# The continuum of RISC-V Compatibility testing and Verification testing

Simon Davidmann (simond@imperas.com)
Chair, Verification Task Group, OpenHW Group
CEO, Imperas
Allen Baum (allen.baum@esperantotech.com)
Chair, RISC-V Architectural Compatibility Tests Working Group
Esperanto

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

Abstract:

- RISC-V International develops and maintains the RISC-V ISA specification and provides basic architectural to confirm if users have read the specification
- OpenHW Group develops and maintains open source RISC-V processor cores and develops a unified open source testbench for all cores and specific tests to verify the cores to industrial quality levels
- There is a continuum of testing
  - Compatibility testing ensures you have met the ISA specification
  - Verification ensures you have implemented what you have specified for your design
- This talk discusses where compatibility testing starts and stops and where it differs from full design verification. The talk also covers the different technologies and tools that are required for these different activities
- Also the new projects in OpenHW Group are introduced that are moving the RISC-V Verification open ecosystem forward in the areas of Functional Coverage, Verification Standards, and Core quality with quantitative measurement with an explanation of how they are used for open source as well as commercial projects
- An example is made of the new open standard RISC-V Verification Interface (RVVI) Virtual Verification Peripheral definitions (VVP) and how they can be used in compatibility testing of privilege mode items such as asynchronous interrupts and debug mode requests.
- Key Point:
  - The RISC-V International Architecture Compatibility Tests (ACTs) are not a full processor verification plan

## Agenda

- Compatibility testing is not full design verification
- Different technologies and tools are required
- Moving the RISC-V Verification ecosystem forward with quantitative measurement
- New open standard RISC-V Verification Interface (RVVI) Virtual Verification Peripheral definitions (VVP) and how it can be used in compliance testing of privilege mode items

# Industry needs compatibility

- Device compatibility is absolutely critical in steering the ecosystem in a consistent direction, which ultimately allows the industry to benefit from shared investments
- There is a balance to strike in compatibility testing as sometimes it can be too little, leading to problems with ecosystem compatibility, or too complex leading to unnecessary cost and time
- Compatibility testing needs to focus on the ISA itself
  - this does introduce many new methodology challenges, especially since different cores can have different microarchitectures that will affect the timing of data

RISC-V® Summit

# The continuum of RISC-V Compatibility and Verification testing

OpenHW Group — Proven Processor IP
imperas

Simon Davidmann (simond@imperas.com)
Chair, Verification Task Group, OpenHW Group
CEO, Imperas

RISC-V®
esperanto.ai

Allen Baum (allen.baum@esperantotech.com)
Chair, RISC-V Architectural Compatibility Tests Working Group
Esperanto

# Why are there Architectural Compatibility Tests?

- ACTs are a subset of design verification, so why do we need them?
- The success of the RISC-V ecosystem depends on the ability to run software written for a RISC-V architecture on any implementation that claims to be RISC-V compatible
- The ACTs are intended to demonstrate that an implementation is Instruction Set Architect (ISA) compatible, and will run that software correctly

# What Architectural Compatibility Tests won't test

- Architectural Compatibility Tests test…. !Architecture!
- They do not test microarchitecture, e.g.
  - Speculation Branch target buffers
  - TLBs Caches
- They don't test that unimplemented extensions/options actually aren't
  - no "negative" testing
- This does NOT mean that the implementation is bug free!
  - And that's why you need DV as well.
- Note that standard distros will have other non-ISA requirements, e.g
  - extensions that must be supported,
  - specific support of various architectural options, and
  - often non-ISA support (e.g. interrupt controllers)

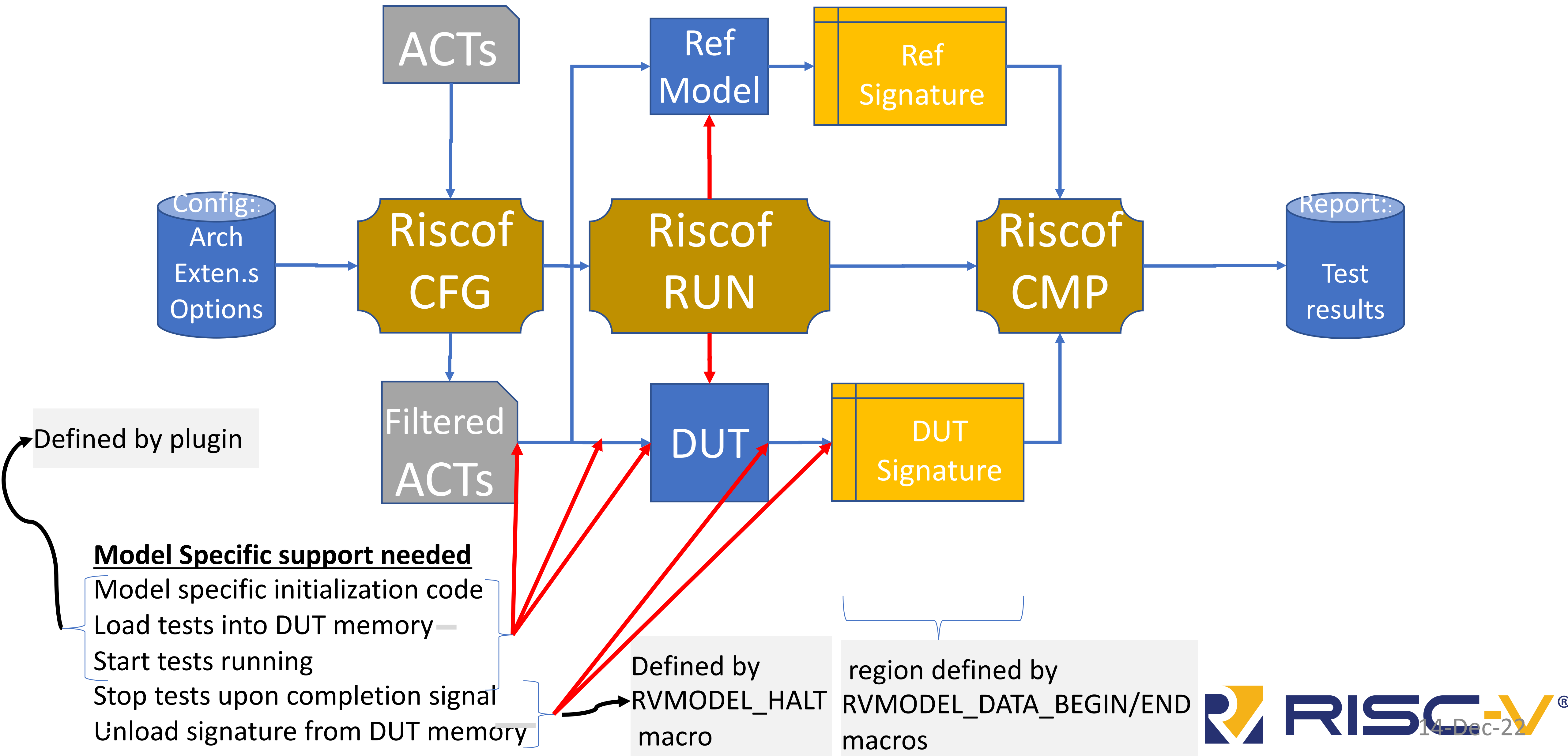
ACTs, however, are intended to support all legal combinations of architectural options, not just those prescribed by a standard distro, or a specific profile.

# The TAO of ACTs

The Architectural Compatibility tests show that

- An Implementor has read the RISC-V Architectural Spec
  - The tests are intended to show that the spec was studied, basic instruction formats were followed, and instructions can be executed.
- An implementor has understood the RISC-V Architectural Spec
  - The tests are intended to show that the spec has not been misunderstood.
  - If it has been  misunderstood, then DV would show that it correctly executes but with incorrect behavior – a false positive. So, a cross-check of DV behavior
- An implementor has implemented the RISC-V Architectural Spec
  - Even if the spec is understood, the test are intended to show that corner case shortcuts that violate the spec were not taken – because OSes, Tools, and Applications just might fall into the cases that were shortcut.

# How does ACT framework run?



Defined by plugin

**Model Specific support needed**
Model specific initialization code
Load tests into DUT memory
Start tests running
Stop tests upon completion signal
Unload signature from DUT memory

Defined by RVMODEL_HALT macro

region defined by RVMODEL_DATA_BEGIN/END macros

# Summary: Architectural Compatibility Tests

- Architectural Compatibility Tests – ACT
  - The ACTs are intended to demonstrate that an implementation is Instruction Set Architect (ISA) compatible, and will run that software correctly
    - With the assumption of exhaustive functional verification having already been achieved
  - ACTs Are a subset of design verification
  - Passing the ACTs does NOT mean that the implementation is bug free!

# The continuum of RISC-V Compatibility and Verification testing

Simon Davidmann (simond@imperas.com)
Chair, Verification Task Group, OpenHW Group
CEO, Imperas

Allen Baum (allen.baum@esperantotech.com)
Chair, RISC-V Architectural Compatibility Tests Working Group
Esperanto

# So what approaches are there for ‘verification’ of new processors

- Does a program run? – ‘hello world’ tests
- Is there simple correct computation? – ‘self checking tests’

Simple tests

- Signature checking – ‘post simulation signature dump compares’
- Trace log checking – ‘post simulation trace file compare’

Compliance

- Simple step and compare co-simulation – ‘instruction retire compare’
- Advanced, e.g. commercial solutions – ‘e.g. ImperasDV’

Verification

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

# Difference in what is to be investigated / measured

Compatibility Testing

- Un-priv. ISA architecture
- Priv. ISA ISA architecture
- Instructions
- CSRs
- Execution modes
- Memory protection, PMP, MMU
- Instruction/data alignment
- Virtualization & Hypervisor
- Synchronous events/exceptions
- Asynchronous events/Interrupts
- Debug mode
- CSR WARL
- Configuration/implementation options
- “constrained unpredictable” options

Verification is all compatibility topics plus:

- Data related – e.g. FPU
- Pipeline related – e.g. hazards
- Memory hierarchy related– e.g. coherency
- Dynamic affects – e.g. async. event races
- Resource contention related
- Side affects/cross related
- Illegal conditions
- Failure modes
- Custom instruction and register extensions
- Performance aspects
- …

# What are the two activities trying to show

- Compatibility Testing
  - Specification has been read
  - Operations work
  - Software from various vendors should 'work out of the box'

- Verification
  - Operation has been tested in all modes and conditions
  - Operations have been exhaustively tested
  - Corner case operation has been explored
  - Combinations of operations have been checked
  - Operations/behaviour compared with reference behaviour

# What are the differences

- RISC-V International Compatibility (~10**7 cycles)
  - Support all RISC-V ISA extensions, options
  - Directed assembler tests
  - Signature based file compare
- RISC-V International requirements:
  - Not require use of commercial tools
  - Not require use of commercial reference model
  - Be free of charge
  - Be available as open source:
    - Tests
    - Models
    - Simulators
    - Framework
    - Debuggers
    - Compilers
    - Directed test generator
    - Coverage tooling
    - Other tools…

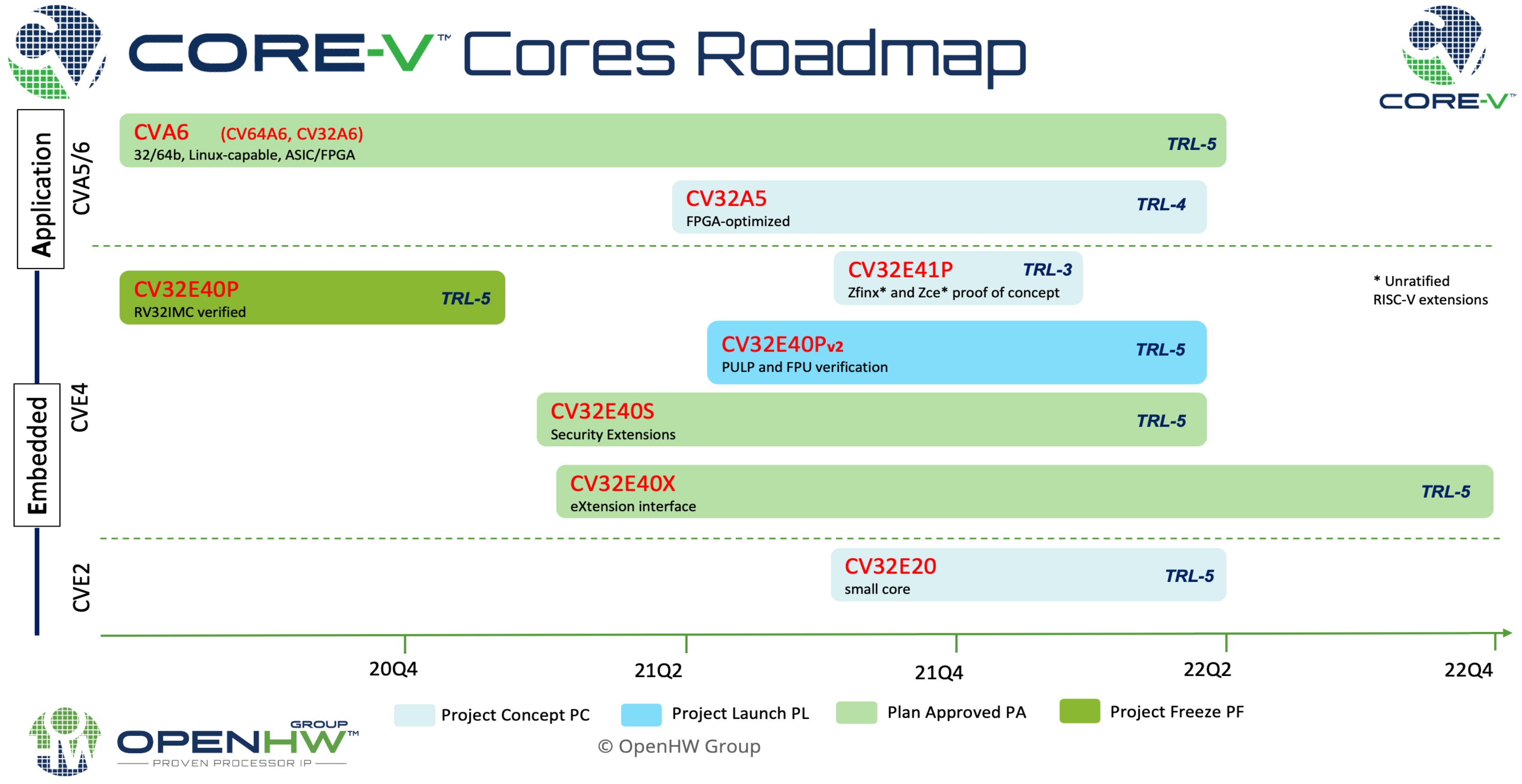
- Verification (>10**15 cycles)
  - Focus on specific core
  - Use any available tool to get job done
    - Proprietary, commercial, open source
  - Commercial quality tooling
  - Best in class tooling, Verification IP (VIP)
    - e.g. industrial grade reference models
    - e.g. async-lock-step-compare VIP
  - Instruction stream generators
  - Large experienced team
  - Utilize commercial verification ecosystem

# What are open source projects like OpenHW doing for verification

Agenda:

- Cores roadmap
- Testbench architecture
- Evolving standards

# OpenHW CORE-V cores – roadmap (c. 2020)



=> Need common verification solutions across core range

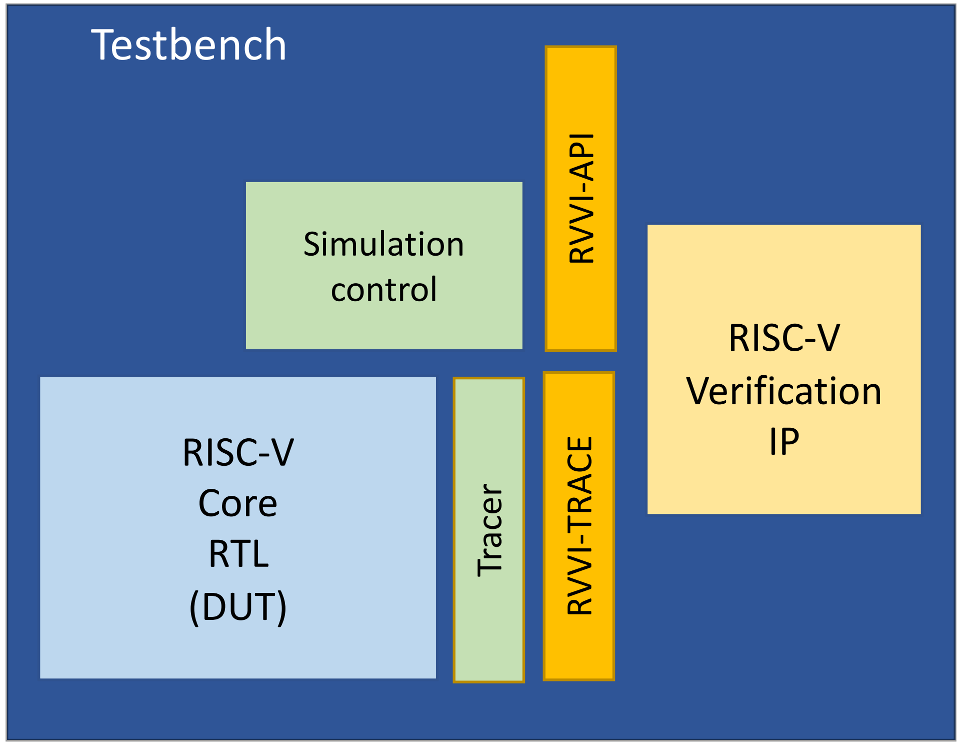
# OpenHW Projects
# Advanced RISC-V Verification Methodologies

- **Methodologies**
  - improving the capabilities and available methodologies available for verification environments
- **Promotion**
  - educating / informing processor verification teams of choices and techniques available across the RISCV community and verification ecosystem (for example tutorials and videos)
- **Testbench Quality***
  - developing quality measurement of test benches - so quality of cores can be predicted (for example defining fault models and tests and relating those to TRL levels)
- **Standards****
  - defining and implementing evolving interface standards for test bench components to enable better test bench component reuse and potentially stimulate availability of compatible VIPs
- **Functional Coverage***
  - developing open-source VIPs that can be used for many different core configurations/implementations
- **SoC Integration***
  - consider requirements and solutions for SoC core integration verification (for example cache coherency with un-core components)
- **Roadmap**
  - ongoing roadmap to accommodate new innovations in RISC-V designs, new ratified extensions, and new tool developments

*active project, **discussed in next slides…

# Use RVVI open standard for testbench, VIP re-use, efficiency across cores

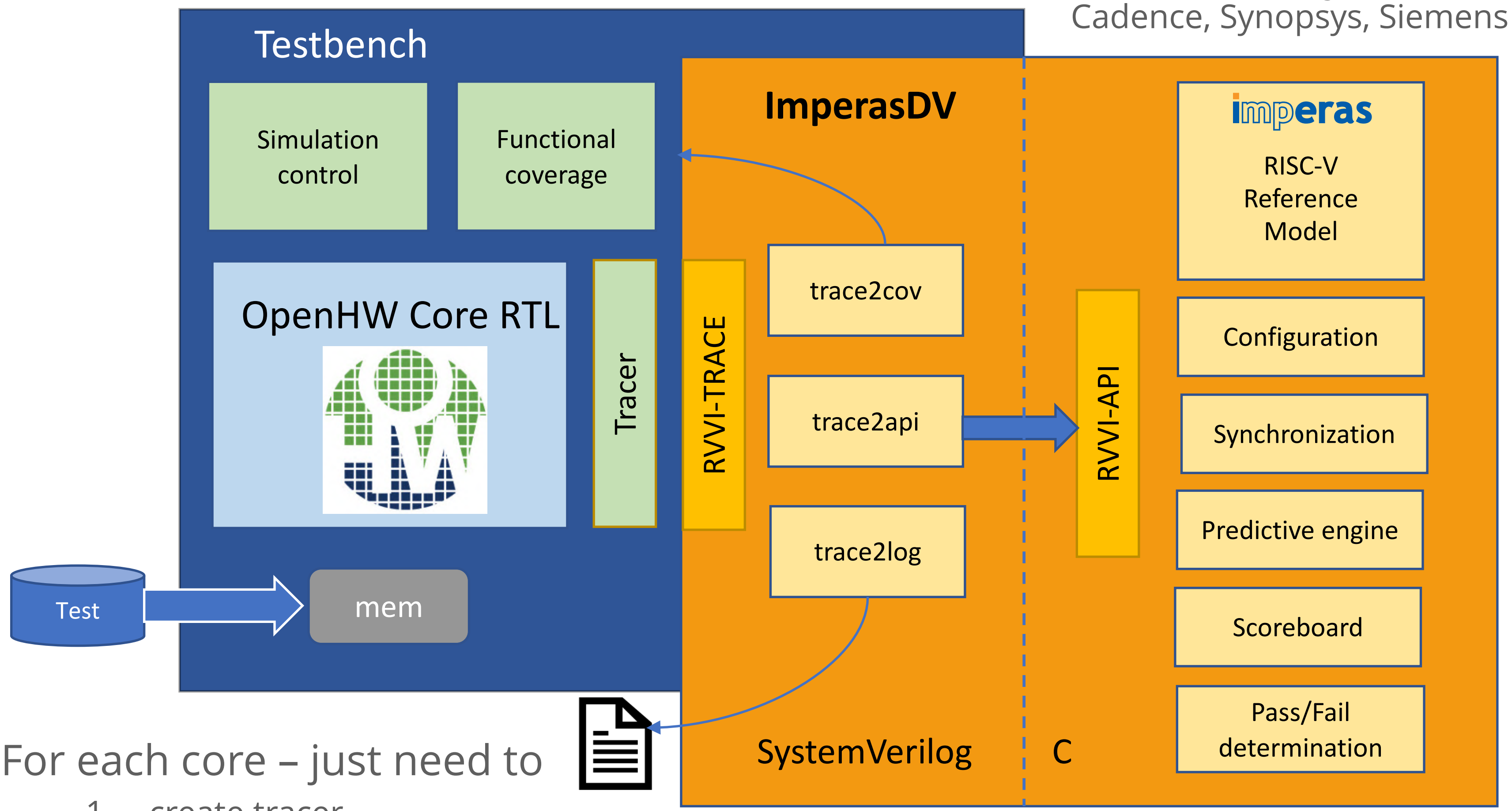
- RVVI = RISC-V Verification Interface
  - https://github.com/riscv-verification/RVVI
- Work has evolved over 2+ years
  - Imperas, EM Micro, SiLabs, OpenHW and others
- Standardize communication between testbench and RISC-V VIP
- Two (main) parts:
  - **RVVI-TRACE**: signal level interface to RISC-V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



=> "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"

# OpenHW use RISC-V VIP from Imperas that uses RVVI – works for any core

Works with SystemVerilog simulators:
Cadence, Synopsys, Siemens EDA



- 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

For each core – just need to

1. create tracer
2. configure Imperas reference/VIP

## RVVI (status: Dec. 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**

## RVVI-VVP (new: under development) Virtual Verification Peripherals

- Agenda
  - What are Virtual Verification Peripherals
  - Why do we need them

## What is a Virtual Verification Peripheral?

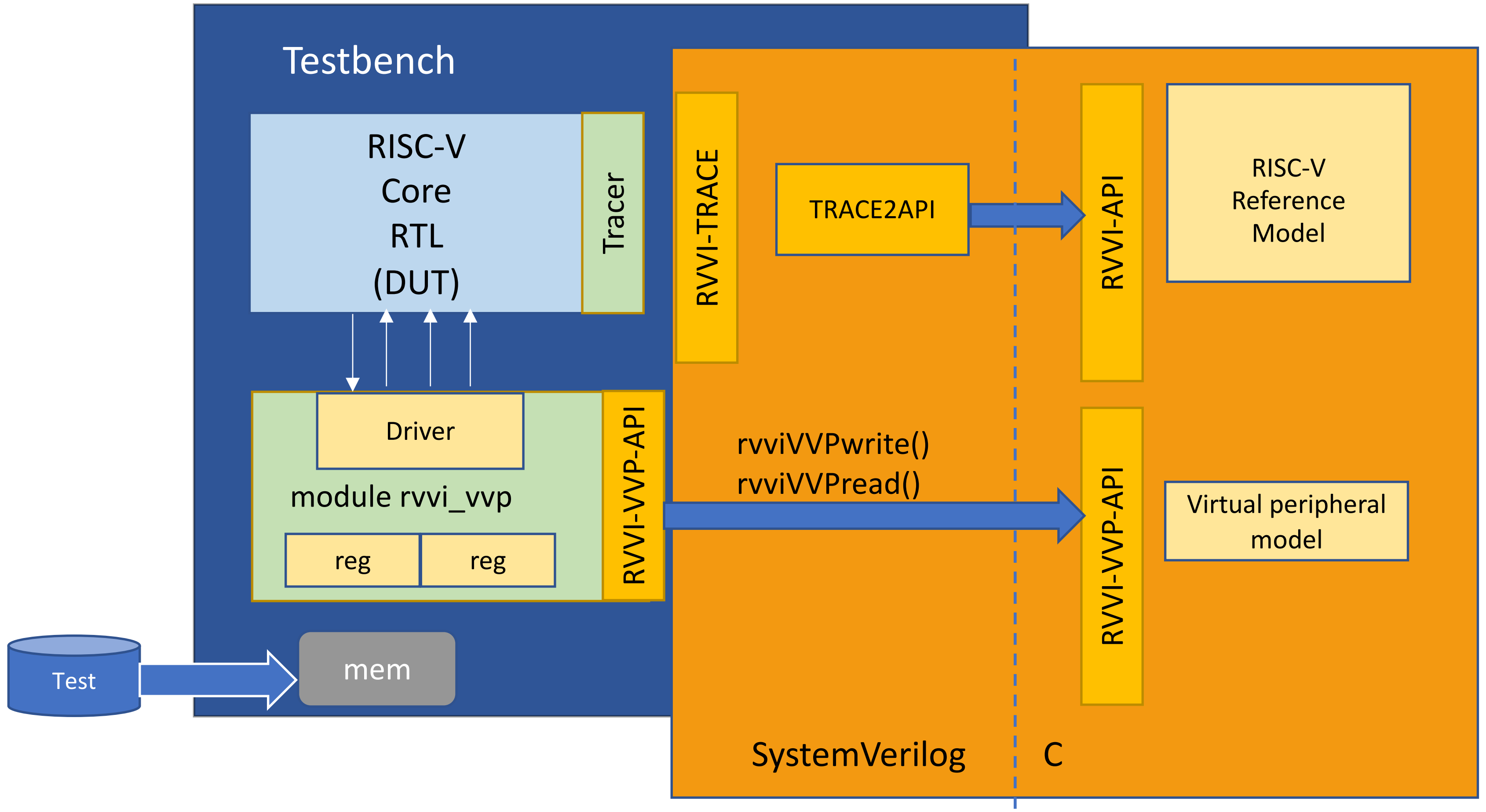
- Memory-mapped testbench component designed to help with compliance & verification of a RISC-V core
- It is not intended to model a real device or piece of hardware
- Will be a model in ISSs, testbenches, VIPs
- Examples used in OpenHW:
  - VVPs for RISC-V compliance
    - Virtual printer, signature file writer, status flags
  - VVPs for real peripheral interfaces
    - Interrupt timer control, debug control, instruction memory stall control
  - VVPs for “convenience functions”
    - Address range check, random number generator, cycle counter

# Why do we need VVPs?

- To be able to run RISC-V compliance tests (e.g. with interrupts)
- To better co-ordinate stimulus on peripheral interfaces with the program running on the core. E.g. interrupts, debug, instruction memory stall(?)
  - To enable better coverage of interactions between instructions and peripheral events
- To make it easier to reproduce random failures
  - The randomness could be contained within the test program

- Without a standard and examples to follow, users will invent their own methods of stimulating peripheral interfaces
  - This can lead to debug issues if the stimulus does not come from the test program
  - Harder for users to achieve thorough functional coverage without the ability to coordinate peripheral events with instruction execution

# VVP used in testbench, ISS, and VIPs



- An open standard for VVPs enable efficient re-use of test programs for compliance and verification of priv. mode sections of RISC-V ISA

# Summary

- RISC-V Compatibility testing & Verification are different

- RISC-V Compatibility testing is not verification
- RISC-V Compatibility testing is necessary for quality products

- RISC-V Compatibility testing is necessary but not sufficient to ship a quality product

- There is synergy and compatibility
- You need both

- Standards are evolving to assist with both

# RISC-V Summit - Tutorials

- Thank you…

- Compatibility testing tutorial Thursday 15th Dec. 9:00am (**Incore**)
  - Running the Architectural Compatibility Tests on your Model: Theory and Practice
    Neel Gala & Pawan Kumar, Incore Semiconductor
- Verification tutorial Thursday 15th Dec. 10:30am (**Imperas**)
  - Choosing Appropriate Verification Techniques for Desired RISC-V Processor Quality
    Aimee Sutton & Lee Moore, Imperas Software

RISC-V® Summit

# The continuum of RISC-V Compatibility and Verification testing

OPENHW GROUP® — PROVEN PROCESSOR IP —

imperas

Simon Davidmann (simond@imperas.com)

Chair, Verification Task Group, OpenHW Group

CEO, Imperas

RISC-V®

esperanto.ai

Allen Baum (allen.baum@esperantotech.com)

Chair, RISC-V Architectural Compatibility Tests Working Group

Esperanto