Impact of RISC-V Adaptability on SoC Verification Methods

S. Davidmann, L. Moore, L. Lapides

Imperas Software Ltd.

25-Feb-20

Agenda

- RISC-V verification issues
- Compliance is not verification
- Reference models and custom instructions
- Processor IP verification
- SoC level verification
- Summary

Agenda

- **RISC-V verification issues**
- Compliance is not verification
- Reference models and custom instructions
- Processor IP verification
- SoC level verification
- Summary

RISC-V Presents New Challenges

- RISC-V is a new ISA – an open standard ISA
  - Managed by the non-profit RISC-V Foundation (riscv.org)
  - This means any designer can build a processor implementation
    - (Feb 2020 – there are almost 100 RTL designs including open source and proprietary)
- Traditionally, processor IP …
  - comes from, and is maintained by, the ISA owner
  - is single sourced
  - comes fully verified and compliant to that specific ISA
- All users need to do is to verify using integration tests
- There is no "standard" approach and there are few available tools for processor verification

- The RISC-V industry / eco-system needs to adapt best practices for SoC verification to processor verification

To be more specific about the RISC-V DV Problem

- Arm processor IP
  - ~ 10^{15} verification cycles per processor
  - Verification of interface between NoC and processor
  - 1,000s of SoC designs successfully produced
- Similar stories for ARC, MIPS, Tensilica, …

- RISC-V IP
  - How well verified is an individual processor (from processor IP vendor, open source, self-built)?
  - How to verify processor subsystems, especially for AI/ML architectures?
  - How well verified is interface between NoC and processor?
  - How to deal with custom instructions?

Agenda

- RISC-V verification issues
- **Compliance is not verification**
- Reference models and custom instructions
- Processor IP verification
- SoC level verification
- Summary

Compliance Testing

- The device works within the envelope of the agreed specifications
  - Have you read and understood the specification
- Testing of the instructions should
  - Attempt to use all registers as source and destination (not combinations)
  - Attempt to operate on all bits which compose the immediate values (1 / 0)
  - Capture a signature in memory region indicating the test result
    - Based upon a particular hardware configuration
  - Compare the signature against a known good reference
    - Static (pre defined signature extraction)
    - Dynamic (runtime generation from YAML configured reference)

Compliance Testing (2)

- Testing of the instructions should NOT
  - Attempt to stress all possible aspects of functional verification, eg
    - All possible combinations of instruction parameters (2-in, 1-out = 32,768)
    - All possible data values
  - Attempt to expose possible micro-architectural aspects
  - Attempt to exercise behaviour which generates an exception
    - Illegal instructions (unsupported extensions)
      - (*) Do not test for missing M instructions in context of RV32I
    - Illegal conditions (misaligned fetch, load, store)

Compliance Testing (3)

- Test Qualification
  - Functional Coverage analysis
  - Mutation Fault Simulation - Testing analysis (Imperas work in progress)
    - Provides Decode Coverage
      - Sees if observe changes on all bits of legal decodes

Compliance Testing (4)

- Test Qualification
  - Functional Coverage analysis
  - Mutation Fault Simulation - Testing analysis (Imperas work in progress)
    - Provides Decode Coverage
      - Sees if observe changes on all bits of legal decodes
    - Verified against RV32I test suite
      - 48 hand coded directed tests (average 150 instructions each)
      - https://github.com/riscv/riscv-compliance/tree/master/riscv-test-suite/rv32i/src
    - Decode Coverage data from the Imperas tool
      - ran 478,390 simulations in 308 secs

Compliance Testing (5)

- Compliance RV32I Base Instruction Testing
  - November/12/2019 – 48 tests
- Compliance RV64V Vector instruction Testing (Imperas work in progress)
  - February/2020 – ~6,000 tests
- RISCV-V compliance suites are still a work in progress

Agenda

- RISC-V verification issues
- Compliance is not verification
- **Reference models and custom instructions**
- Processor IP verification
- SoC level verification
- Summary

Key Issue – Reference Comparison

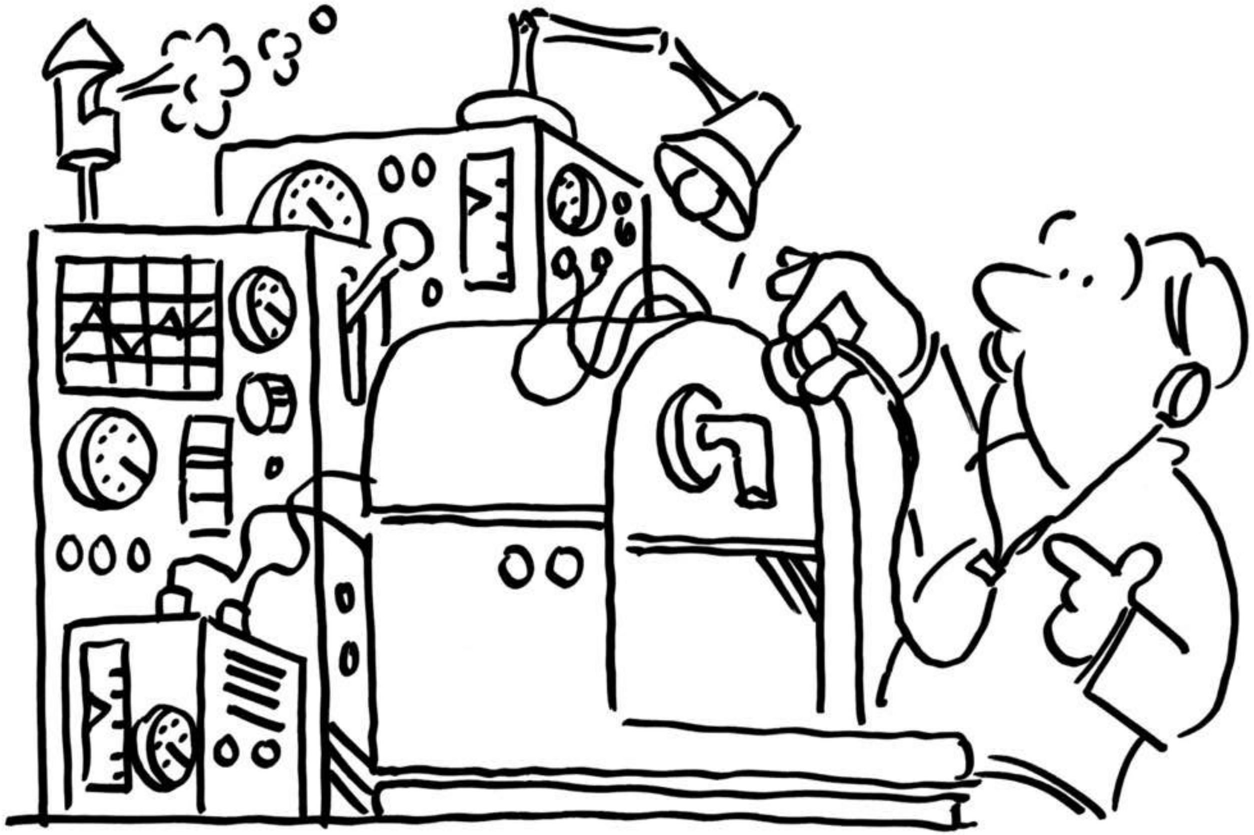
- One thing compliance, directed, random have in common...
  - Is a need for a reference implementation to compare with

- So why do I need a reference as part of my verification ?
  - Comparison for the observed behavior
  - Covering all possible aspects of the ISA envelope

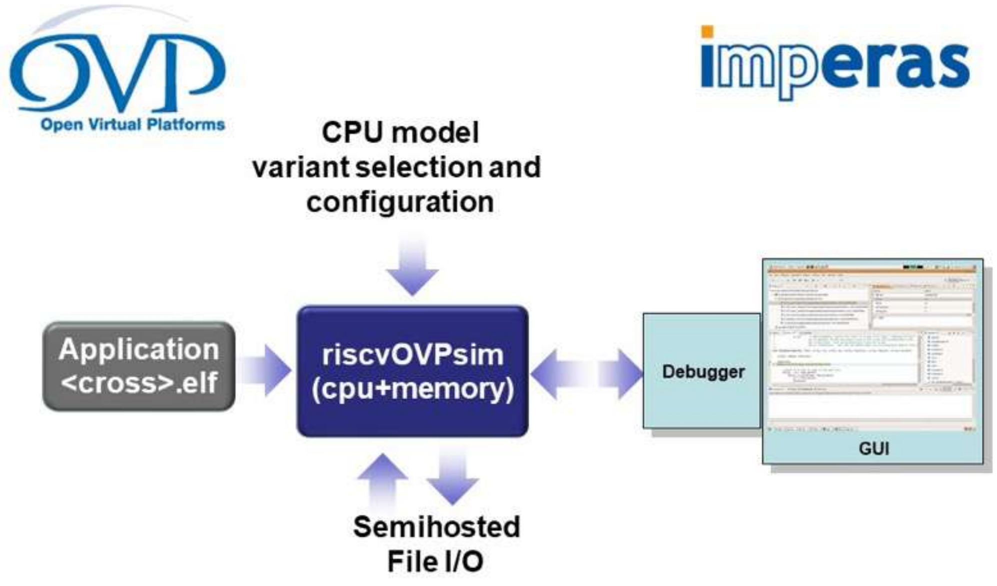
- And – it needs to represent your exact design and architecture:
  - XLEN
  - Vectors: VLEN, SLEN, ELEN, (version: 0.7.1, 0.8, 0.9 Draft, …)
  - Bit Manipulation (version: 0.9, 0.91, 0.92, …)
  - Custom Extensions
  - M+U (No S)
  - Hardware LSU Misalignment Support (no exception)
  - CSR: MTVEC ReadOnly
  - …

RISC-V Reference Choices

- RISC-V is highly configurable
- So it can get a little .... complicated
- 60 Questions ?



riscvOVPsim as the Reference Model *for Compliance Testing*



Imperas riscvOVPsim Compliance Simulator

- Industrial quality, free ISS / reference model for compliance testing
  - GitHub.com/riscv/riscv-compliance
  - GitHub.com/riscv/riscv-bitmanip
- Model is built using Open Virtual Platforms (OVP) APIs
- Implements full RISC-V envelope
  - Configurable for all features and version
- Includes full open source Apache 2.0 model
- Kept up to date for specification changes
- Works ‘out of the box’ with full tracing, debug, and many other options
- Video: http://www.imperas.com/riscvovpsim-a-complete-risc-v-iss-for-bare-metal-software-development-and-specification-compliance
- Has some limitations which make it not appropriate as a reference model for RTL design verification (DV)

Additional Capabilities Needed for a DV Reference Model

- Support for multi-hart processors
- Support for custom instructions
- Support for injection of external / asynchronous events
- Support for step-and-compare DV flow

OVP RISC-V Model and Imperas Simulator as Reference

imperas

RISC-V Reference Model & Simulator

http://www.imperas.com/riscv

- Support for multi-hart processors
- Support for custom instructions
- Support for injection of external / asynchronous events
- Support for step-and-compare DV flow
- Used as golden reference in RISC-V Foundations' Compliance Suite and Bit Manipulation group

- In use as reference with customers for RTL DV

Flow to add new custom instructions

imperas

Characterize C Application

- Instruction Accurate Simulation
- Trace / Debug
- Timing Simulation
- Function Timing / Profiling

Develop New Custom Instructions

- Design Instructions
- Add to Application
- Add to Model
- Add Timing

Characterize New Instructions in Application

- Instruction Accurate Simulation
- Trace / Debug
- Timing Simulation
- Function Timing / Profiling

Optimize & Document model

- Instruction Coverage
- Line Coverage
- Instruction Performance
- Generate PDF model doc

Release & Deploy

- Check RISC-V Compliance
- Use as reference for RTL Design Verification
- Use in Imperas/OVP Platforms, EPKs
  - Heterogeneous / Homogeneous
  - Multi-core, Many-core
- Imperas Multi-Processor Debug, VAP tools
- Port OS, RTOS (Linux, FreeRTOS…)
- Use in many simulation envs (inc. SystemC)
- Deliver to end users

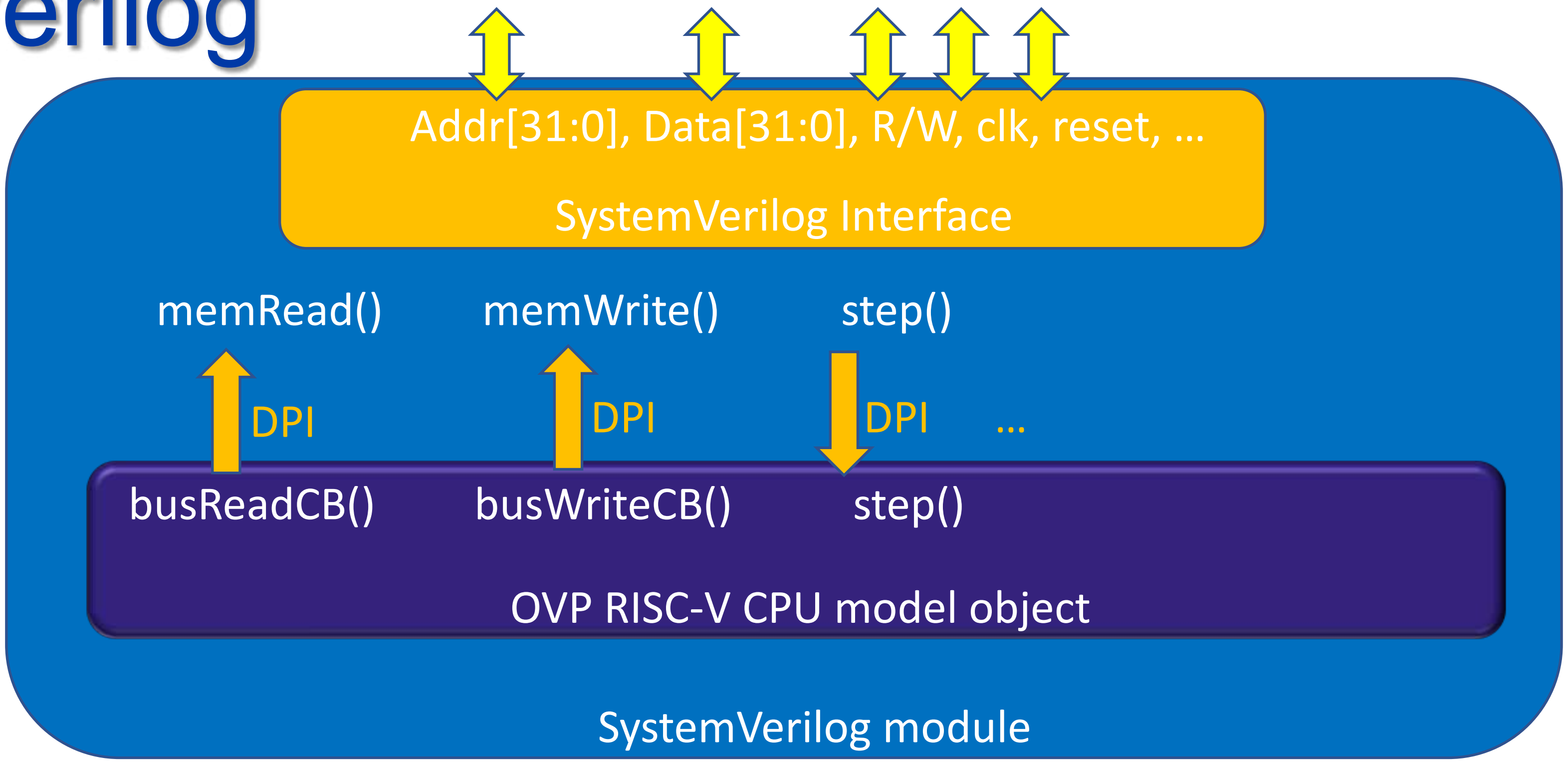
Agenda

- RISC-V verification issues
- Compliance is not verification
- Reference models and custom instructions
- **Processor IP verification**
  - **Step and compare methodology**
  - **Directed tests**
  - **Instruction stream generation**
  - **Test generation and execution**
- SoC level verification
- Summary

DV Methodology: Step and Compare vs Trace/Signature Compare

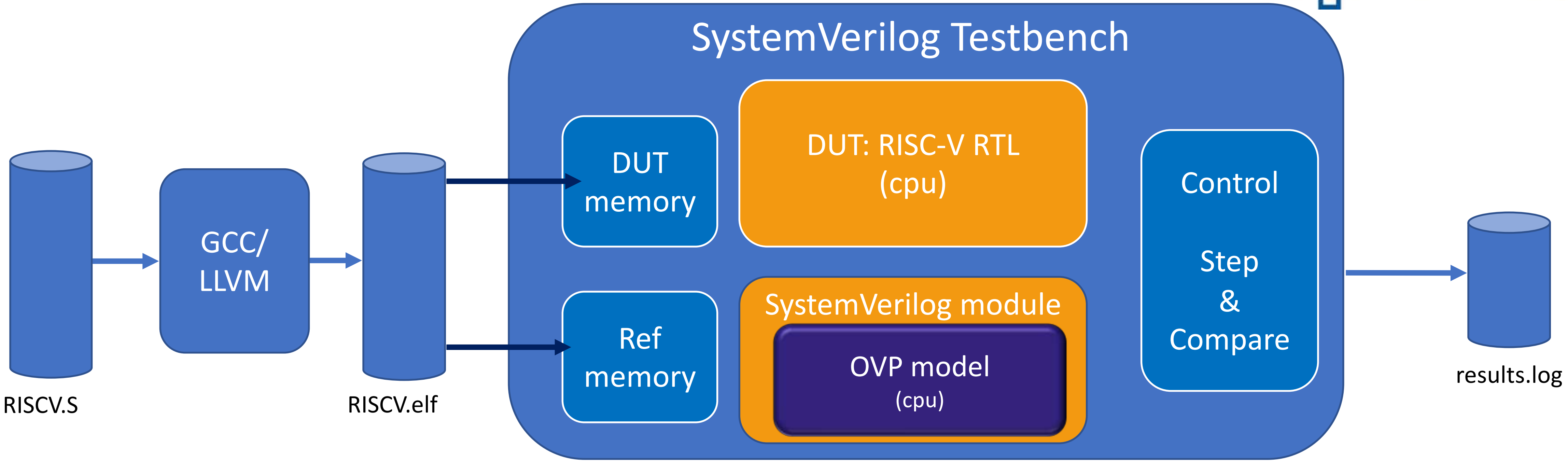
- Short answer: bottom line is DV resources used
- With trace / signature comparison, failures are not known until after the simulation has completed; this can be a long time for a complex test, and therefore could waste simulation resources
- Step and compare enables failures to be flagged and the simulation stopped when the failure occurs

Step and Compare Requires Encapsulation of the Reference in SystemVerilog



- The OVP model is a binary shared object of a RISC-V CPU model
- Encapsulated into a SystemVerilog module, using SystemVerilog DPI
  - Interfaces being: reset, step, address bus, data bus, interrupts, etc.,…
- Instanced in SystemVerilog design or testbench like any module

Step and Compare Flow



- Testbench loads .elf program into both memories, resets CPUs (RTL and OVP model)
- Steps CPUs (DUT and reference), extracting data, and comparing
  - There is no stored log file – test log data is dynamic and requires two targets to be run and compared

Directed Testing

- Test Encoded Self Checking
  - Tests are written with expected behaviour encoded
  - Tests can introspect the state and (self) diagnose faults
- Reference Comparison Checking
  - Tests are written without predicting the result
  - A reference is consulted for the correct value

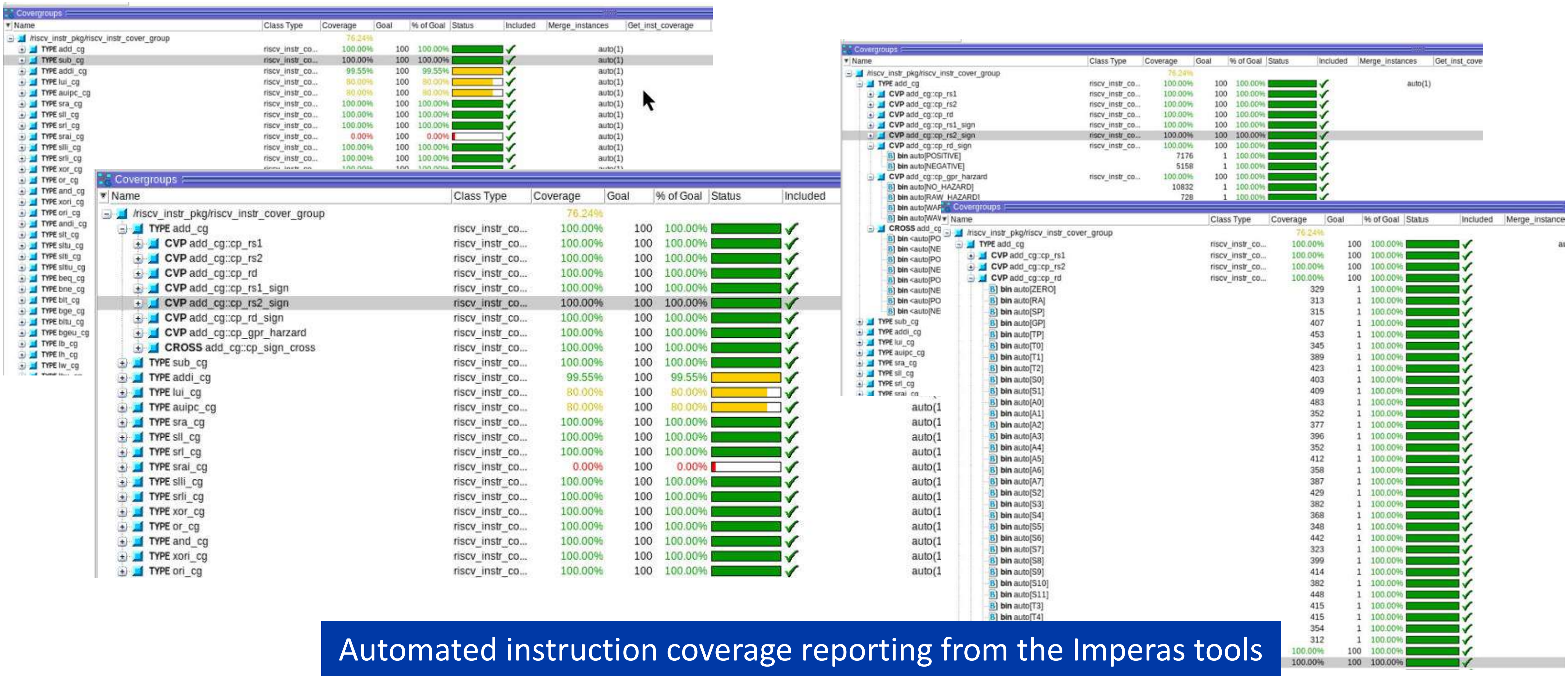
```
// Device Under Test
int a = 4; int b = 5;
int c = a + b;
// c == ?
```

```
// Reference
int Ra = 4; int Rb = 5;
int Rc = Ra + Rb;
// Rc == 9
```

```
assert(c == Rc) // external (@runtime or post-processed)
```

Key Issue for Directed Testing: Coverage

imperas



Automated instruction coverage reporting from the Imperas tools

Coverage images from Mentor Questa SystemVerilog UVM Simulator

Instruction Stream Generation

- Instruction stream generation (ISG) generates random streams of instructions
- Generator given guidance to target specific instruction types and values
  - Many constraints required to get legal instruction sequences
- No predicted results, relies upon reference

- This is just constrained random generation repurposed to processor DV
- Constrained random generation is a well-established part of SoC DV flows

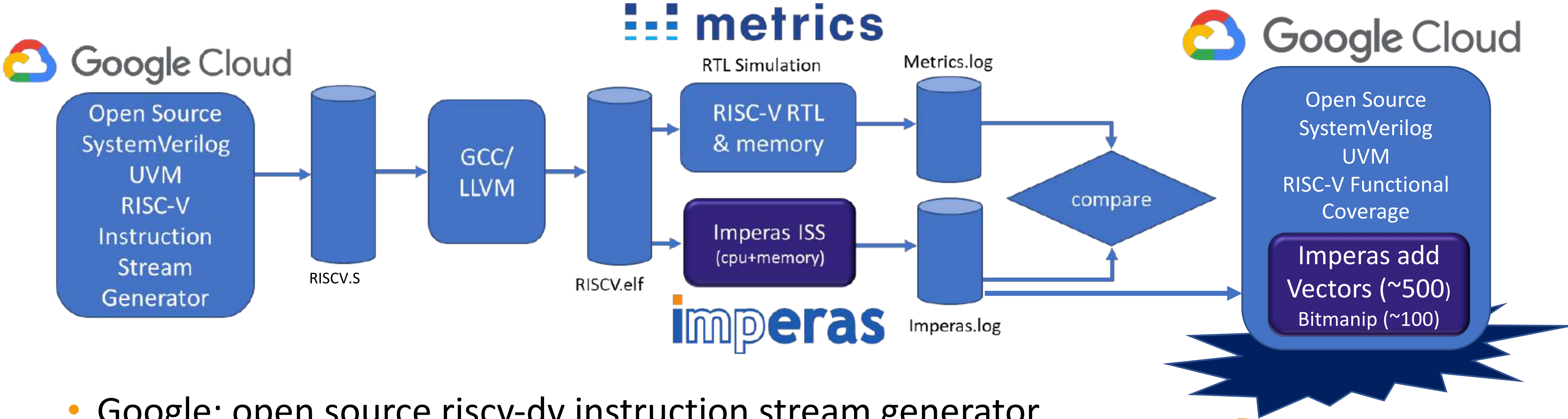
Google RISC-V Instruction Stream Generation

- High quality SystemVerilog UVM DV infrastructure
- Open source (Apache 2.0)
- Drives a RISC-V core through corner cases and pushes it to the limit
- Requires reference and DUT to generate instruction trace disassembly
- Traces compared as post-process (neutral CSV format)
- Can compare values **and** program flow
  - dependant upon target capability
- Provides coverage for test quality, and to aid guidance

Open Source SystemVerilog UVM RISC-V Instruction Stream Generator

https://github.com/google/riscv-dv

Constrained Random Testing

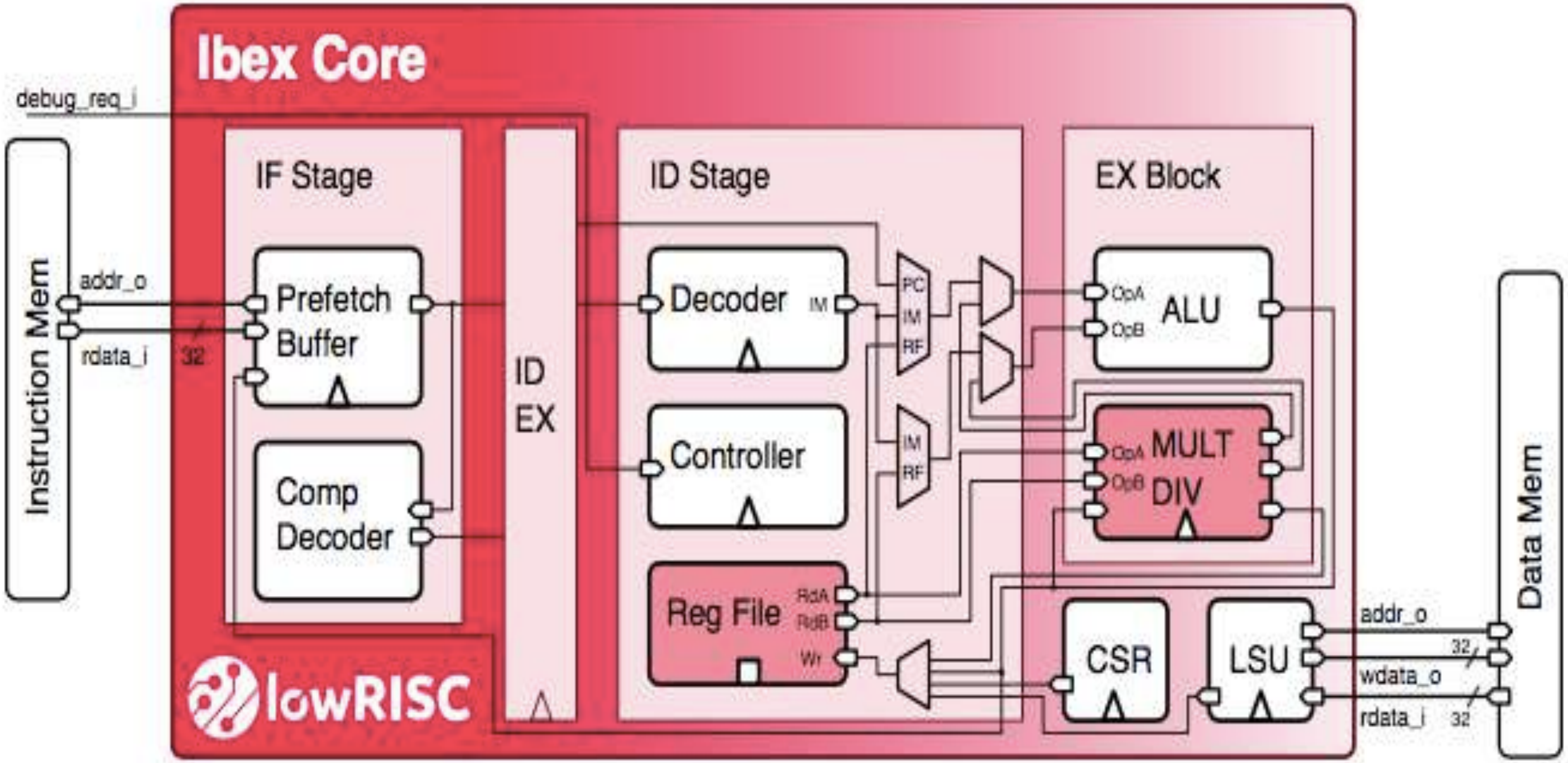


- Google: open source riscv-dv instruction stream generator
- Metrics : SystemVerilog design + UVM simulator for RTL
  - Now working with Cadence, Mentor, Synopsys RTL simulators
- Imperas: model and simulation golden reference of RISC-V CPU

- <u>Imperas have added Vector and Bitmanip extension instructions to the Functional Coverage</u>

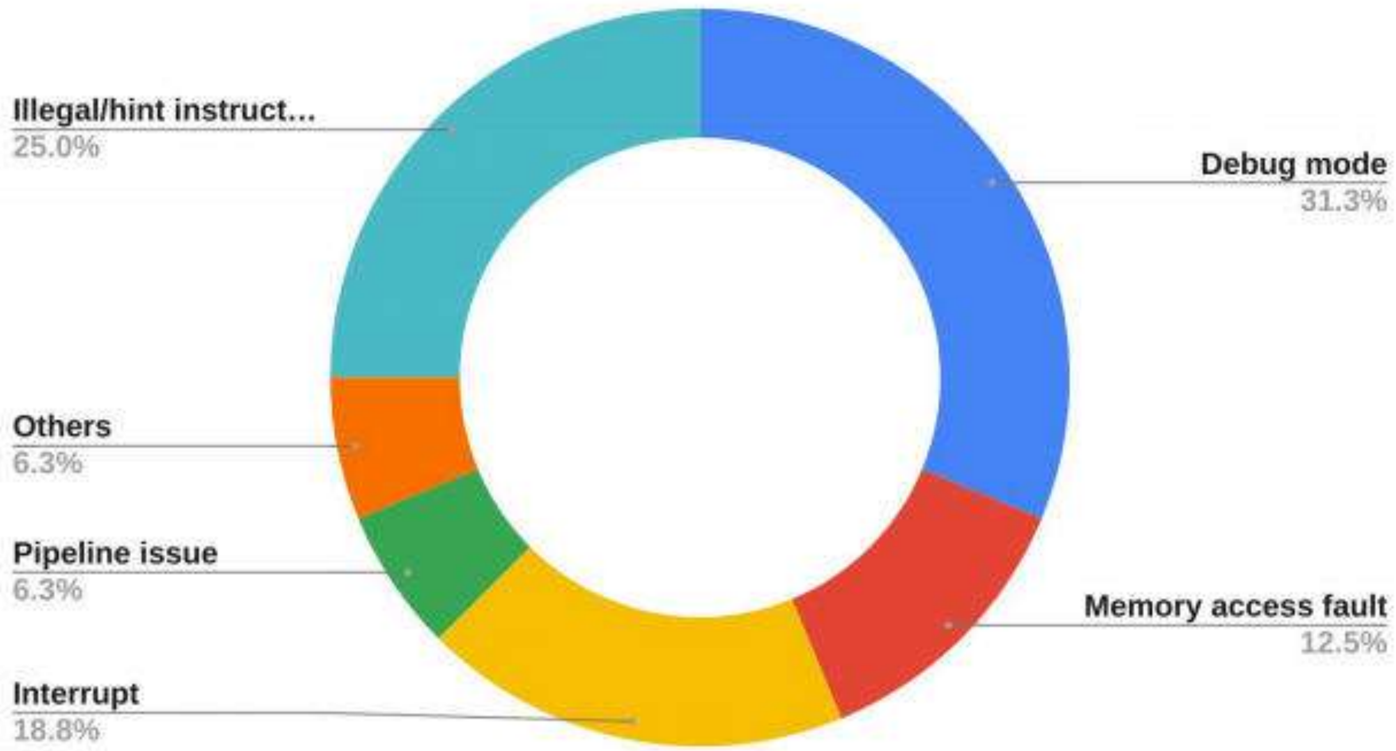
(not yet publicly released)

Case Study: lowRISC Ibex



- Ibex is a small 32 bit RISC-V CPU core (RV32IMC/EMC) with a two stage pipeline, previously known as zero-risky (PULP)
- https://github.com/lowRISC/ibex

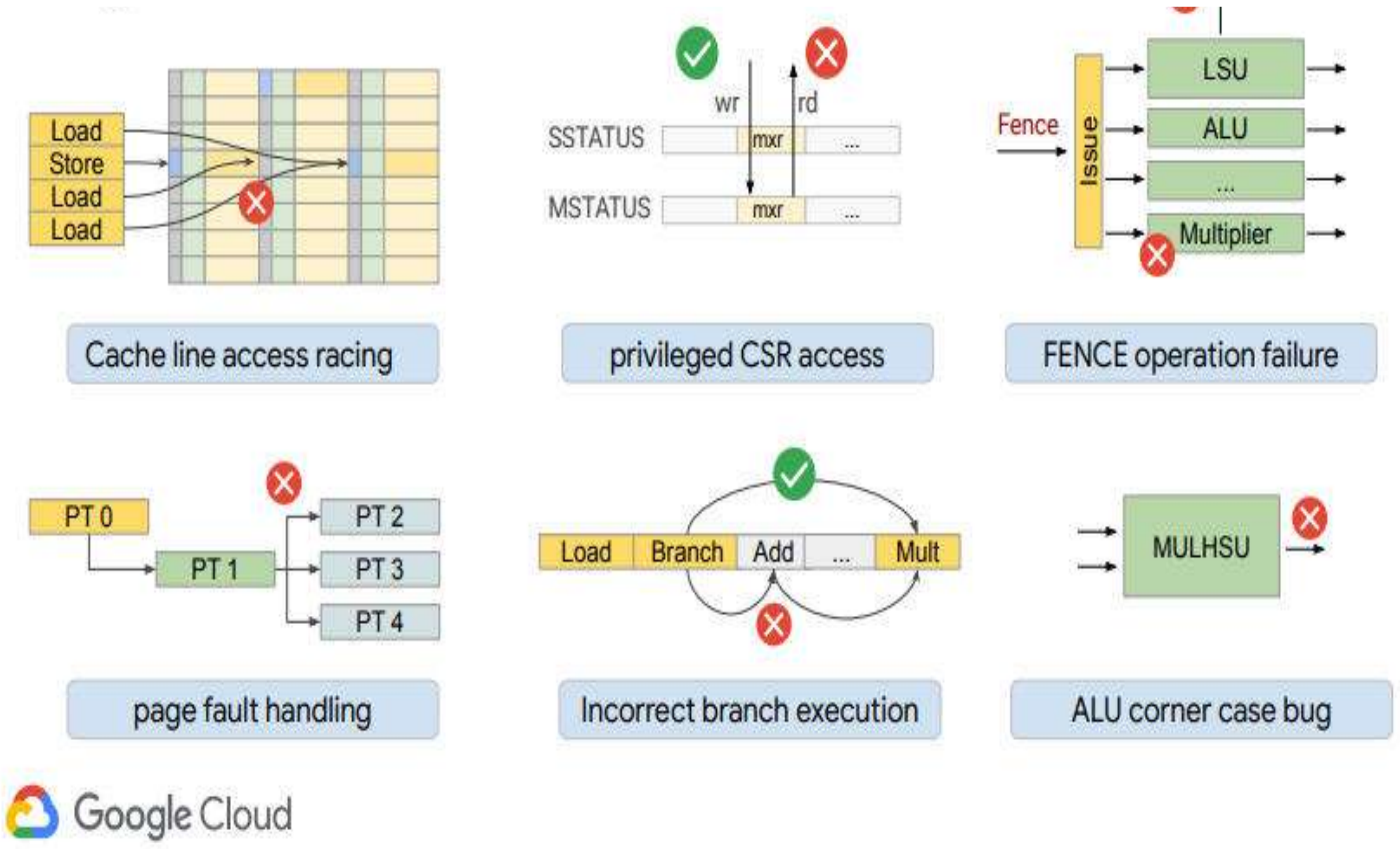
Case study : Ibex core verification



Google Cloud

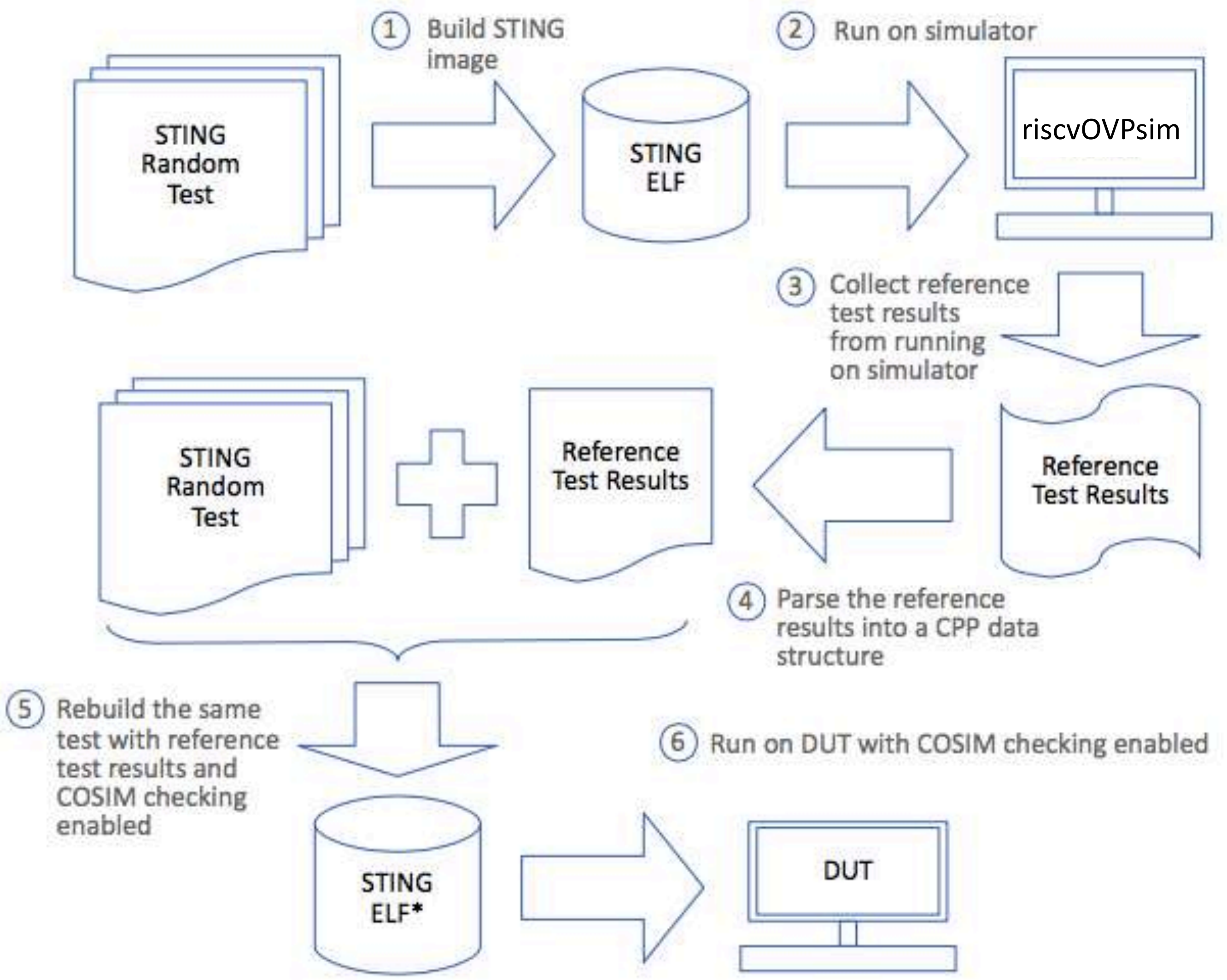
Categories of found bugs

Bugs Found Using ISG Approach



Valtrix Builds Executable Test Benches

- Imperas working with Valtrix
- riscvOVPsim as reference model
- Alternative/complementary approach to Instruction Stream Generator



Agenda

- RISC-V verification issues
- Compliance is not verification
- Reference models and custom instructions
- Processor IP verification
- **SoC level verification**
- Summary

SoC Level Verification

- What about …

- Verification of processing elements with multiple RISC-V cores, as is common in AI/ML SoCs?
  - This flow is still evolving / being invented
- Verification of the interface between the processor or processing element and the NoC
  - Simple answer is that NoC verification IP is used
  - This takes some effort, which is taken for granted with traditional processors

Agenda

- RISC-V verification issues
- Compliance is not verification
- Reference models and custom instructions
- Processor IP verification
- SoC level verification
- **Summary**

Summary

imperas

- DV is a critical issue for RISC-V processor IP and SoCs
- Compliance testing is a subset of DV
- Reference models are needed, and are now available
- Directed testing, instruction stream generation and test generation/execution are being used for processor IP DV
- Step and compare methodology provides the most efficient DV flows
- More work is needed

Thank you

Larry Lapides

LarryL@imperas.com