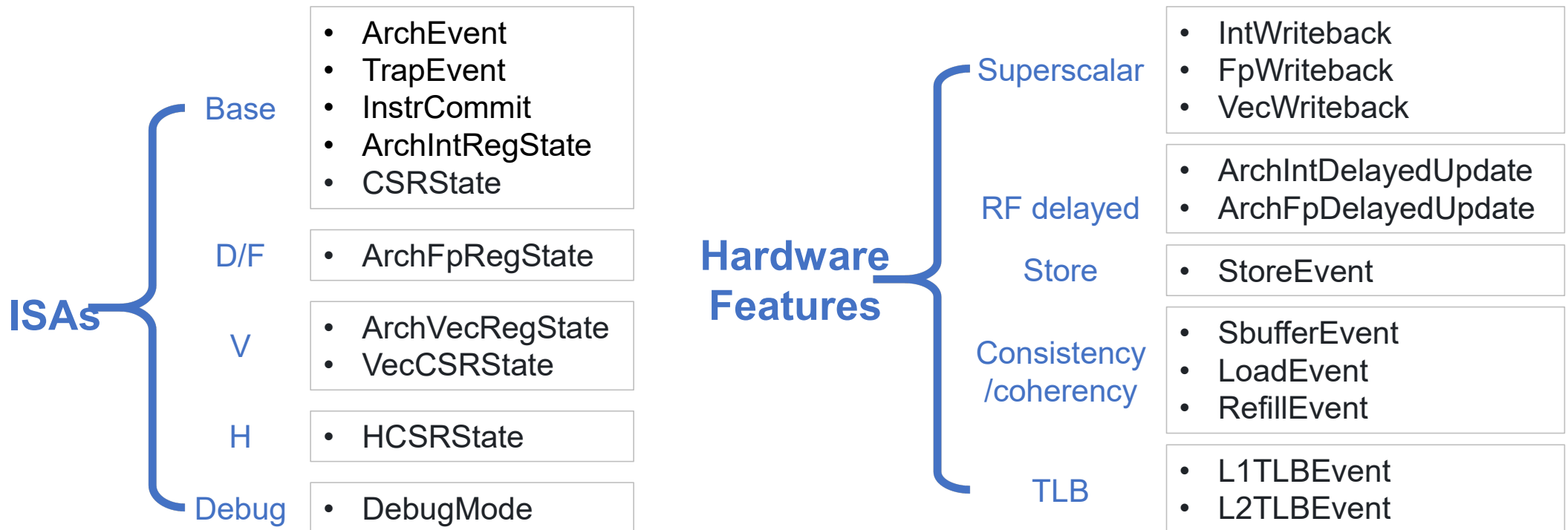
Chip Development Tools

- **Design description languages**
 - Hardware description languages – **Use Chisel**
 - High-level architectural simulators – **XS-GEM5 (calibrated GEM5 for XS)**
- **Functional Verification**
 - RTL-simulation – **DiffTest**
 - Test generation – **xfuzz**
- **Performance Exploration**
 - Evaluation methodologies (simulation, sampling, ...)
 - Analysis methodologies



Standard Interfaces for RISC-V CPU Verification

- Key idea: information probes support flexible combination for different scenarios





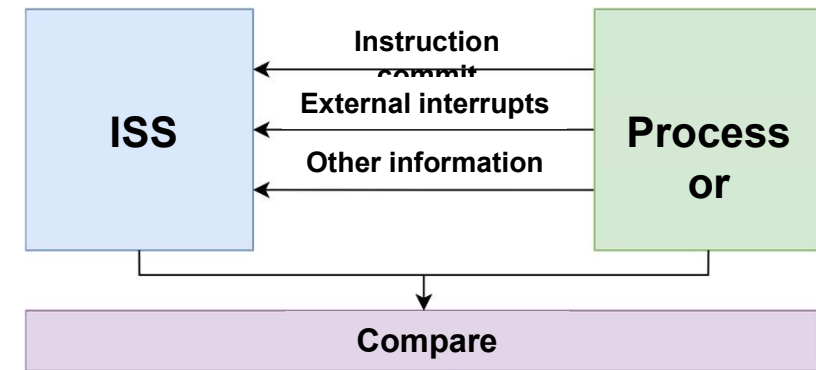
DiffTest: a Co-simulation Verification Framework

• Co-simulation workflow

- Instructions commit/other states update
- The simulator executes the same instructions
- Compare the architectural states
- Abort or continue

• Verification infrastructures for CPUs

- APIs for HDLs such as Chisel/Verilog
- RTL simulators (Verilator/VCS, Palladium, FPGAs)
- RISC-V ISS such as Spike, NEMU
- Published at MICRO'22



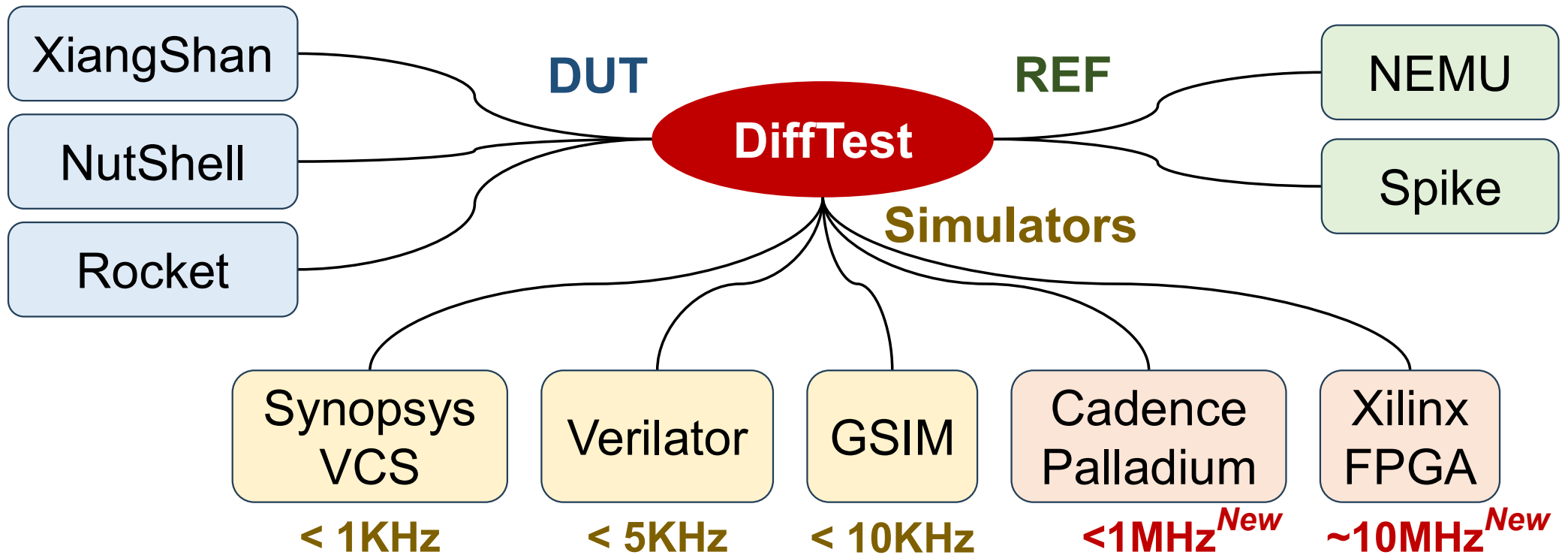
Basic architecture

```
while (1) {  
    icnt = cpu_step();  
    ref_step(icnt);  
    r1s = cpu_getregs();  
    r2s = ref_getregs();  
    if (r1s != r2s) { abort(); }  
}
```

Online checking

Accelerated Co-simulation on Emulator/FPGA

- DiffTest now supports hardware-accelerated co-simulation



DiffTest-H: Semantic-aware Co-sim Acceleration

- Optimized communication overhead for verification data packets
- **13.8 MHz** on FPGA, **instruction-level** debugging
- Deployed on **XiangShan**, with **151 bugs uncovered**

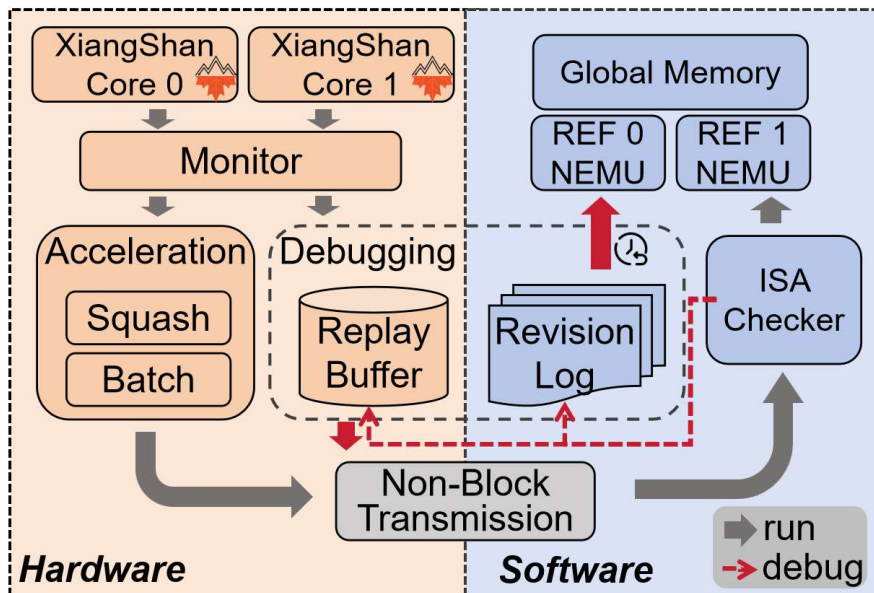


Figure: workflow

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

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

香山开源处理器社区
XiangShan Open-Source Processor Community

DiffTest-H: Toward Semantic-Aware Communication in Hardware-Accelerated Processor Verification

Kunlin You^{1,2}, Yinan Xu¹, Kehan Feng³, Luoshan Cai^{1,2}, Yaoyang Zhou³ and Yungang Bao^{1,2}

¹ State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences
² University of Chinese Academy of Sciences
³ Beijing Institute of Open Source Chip

Published at MICRO'25

Test Inputs for CPUs

- Test cases we are *currently* using for the system-level DV of CPUs
 - Modern CPU DV reaches a good coverage

1) hand-written directed tests

- riscv-software-src/riscv-tests
- riscv-non-isa/riscv-arch-test
- riscv-ovpsim/imperas-riscv-tests
- litmus-tests/litmus-tests-riscv
- josecm/riscv-hyp-tests

2) instruction-stream generators

- chipsalliance/riscv-dv
- openhwgroup/force-riscv
- ksco/riscv-vector-tests
- sifive/riscv-vector-intrinsic-fuzzing
- chad-q/andes-vector-riscv-dv

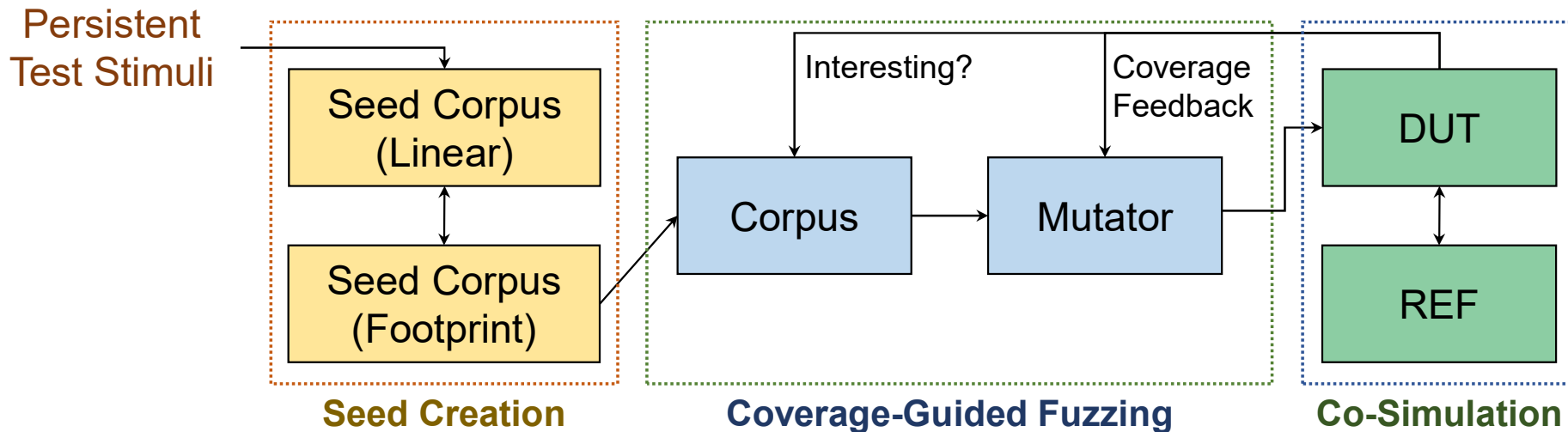
3) real-world programs

- ucb-bar/riscv-benchmarks
- eembc/coremark
- SPEC CPU® 2017
- SPECjbb® 2015
- gcc,clang,rustc,verilator

- Let fuzzers take a step further!



xfuzz: Fuzzing CPUs with Coverage Guidance

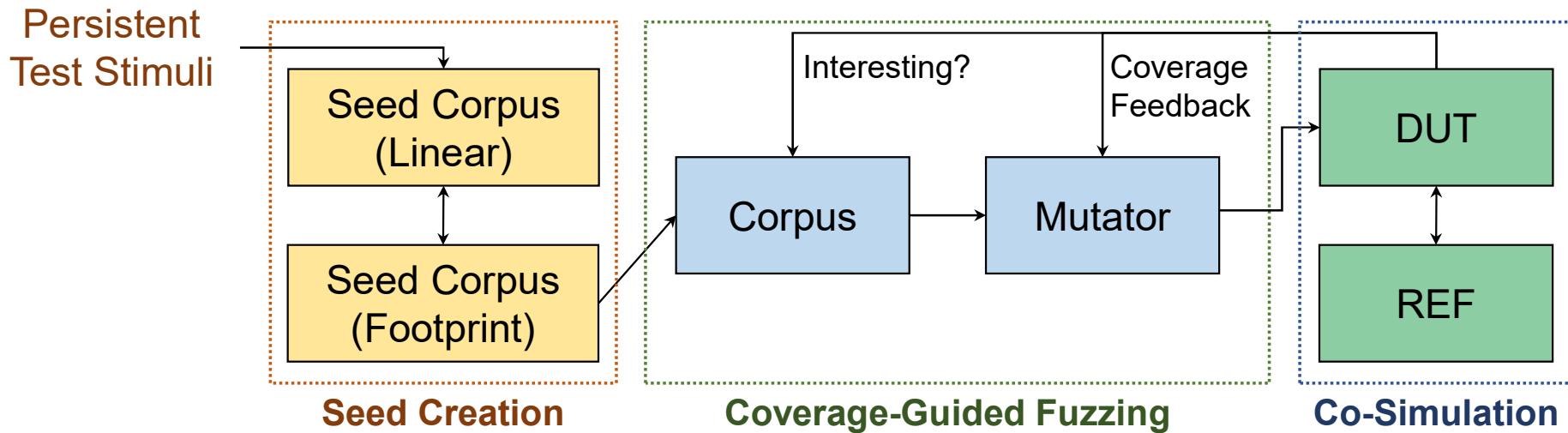


Coverage-Guided Mutational Fuzzing

- Starting with **seed corpus** as the search base
- Exploring the search state space using **mutations**
- Guiding the exploration based on **coverage metrics**
- Enhanced search efficiency with **Footprint Memory**



Fuzzing and Verifying Your RISC-V CPUs



**Any
Testcases**

XFUZZ
[OpenXiangShan/xfuzz](https://github.com/OpenXiangShan/xfuzz)

DiffTest
[OpenXiangShan/difftest](https://github.com/OpenXiangShan/difftest)

Available on GitHub with off-the-shelf examples on XiangShan, Rocket Chip, and NutShell

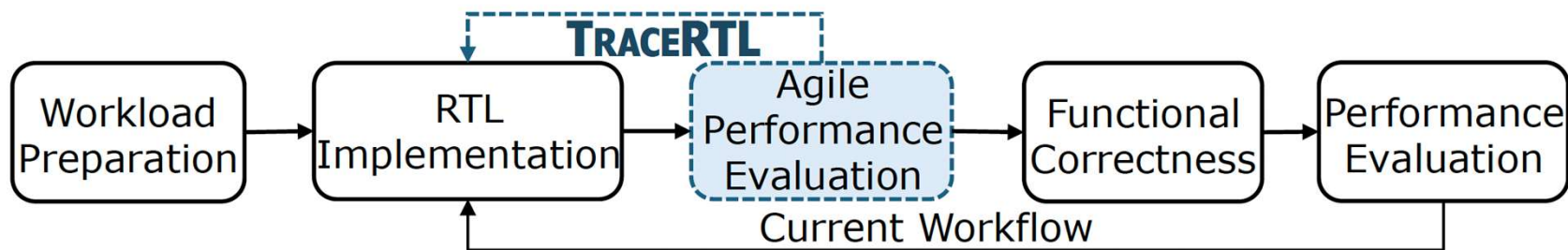
Chip Development Tools

- **Design description languages**
 - Hardware description languages – **Use Chisel**
 - High-level architectural simulators – **XS-GEM5 (calibrated GEM5 for XS)**
- **Functional Verification**
 - RTL-simulation – **DiffTest**
 - Test generation – **xfuzz**
- **Performance Exploration**
 - Evaluation methodologies (simulation, sampling, ...) – **TraceRTL**
 - Analysis methodologies – **Top-Down, TIP, TEA implementations**



How RTL CPU is Simulated: Execution-driven

- Given an ELF/binary, the CPU fetches, executes and branches
 - The assumption: performance runs *after* functional correctness

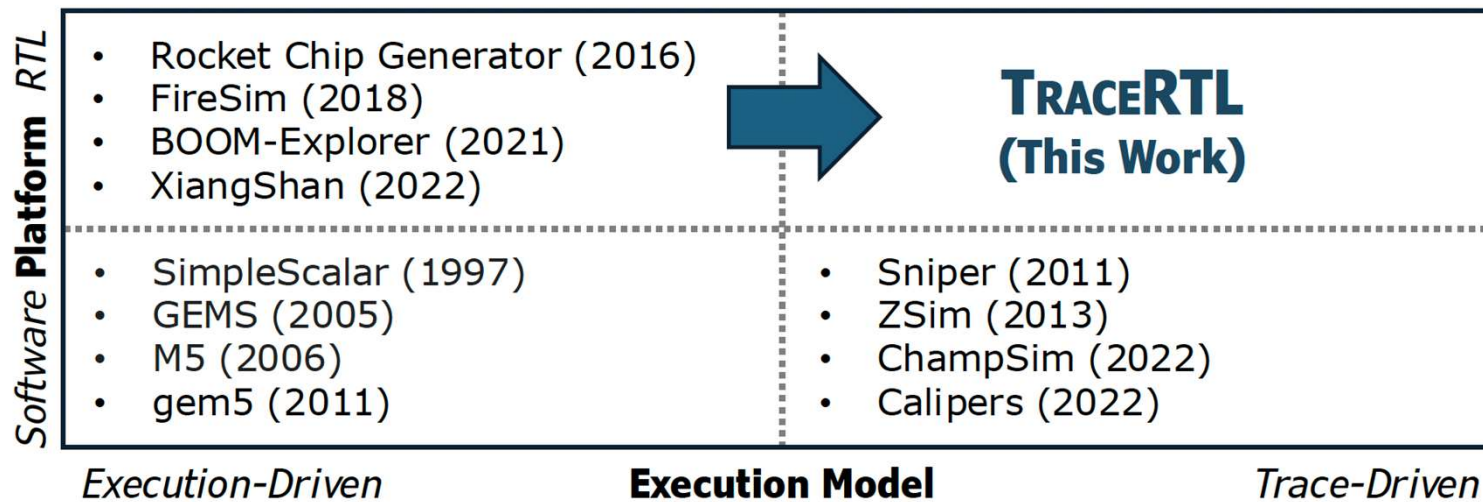


- **Is it possible to evaluate performance *before* functionalities?**



TraceRTL: Trace-driven RTL CPU Model

- However, architecture exploration only requires essential performance components
- Insight: applying trace-driven simulation from simulators to RTL
 - Therefore, performance evaluation can be performed before ensuring CPU functional correctness

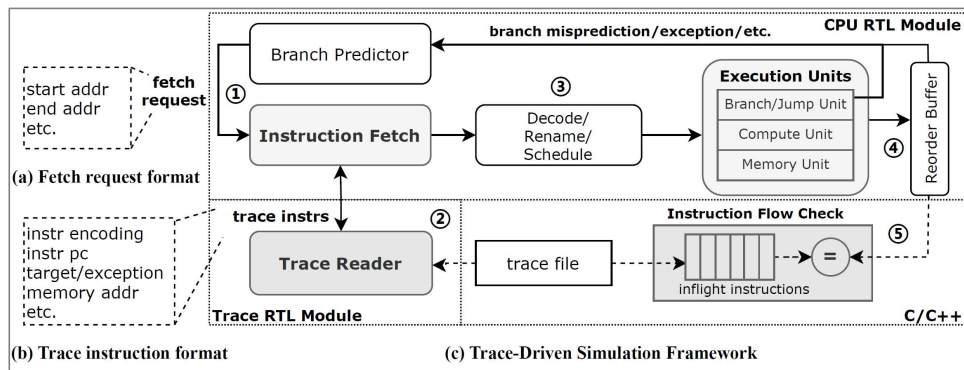




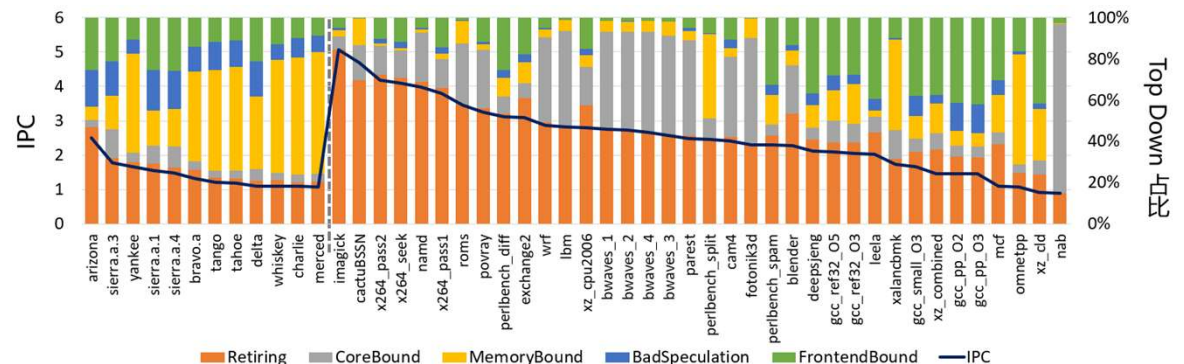
TraceRTL: Trace-driven RTL CPU Model

HPCA'26
Tuesday 14:50
@Coogee

- Low-intrusion modification of **a trace-driven XiangShan**
 - **Eliminating functional dependencies:** Chisel, circuitry, architecture, functional abstraction, performance sensitivity, etc.
 - **Eliminating performance errors:** The impact of lost information in the trace, including addresses, data, etc.

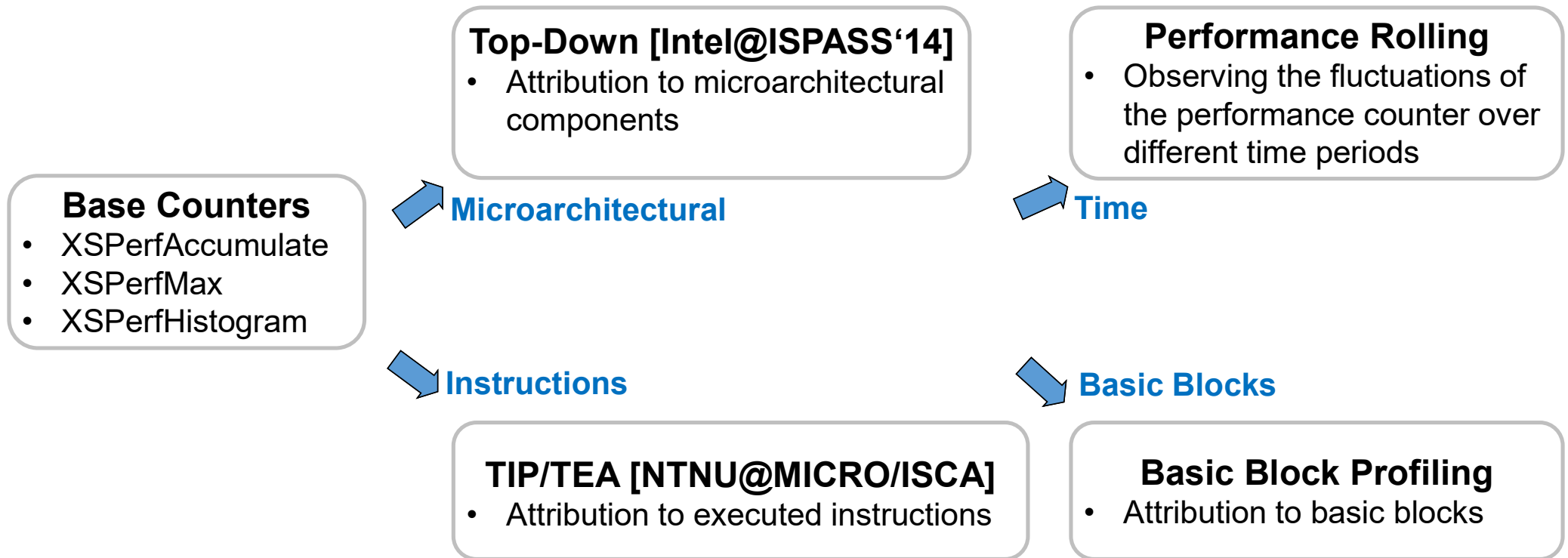


Trace-driven CPU from XiangShan



Top-Down differences between Google Datacenter Workloads and SPEC CPU

Performance Counters





Open Problems: Challenges and Opportunities

- **Design description languages**
 - Hardware description languages
 - High-level architectural simulators
- **Functional Verification**
 - RTL-simulation
 - Test generation
- **Performance Exploration**
 - Evaluation methodologies (simulation, sampling, ...)
 - Analysis methodologies



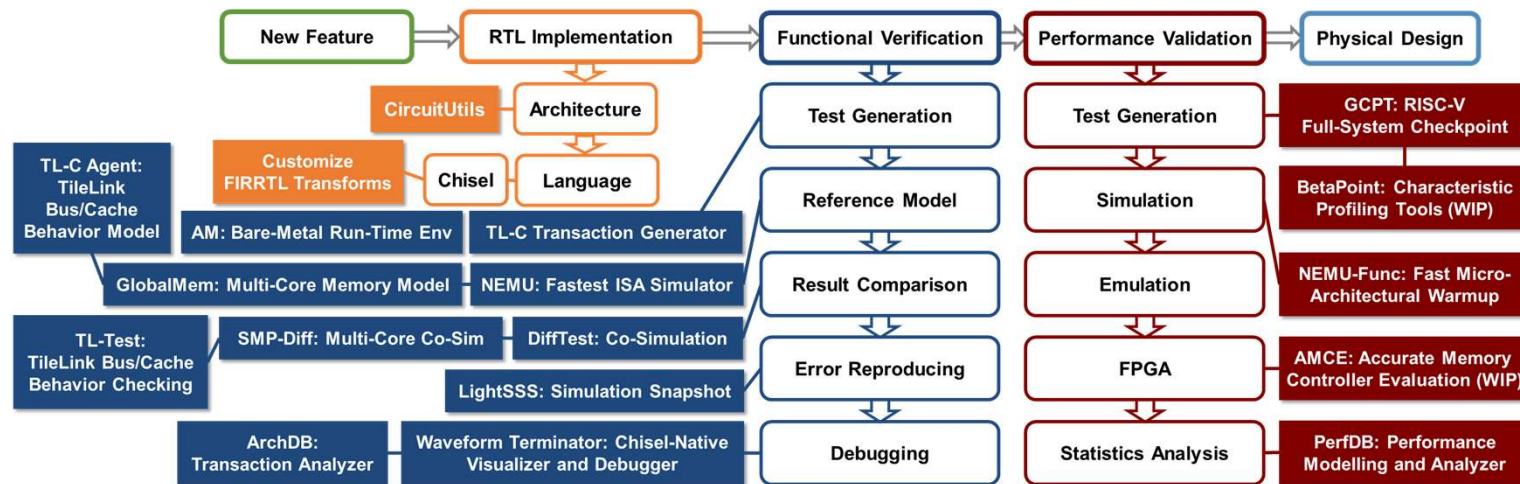
OpenChipDevInfraProblems-XiangShan

<https://docs.google.com/spreadsheets/d/1zStBR8Q9a8H5eWlbs0R4aMKIPWd5HlSV6bDNnXPe3xU/edit?usp=sharing>



Conclusions: Open Tools and Problems

- Infrastructure is the key outcome of the XiangShan Project
- Open source to benefit both academia and industry
- Next Session: XS-GEM5



MinJie: more in the detailed hands-on session!