RISC-V processor verification with new open standard RVVI based methodology

Simon Davidmann (simond@imperas.com)

29-Nov-2022

The Design and Verification Club (Europe)

RISC-V processor verification with new open standard RVVI based methodology

Abstract:

- This talk outlines the RISC-V Verification Interface (RVVI), a new open standard interface for RISC-V verification including the integration methodology for the processor RTL and reference model within a unified SystemVerilog testbench
- It discusses a range of approaches based on the verification test plan needs for proof-of-concept test chips or research projects, to high reliability application and high-volume silicon production
- RVVI also address the complexity of the functional verification for superscalar, out-of-order, multi-hart, multi-thread, vector accelerators, privileged and debug modes of operation
- Together these guidelines help to adapt the current industry standard SoC verification methods for RISC-V processor DV, and establish a common framework that supports reuse and shared contributions across the whole DV community
- This talk highlights the experience in using an RVVI based methodology for the testing of a popular series of open-source IP cores, and guidance for implementing RVVI for new processor DV projects
- Key Points:
  - RISC-V is dramatically moving the processor DV task from the traditional mainstream IP providers to all SoC developers
  - This represents the biggest migration in responsibility for verification in the history of the semiconductor industry
  - RVVI enables verification IP reuse and engineering efficiency via open standards in RISC-V processor DV

Agenda

- Background
- Industry challenges
  - Need collaboration, re-use, ecosystem to afford the time, cost, resources
- RVVI addresses the challenge
- Drill down into RVVI-TRACE, RVVI-API
- RVVI in practical use
- Questions

Background

- Traditionally, SoC developers licensed in pre-verified processor IP
- Now, every RISC-V processor developer is an architecture licensee
- So what approaches are there for verification of these new processors

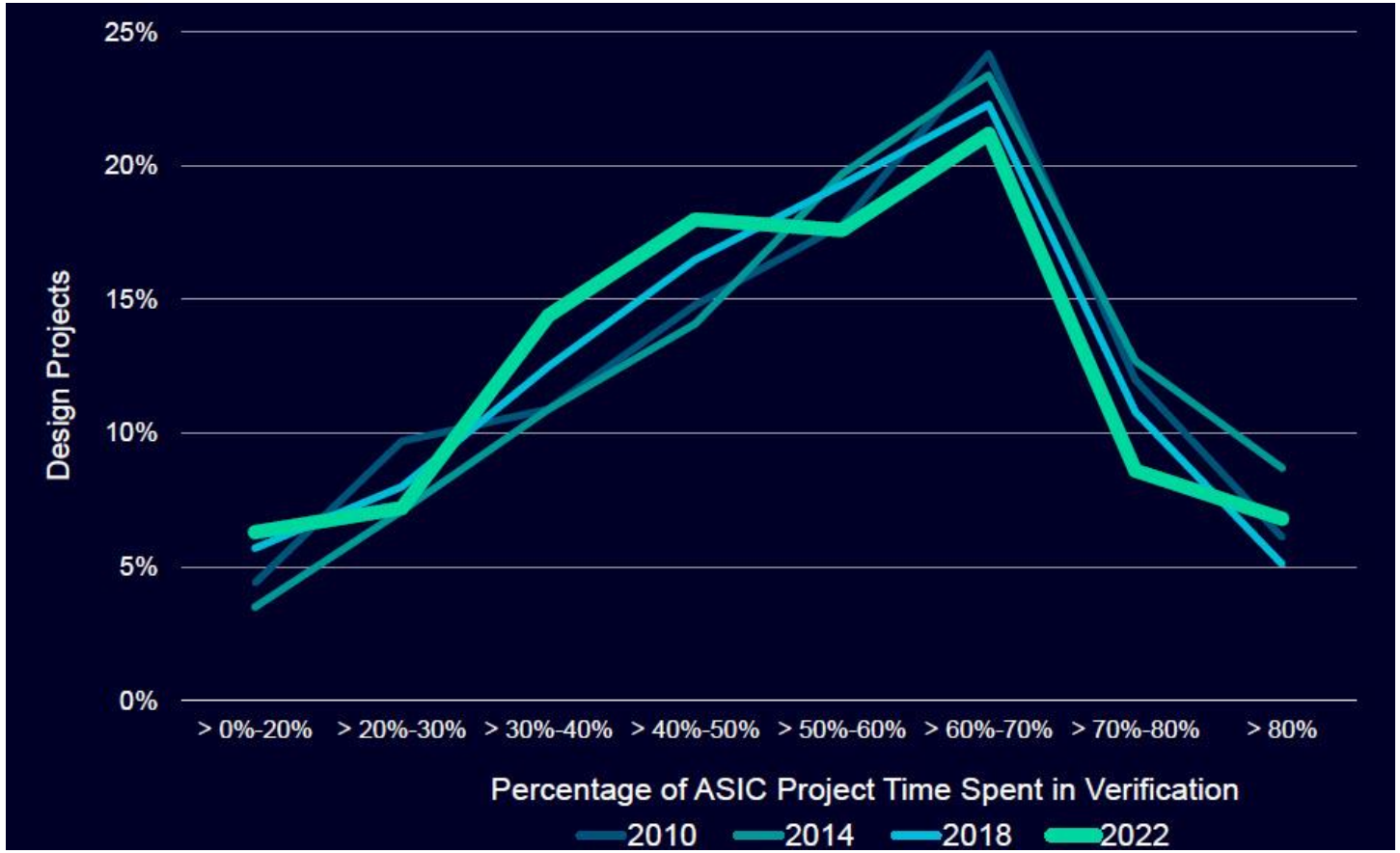
Traditionally, SoC developers licensed in pre-verified processor IP

- "On average every IP is subjected to 5-6 trillion emulator cycles and 2-3 peta FPGA cycles of system validation"

- *"relies on pre-verified ARM cores and subsystems that can be easily integrated into the designs"*

https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/system-validation-at-arm-enabling-our-partners-to-build-better-systems

And DV is still the biggest nightmare…



- *Source: Wilson Research Group and Siemens EDA, 2022 Functional Verification Study*

Now, every RISC-V processor developer is an architecture licensee

- Tremendous choice of instruction extensions
- Large configuration options palette
- Many ‘implementation specific’ choices
- Full ability to add state and instructions for custom operation

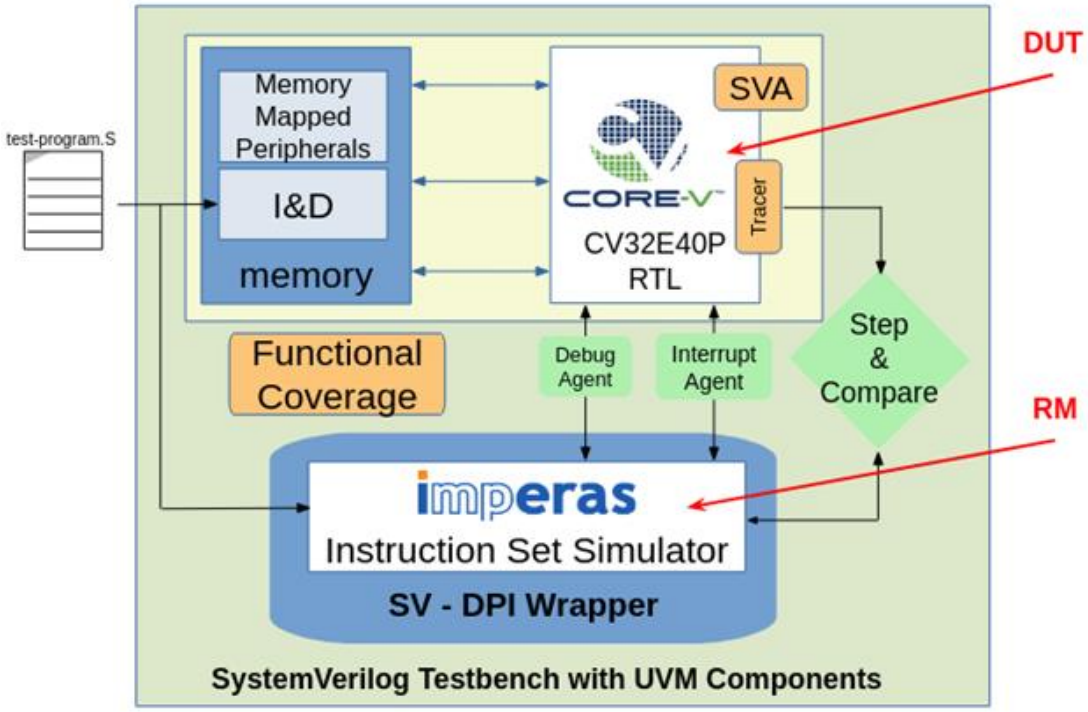
- Requiring not only tremendously skilled design team, but also huge verification resources
  - “every design option doubles verification”

So what approaches are there for verification of these new processors

- Does a program run? – ‘hello world’ tests
- Is there simple correct computation? – ‘self checking tests’
- Signature checking – ‘post simulation signature dump compares’
- Trace log checking – ‘post simulation trace file compare’
- Simple step and compare co-simulation – ‘instruction retire compare’
- Advanced, e.g. commercial solutions – ‘e.g. ImperasDV’

- [Note: this discussion is only about dynamic simulation verification – there are of course many excellent commercial formal verification solutions]

Exploring Imperas co-operation with OpenHW – 1st gen. (c. 2020)

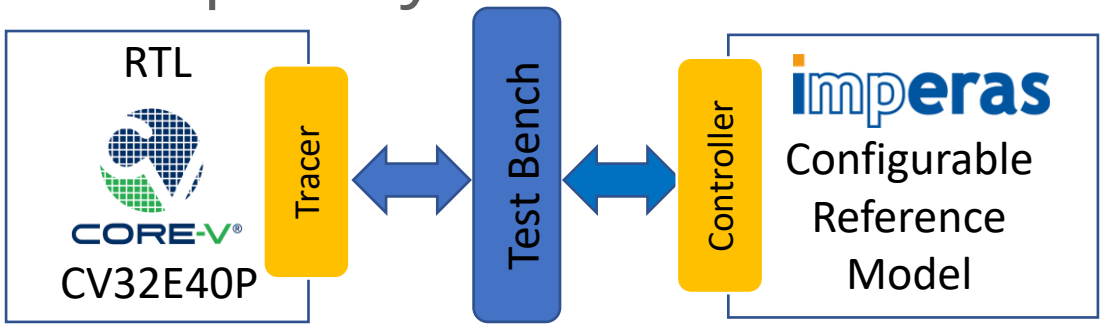


First Generation CORE-V-VERIF

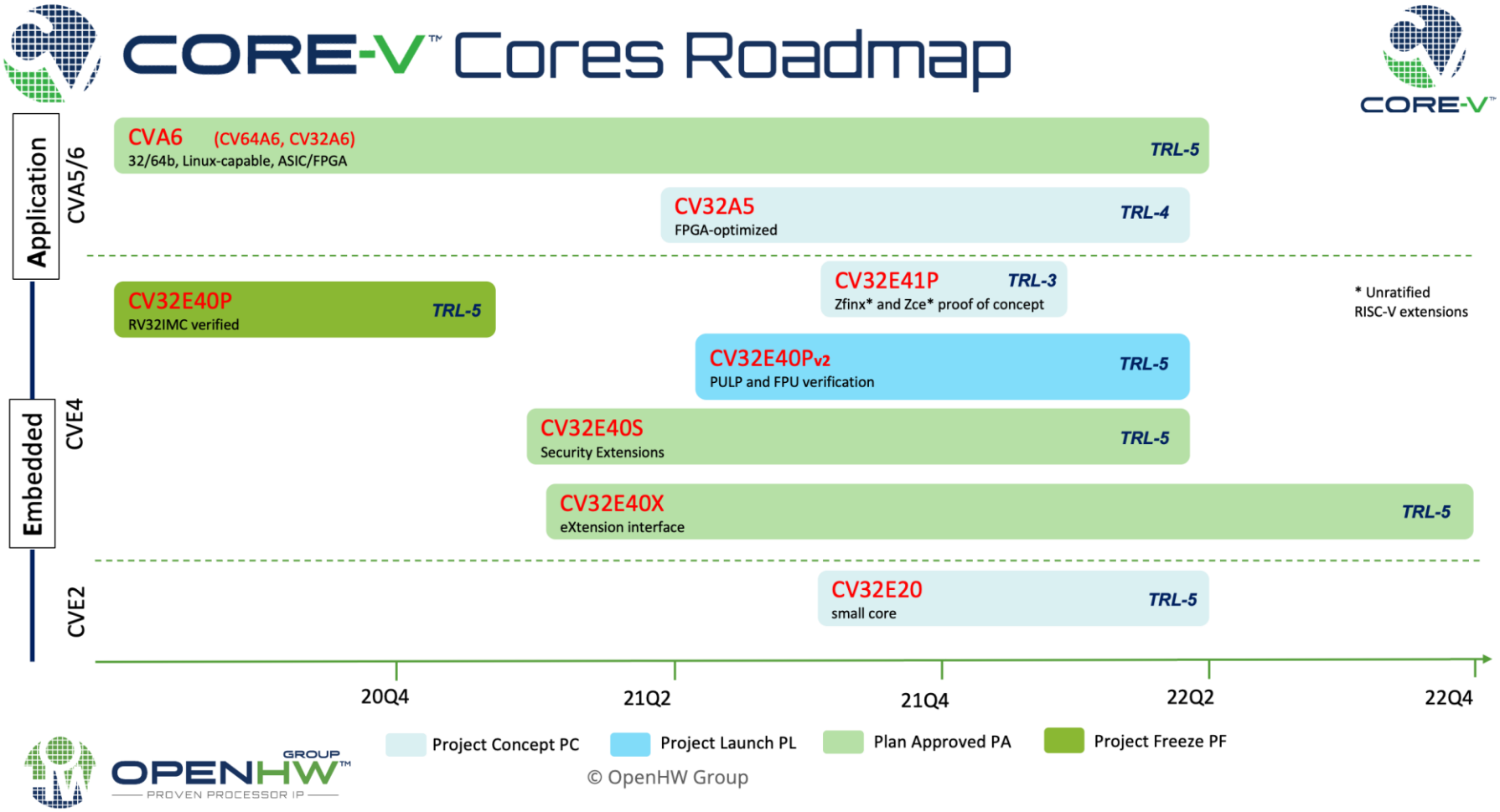
- 3 main components
  - Test bench
  - Core/DUT
  - CPU Reference model

- 2 main interfaces
  - DUT (tracer) - Test bench
  - Test bench - reference model

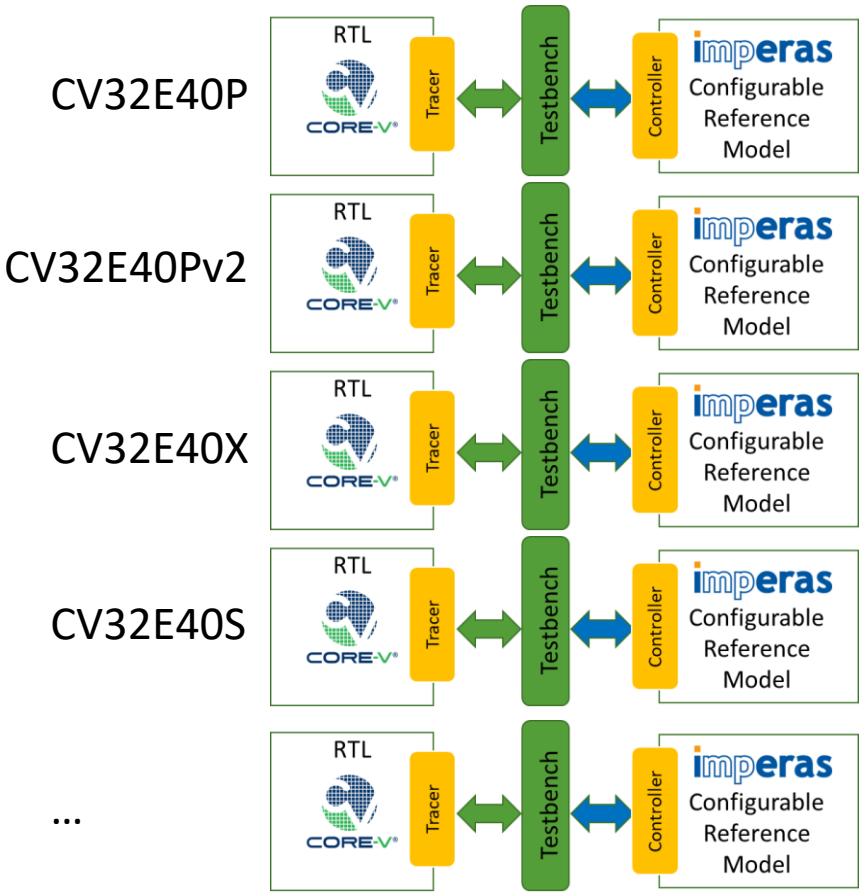
Conceptually:



OpenHW CORE-V cores – current, visible roadmap (c. 2020)



OpenHW roadmap has multiple cores – what are we going to do?



- Are we really going to create bespoke ad-hoc test bench and interfaces for each core and reference model? …
- No… we need standards…

Why do we need standard interfaces?

- There are many blocks required in DV solutions
- They all have different interfaces and these interfaces need defining
- They may come from different developers / suppliers
- They may be used in different projects and processor configurations / generations

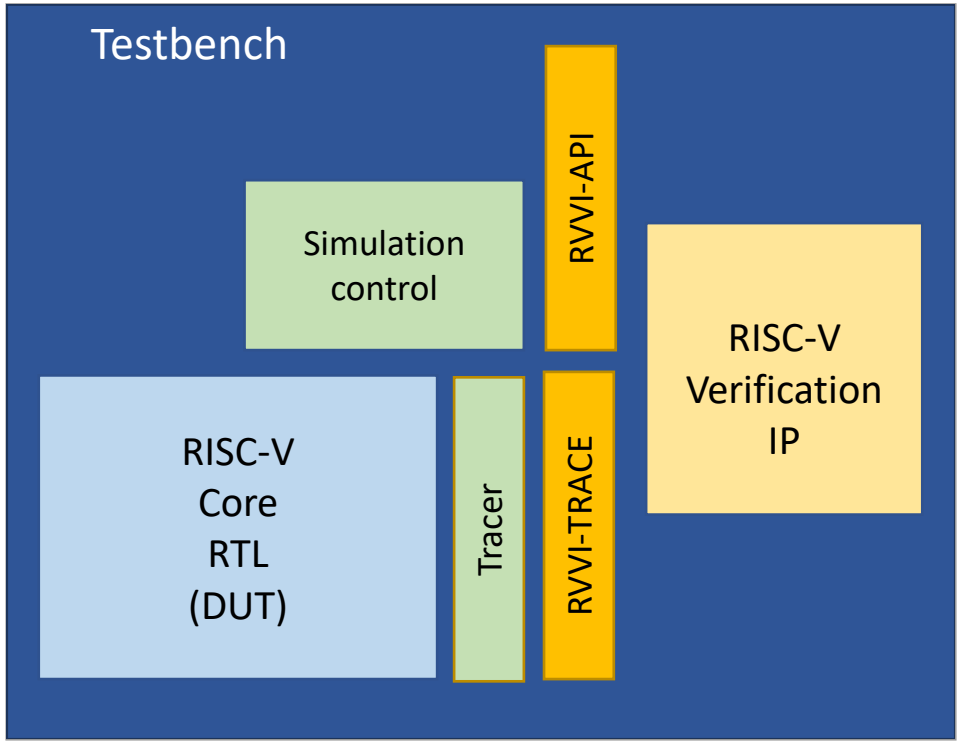
Why do we need standard interfaces?

- There are many blocks required in DV solutions
- They all have different interfaces and these interfaces need defining
- They may come from different developers / suppliers
- They may be used in different projects and processor configurations / generations

- Goals when developing standard interfaces:
  - re-use
  - common ways to do things
  - quick start-up
  - efficiency

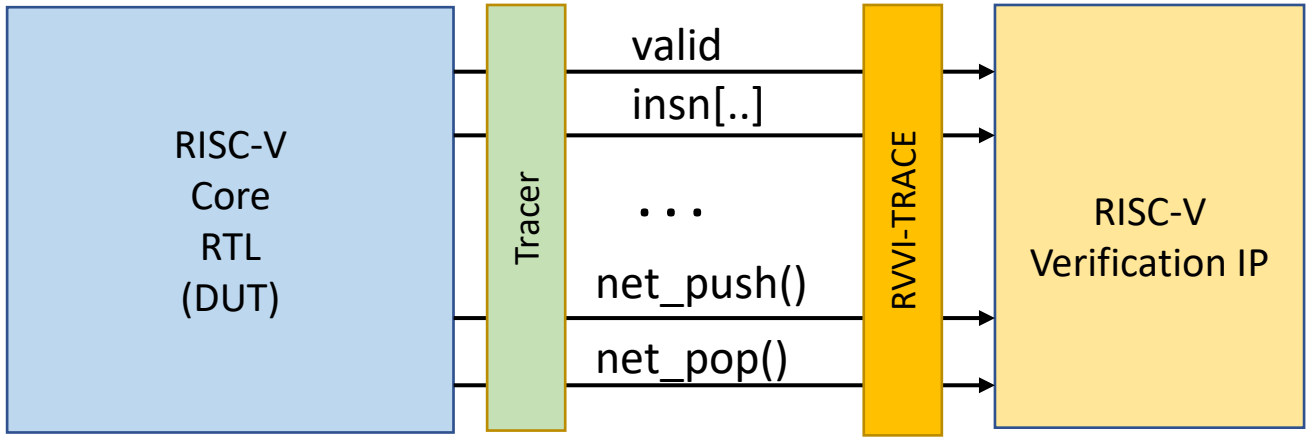
RVVI: the open standard RISC-V Verification Interface

- RVVI = RISC-V Verification Interface
  - https://github.com/riscv-verification/RVVI
- Work has evolved over 2+ years
  - Imperas, EM Micro, SiLabs, OpenHW and others
- Standardize communication between testbench and RISC-V VIP
- Two (main) parts:
  - **RVVI-TRACE**: signal level interface to RISC-V DUT
  - **RVVI-API**: function level interface to RISC-V VIP



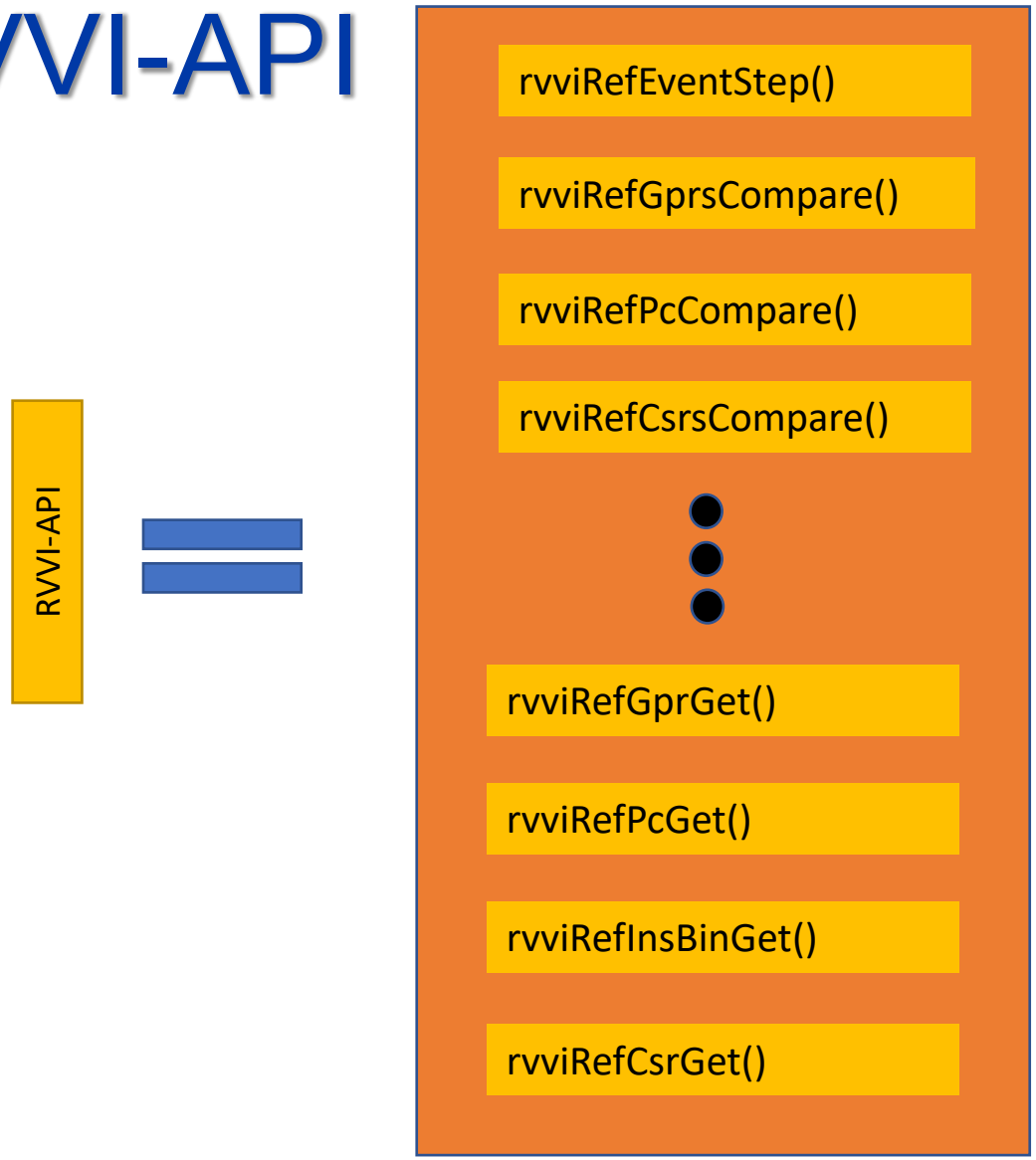
RVVI-TRACE

- 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 fully asynchronous verification methodology
- C and SystemVerilog versions available

https://github.com/riscv-verification/RVVI/blob/main/include/host/rvvi/rvvi-api.h

Why RVVI?

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

Adopting RVVI

- Upside to adoption
  - Potentially make use of other tools / code
  - Benefit from experience of others
- What tools / technologies can potentially be (re)used
  - Encapsulation of reference models, verification IP
  - Test benches & test bench components (inc. onward connection to reference models)
  - Functional coverage
  - Log file writer

RVVI Usage

- Usage:
  - testbench interfaces for components such as loggers, functional coverage, scoreboards
  - RTL tracer interfaces for processor cores with functionality being: in-order, out-of-order, single-issue, multi-issue, single-hart, multi-hart, sync and async nets, debug mode events and those related to advanced ISA extensions
  - encapsulation, control, and introspection interfaces for reference models
  - interfaces and functionality definitions of testbench virtual peripherals & event generators (wip)

- “Make simulation based RISC-V processor verification projects more efficient by helping provide open interface standards so that verification components can be used and reused efficiently”

RVVI (status: Nov. 2022)

- Many users are using RVVI…
  - For example within OpenHW: CV32E40X, CV32E40S, CV32E40P, and soon CV32E20

Public statements:

"An open verification standard such as RVVI provides the **essential framework and guidelines** to configure the test environment for RISC-V and allows the flexibility necessary to address all aspects of a modern processor yet maintain a common base that allows verification IP reuse across projects."

- Melaine Facon, Director of **Codasip**'s French Design Centre

"The verification requirements to achieve the ASIL D safety requirement level of ISO 26262 with a processor-based design are extensive, however verification IP reuse through standards such as RVVI help **improve efficiency and achieve time to market schedules** with all the design innovations that RISC-V enables."

- Hideki Sugimoto, CTO of **NSITEXE**, Inc., a group company of DENSO Corporation

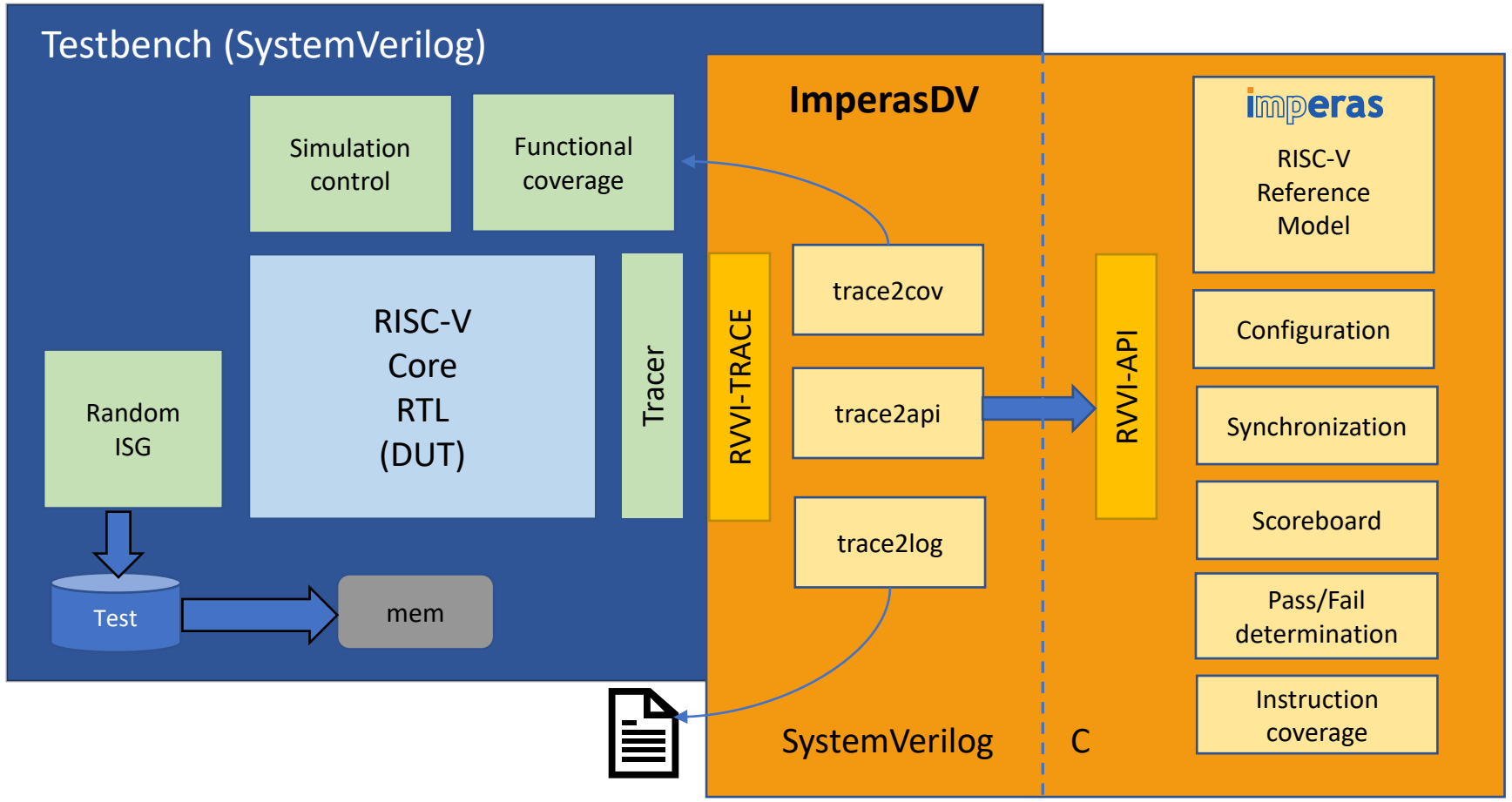
"Verification standards such as RVVI provide a **solid foundation that supports all RISC-V adopters**, from basic embedded cores through to complex application processors with multi-cluster, multi-core, multi-threading and out-of-order pipelines."

- Itai Yarom, VP of Sales and Marketing at **MIPS**

"Adopting RVVI virtual peripherals **provides additional flexibility and efficiency** for our flagship verification product STING to target asynchronous event verification, which is essential for quality RISC-V processor functional design verification."

- Shubhodeep Roy Choudhury, Managing Director & Co-founder, **Valtrix**

An example of a commercial RISC-V VIP from Imperas that uses RVVI



- Open standard
- RVVI-TRACE
  - Core tracer to test bench
  - SystemVerilog interface
- RVVI-API
  - Test bench to DV VIP subsystem including reference model
  - C/C++, SystemVerilog DPI

Works with SystemVerilog simulators: Cadence, Synopsys, Siemens EDA

Summary

- RISC-V is dramatically moving the processor DV task from the traditional mainstream IP providers to all SoC developers
- This represents the biggest migration in responsibility for verification in the history of the semiconductor industry
- RVVI enables verification IP reuse and engineering efficiency via open standards in RISC-V processor DV