imperas

Methodologies for RISC-V Processor Verification

Aimee Sutton, Director of Product Engineering, Imperas Software Ltd.

Lee Moore, Senior Applications Engineer, Imperas Software Ltd.

Simon Davidmann, CEO, Imperas Software Ltd.

RISC-V Verification Challenges

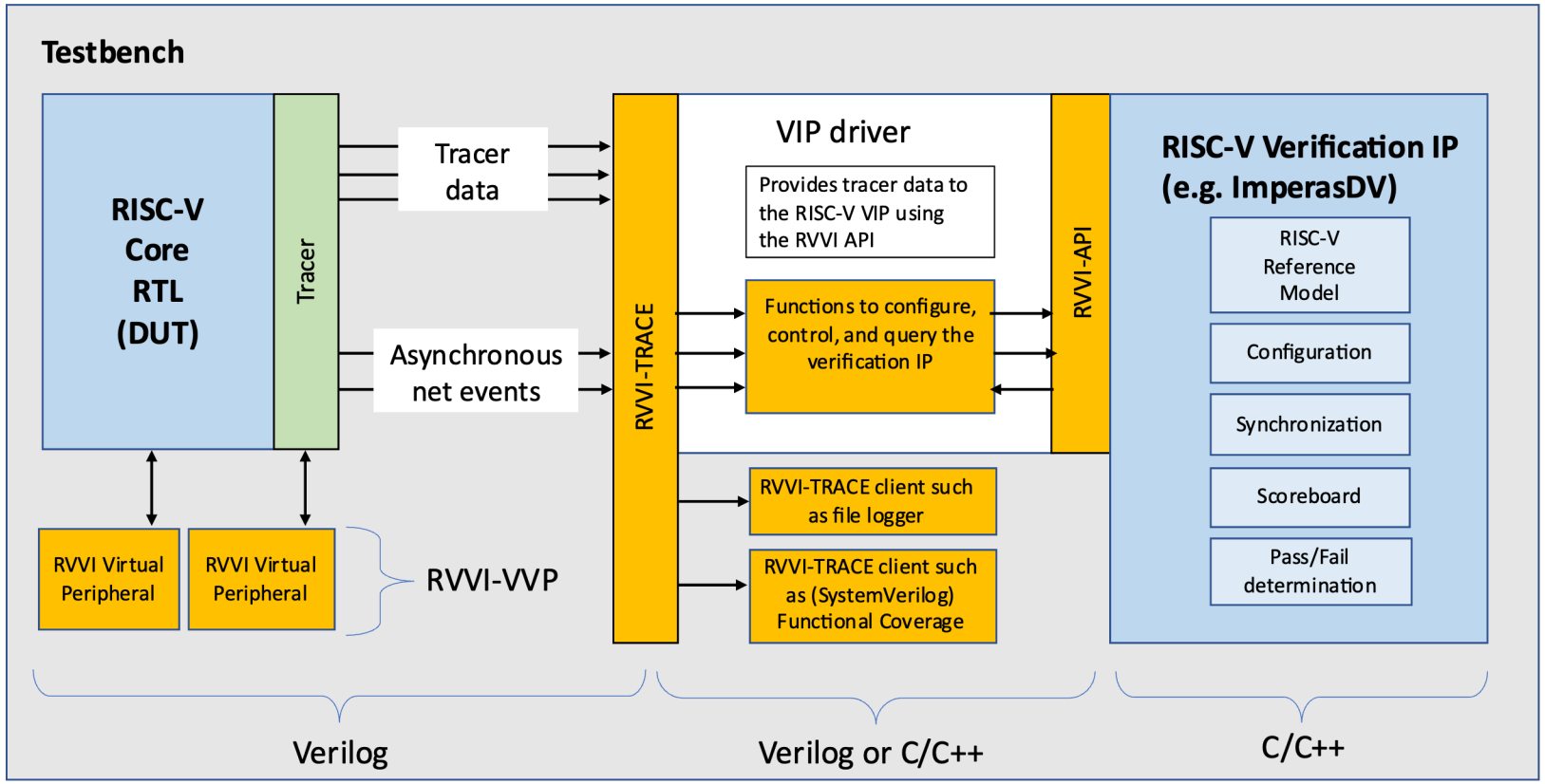
- Processor users are now processor developers
  - Accustomed to pre-verified IP, now must assume the verification burden
- RISC-V configurability and customizability
  - Need to verify base ISA functionality as well
- Processor DV techniques are not well-known
  - Knowledge was contained within specialized companies
  - Existing techniques for ASIC/SoC are useful but insufficient
- Commercial solutions largely unavailable
  - Fortunately, this is now changing

RISC-V Verification Innovation

- Open standards: RISC-V Verification Interface (RVVI)
- Comprehensive functional verification: continuous compare lock-step co-simulation methodology
- Functional coverage: RISC-V ISA, MMU, PMP
- Hardware assisted verification: running application software using FPGA prototyping / emulation
- Case study: OpenHW Group CVW (Wally)

Open standard: RISC-V Verification Interface (RVVI)

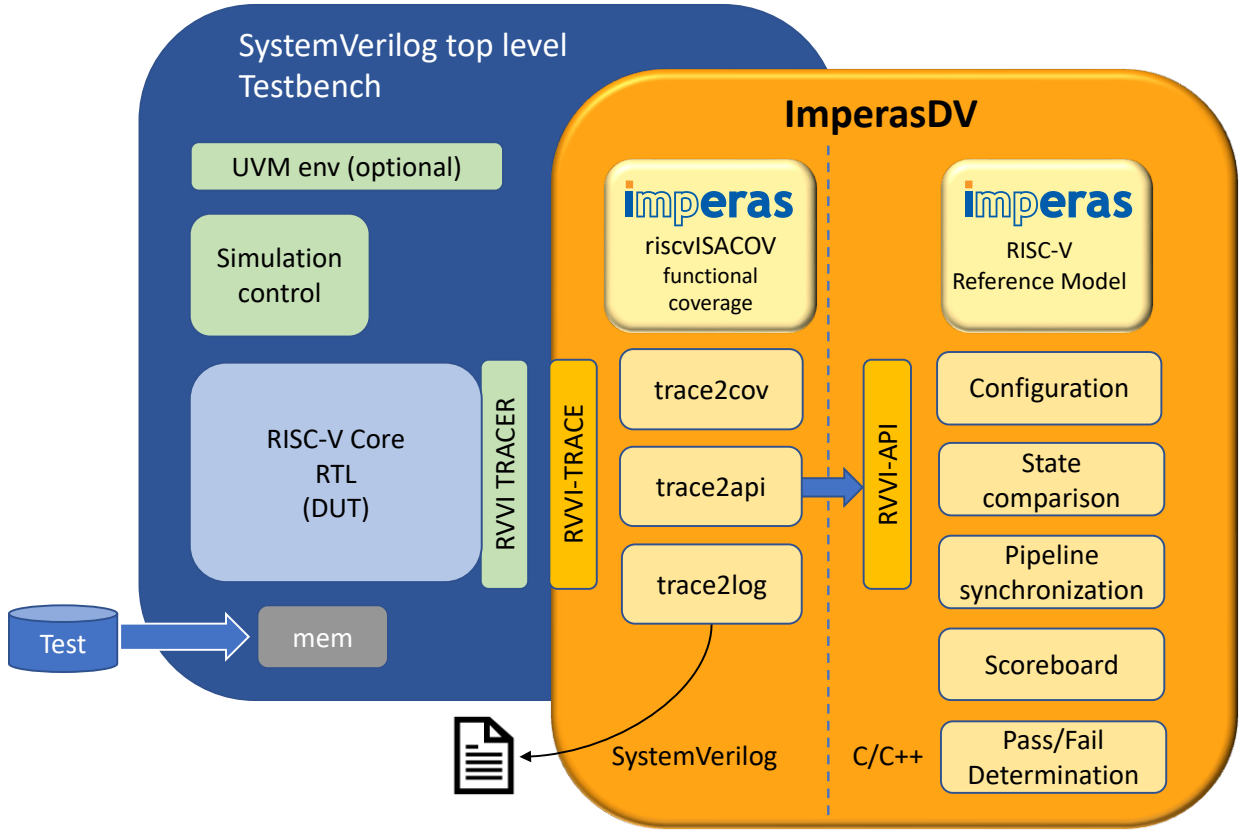
- Standardize communication between DUT, testbench, and RISC-V VIP
  - RVVI-TRACE: provides tracer data to RISC-V VIP
  - RVVI-API: function level interface to RISC-V VIP
  - RVVI-VVP: virtual peripherals
- Collaborative work has evolved over 3 years
  - Imperas, EM Micro, SiLabs, OpenHW Group



https://github.com/riscv-verification/RVVI

Comprehensive Verification:
Continuous Compare Lock-Step Co-Simulation

- Same software running on DUT and processor reference model
- DUT state and asynchronous events communicated via RVVI-TRACE
- As each instruction retires, internal state is continuously compared
- Errors flagged immediately: no wasted simulation cycles
- Enables debug at the point of failure
- Interrupts, debug mode requests handled by VIP



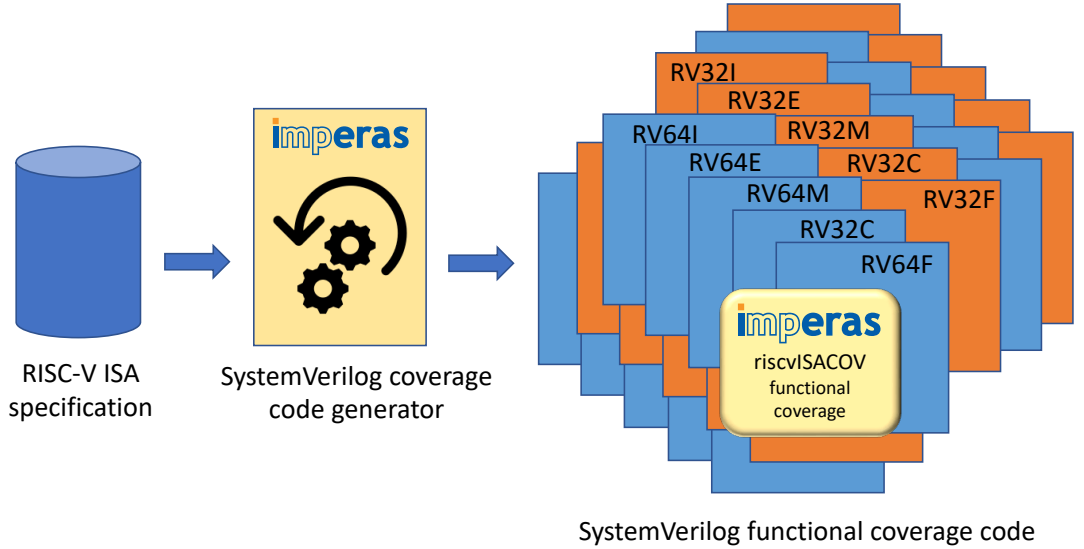
Functional Coverage for RISC-V ISA

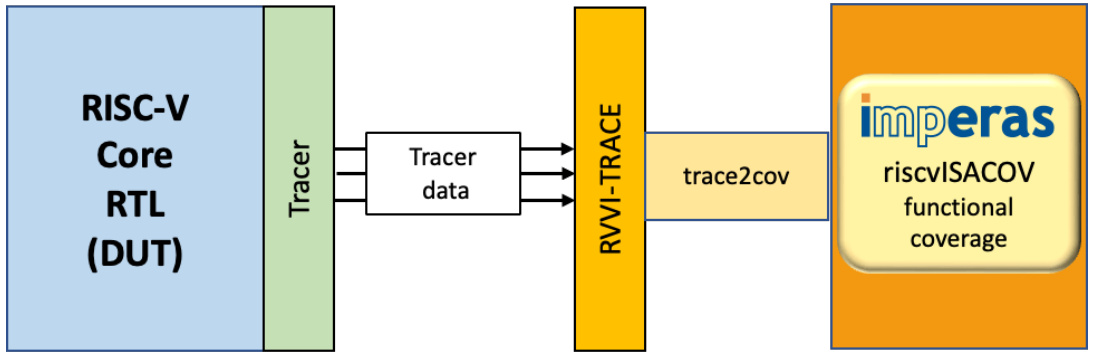
The problem:

- Many extensions = significant number of instructions, operands, registers to cover
- Large amount of SystemVerilog code required: tedious and error-prone

The solution:

- Use a machine-readable version of the RISC-V ISA to auto generate reusable covergroups and coverpoints
- Use RVVI-TRACE data to sample the generated covergroups





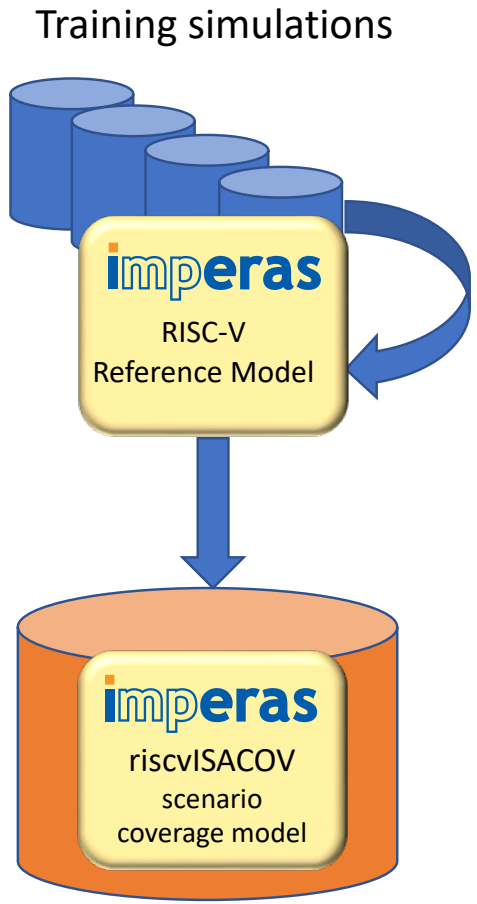
Scenario Coverage for MMU and PMP

Problem:

- Observing MMU and PMP operation requires deep white-box probing into RTL
- How to measure the occurrence of interesting scenarios?
- How to identify the interesting scenarios and build a coverage model?

Solution:

- Use processor modeling experience and proprietary techniques to develop trained scenario coverage models
- Take advantage of processor model visibility to eliminate the need for white-box probing



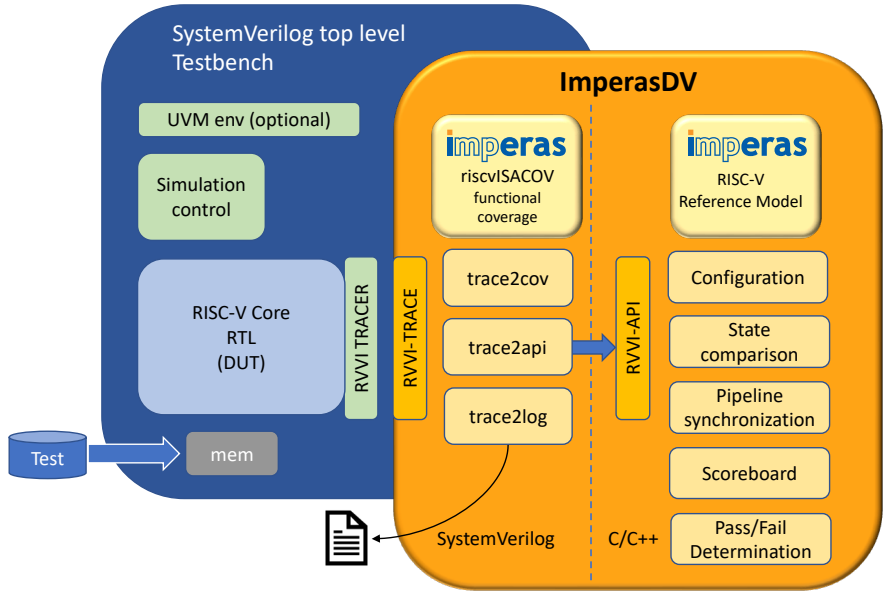
RVVI + Hardware Assisted Verification

Problem:

- Need to simulate application software running on a custom RISC-V processor
- Traditional simulation is too slow, however the same level of checking is desired

Solution:

- Connect ImperasDV to Verilog HW Simulation acceleration using a synthesized RVVI tracer



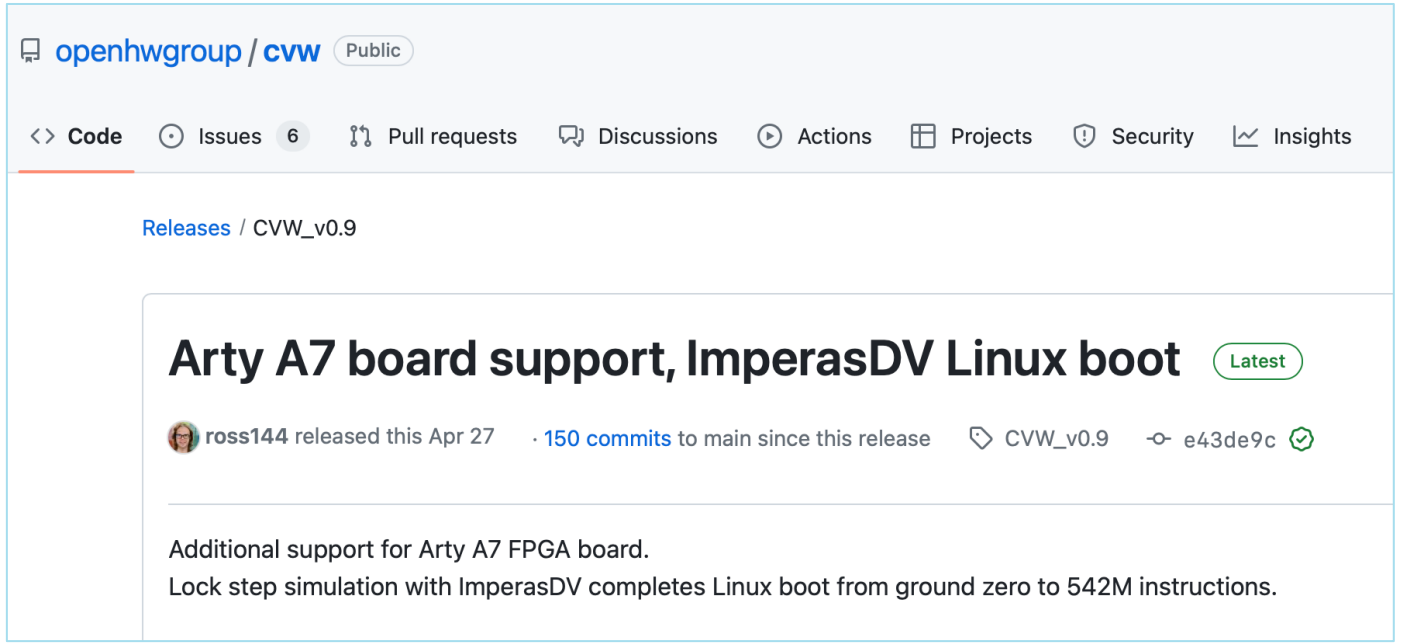
Case Study: Wally RISC-V Core

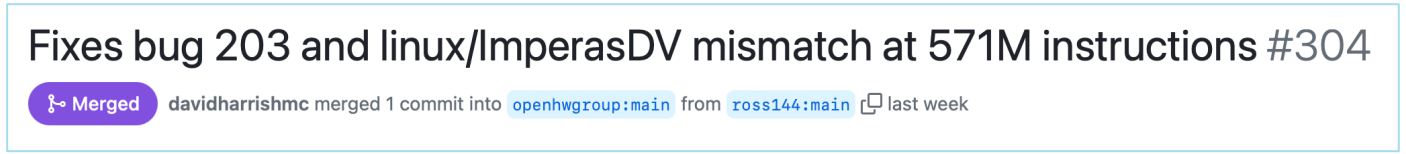
- Configurable core:
  - RV32I, RV32E, RV64I, RV64E
  - A, C, F, D, M extensions, privileged modes, CSRs
  - MMU/TLB virtual memory, caches
- Developed at Harvey Mudd College / Oklahoma State University
  - Focus: high quality core for processor architecture education
  - Now in OpenHW as CORE-V Wally (https://github.com/openhwgroup/cvw)

- Status in January 2023 – before starting to use RVVI + ImperasDV for verification:
  - Passing all RISC-V International compliance tests, Imperas compatibility tests
    - Using Compliance Level post-simulation signature file compare
  - Boots Linux

Wally + RVVI – Status (July 2023)

- RVVI Tracer + testbench integration: 3 days of effort
- Results:
  - 20+ bugs found in simulation almost immediately using ImperasDV and riscv-dv
  - Reached Linux prompt with continuous checking: 2 days of simulation
  - One bug found just after the Linux command prompt (!)
  - Functional coverage achieved by booting Linux: covergroups 37% (bins 3%)
- Future work:
  - Boot Linux with co-sim using hardware assisted verification
  - Achieve 100% functional coverage using constrained-random tests





Conclusion

- Innovation is the solution to the RISC-V verification challenge
- RVVI enables the implementation of many techniques
- Continuous compare co-sim: highest quality and most efficient verification method
- Functional coverage: reusable machine-generated ISA functional coverage; trained scenario coverage model
- Accelerated simulation of application software using hardware platforms
- Many more innovations are possible in the future
- Thank you!
- aimees@imperas.com moore@imperas.com simond@imperas.com