RISC-V SUMMIT

imperas

An introduction to RISC-V Verification with RVVI

The RISC-V Verification Interface

Aimee Sutton aimees@imperas.com

@ImperasSoftware

Agenda

- RISC-V verification challenges
- Standardization: RVVI
- Applications of RVVI

Agenda

- **RISC-V verification challenges**
- Standardization: RVVI
- Applications of RVVI

Challenges in RISC-V Processor DV

- Feature selection and design choices require serious consideration
  - Must consider verification impact
- Current SoC cost is 50% for HW DV (with CPUs bought in as proven IP)
  - Developing own CPU adds incremental schedule, resource, quality challenges
- Processor DV is a new challenge for many teams
  - Traditionally, SoC developers licensed in pre-verified processor IP
  - Now, every RISC-V processor developer is an architecture licensee
- Existing DV methodologies don’t completely address the challenge
- As of 2021, no commercial products available to support DV of processors

Agenda

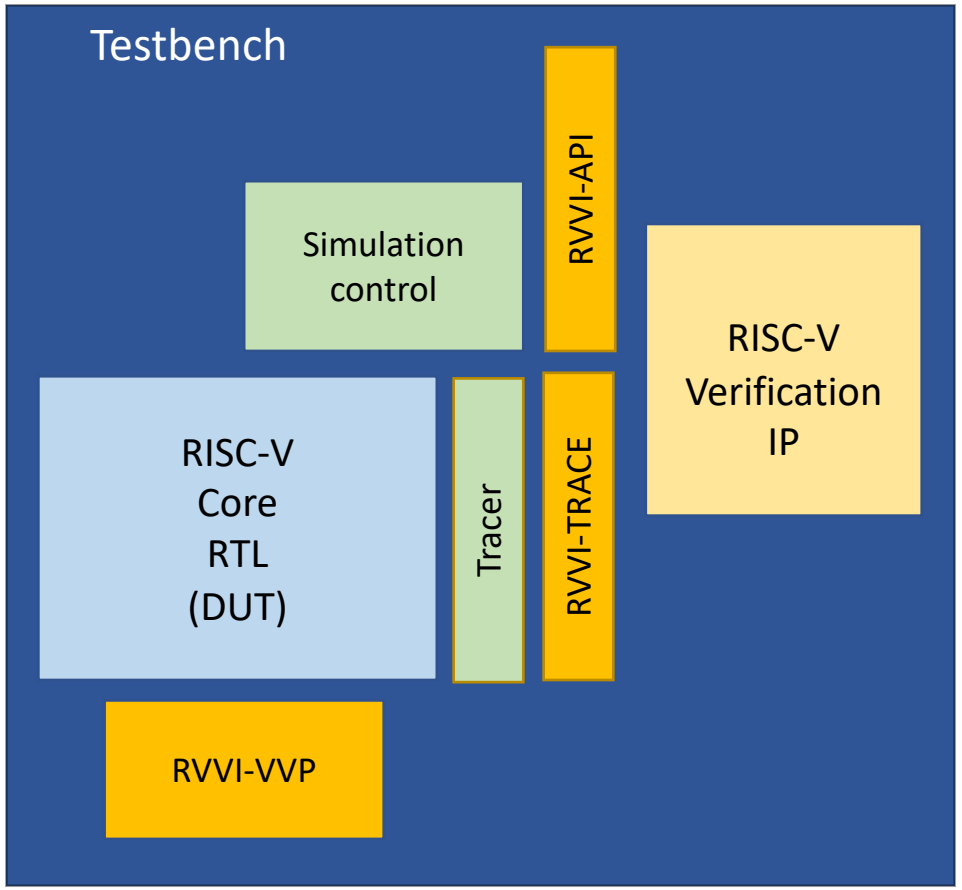
- RISC-V verification challenges
- **Standardization: RVVI**
- Applications of RVVI

Standardization

- Good for IP users and IP vendors
- For users:
  - Supports best practices
  - Reuse and portability
  - Get up and running faster
- For vendors:
  - Less configurability, better quality
  - Ease of customer support
- UVM standard is a good example

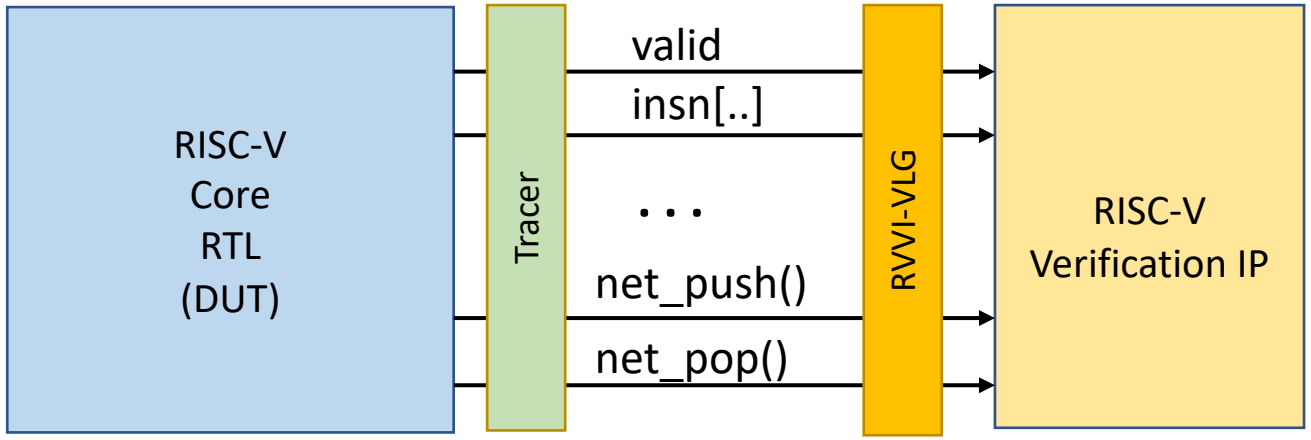
Standardization: RVVI

- RVVI = RISC-V Verification Interface
  - https://github.com/riscv-verification/RVVI
- Work has evolved over 2 years
  - Imperas, EM Micro, SiLabs, OpenHW
- Standardize communication between testbench and RISC-V VIP
- Three parts:
  - **RVVI-TRACE**: signal level interface to RISC-V VIP
  - **RVVI-API**: function level interface to RISC-V VIP
  - **RVVI-VVP**: virtual verification peripherals



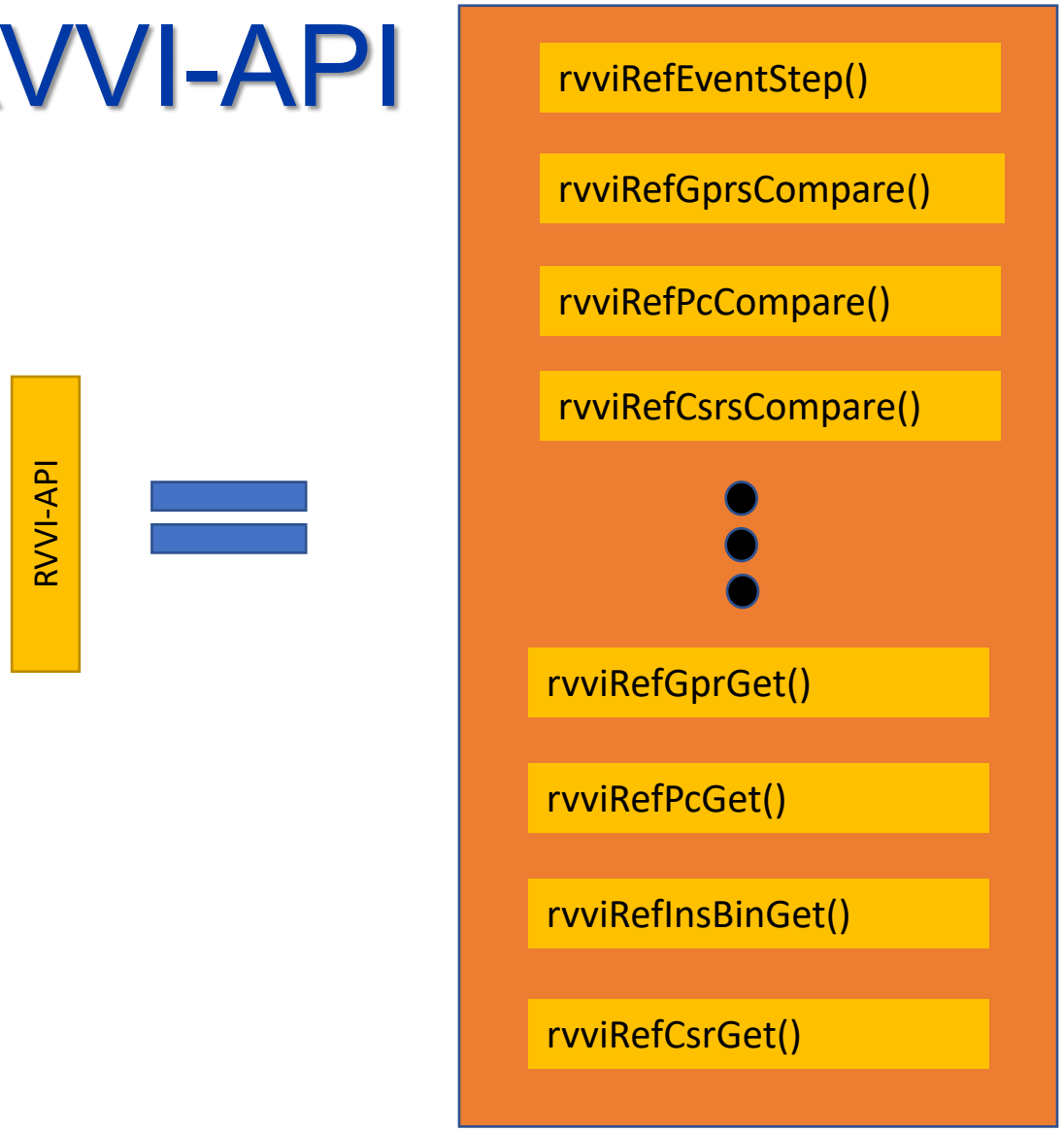
RVVI-TRACE

- Defines information to be extracted by tracer
- SystemVerilog interface
- Includes functions to handle asynchronous events
  - E.g. interrupts, debug req



- https://github.com/riscv-verification/RVVI/tree/main/RVVI-TRACE

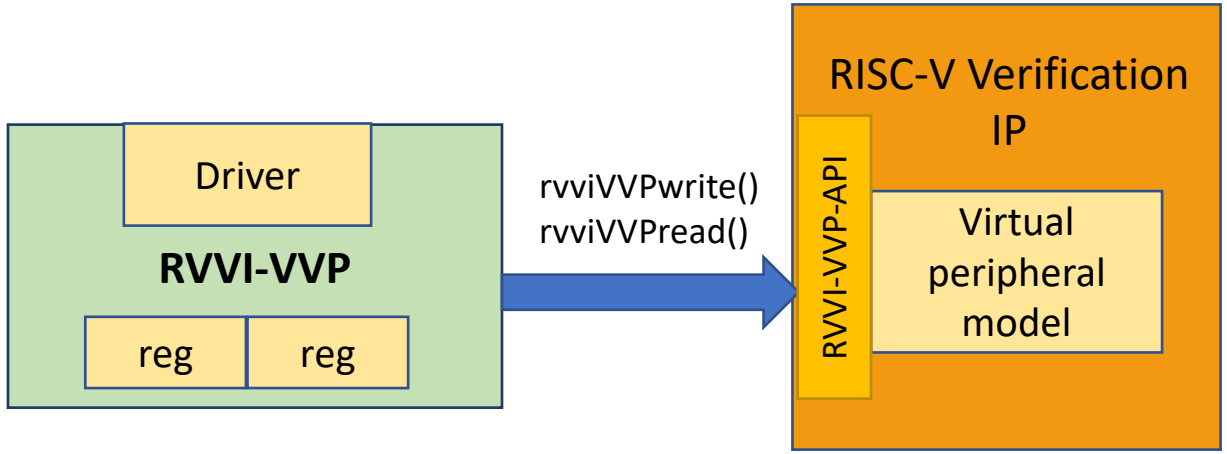
RVVI-API



- Standard functions that RISC-V processor VIPs need to implement
- Supports a step-and-compare methodology
- C and SystemVerilog versions available
- https://github.com/riscv-verification/RVVI/blob/main/include/host/rvvi/rvvi-api.h

RVVI-VVP

- VVP = Virtual Verification Peripheral
- Memory-mapped testbench components
  - E.g. Virtual printer/UART, signature file writer, status interrupt timer control, debug control, instruction memory stall control
- Allows better co-ordination of stimulus on peripheral interfaces with the program running on the core
- Needed for RISC-V compliance tests
- WIP



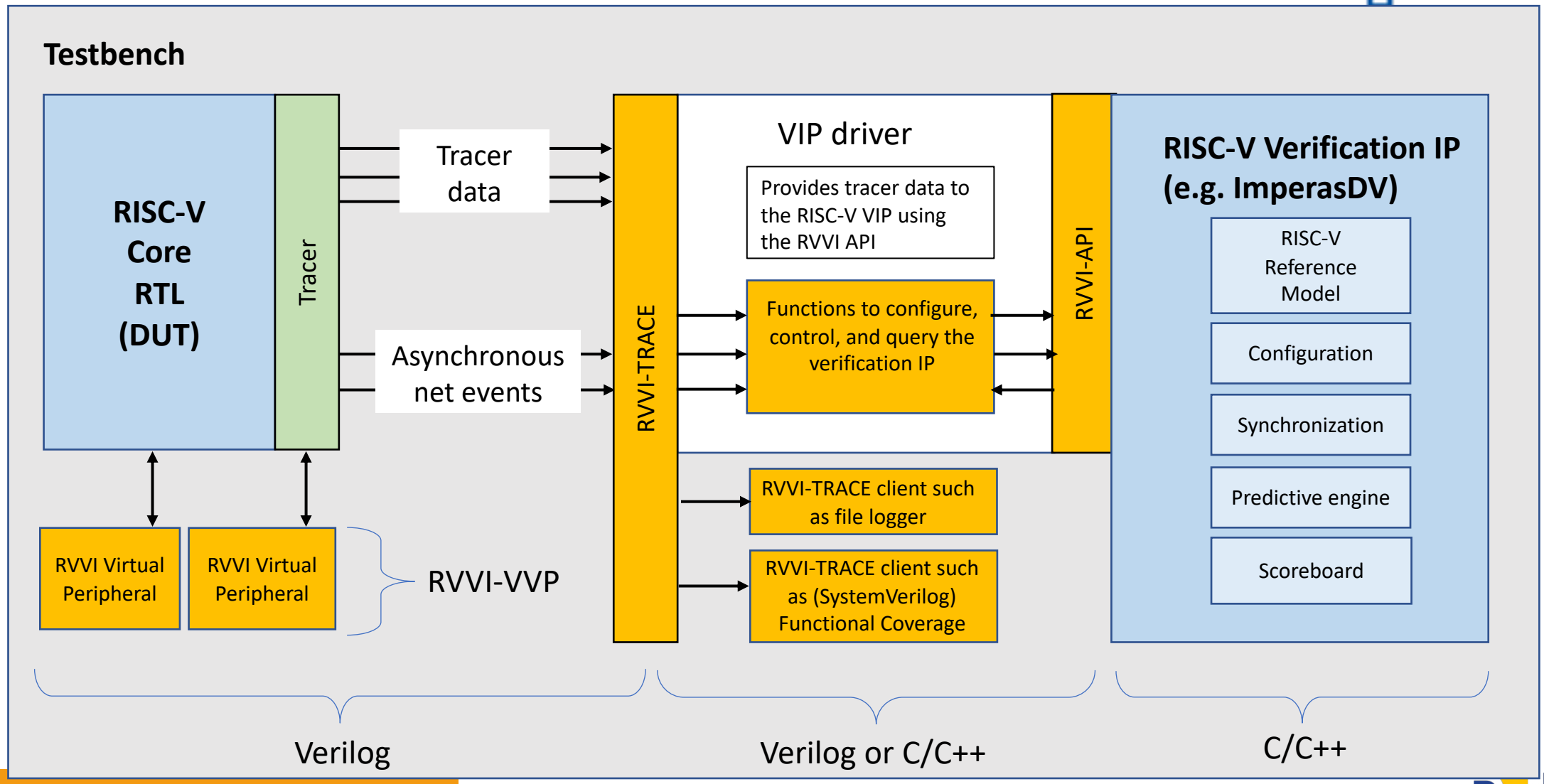
Why RVVI?

- You have to use some interfaces
- No need to re-invent on your own – they do not need to be proprietary
- RVVI (and its predecessor) have already been flushed out
  - in use with several tools, users, cores
- There is no downside to adoption
- RVVI is an open standard available on GitHub
- RVVI helps you understand what you need to develop (in e.g. your tracer)
- RVVI supports RISC-V processor DV best practices
  - step-and-compare, asynchronous events

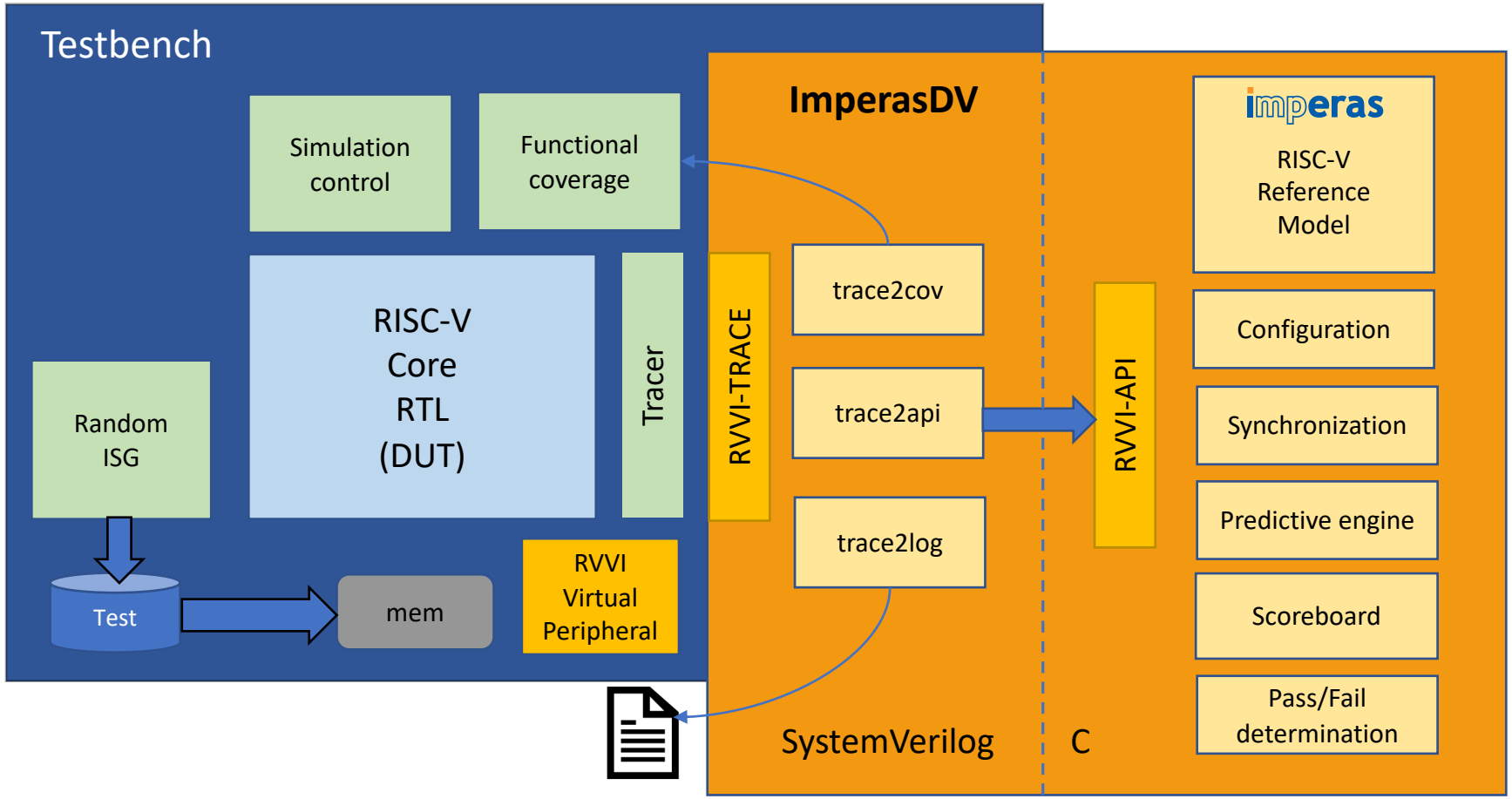
Agenda

- RISC-V verification challenges
- Standardization: RVVI
- **Applications of RVVI**

Testbench for Advanced RISC-V Design Verification



Example usage of RVVI for RISC-V CPU DV (with ImperasDV)



5 components of RISC-V CPU DV

- DUT subsystem with 'tracer'
- Tests: (random) instruction test generator and directed tests
- Functional coverage measurement
- Test bench / harness
- ImperasDV subsystem

- RVVI-TRACE i/f to core tracer
- RVVI-API i/f to verification IP
- RVVI-VVP virtual verification peripherals

Summary

- RISC-V is dramatically moving the processor DV task from the traditional mainstream IP providers to all SoC developers
- Processor DV methodology has been evolved by Imperas, together with customers and partners => RVVI
- Open standards like RVVI are essential to enabling efficient methodologies and advancing the RISC-V ecosystem
- RVVI enables verification IP reuse and engineering efficiency
- Verification IP with RVVI = best practices, open standards, verification productivity gain

Thank you

Aimee Sutton aimees@imperas.com