



# The evolution of RISC-V and the adoption of new open verification standards

Jon Taylor

Imperas Software, January 2023



# Agenda

- Imperas introduction
- RISC-V design verification (DV) challenges
- Standardization: RISC-V Verification Interface (RVVI)
- Summary



# Imperas and RISC-V



- Imperas founding team has background in Electronic Design Automation (EDA) tools, and FPGA and processor IP companies
- 2007: Imperas founding team saw the need for tools and methodology similar to EDA for software debug, test and analysis, based on software simulation
  - This required high quality models of the embedded processors
  - Self-funded the company, and built a business on virtual platform products serving Arm, MIPS, Renesas, etc. users
- In 2016, we realized the the RISC-V community would need RISC-V models for software development
- While there was, and still is a large demand for RISC-V models for software development, there is also a need for models of custom RISC-V cores, and a need for verification of RISC-V cores
- Imperas started working on RISC-V compliance, and then on processor verification, with RISC-V International and OpenHW Group, and customers such as Nvidia Networking, NSITEXE, MIPS, Cudasip, Silicon Labs, and more
- Now Imperas RISC-V processor models are used in almost every RISC-V project

# OVP Library of RISC-V Fast Processor Models

- Available at <https://ovpworld.org>
  - Generic or envelope models of RV32/64 IMAFDCEVBHKP M/S/U privilege modes
  - Models of processor IP vendors cores: Andes, Cudasip, Imagination, Intel, MIPS, OpenHW, SiFive,
  - Custom models for users building their own RISC-V processors
    - Support for custom instructions
- Models are built using Test Driven Development (TDD) methodology
  - Tests are built at the same time as features are added
  - Continuous Integration (CI) test flow used
  - > 15,000 directed tests for models + simulator
  - Additional testing by processor IP vendors to validate models



“The Imperas virtual platform solutions for software development, debug and test, along with their open-source models, will help accelerate SoC and embedded software development for our customers.”

*Charlie Hong-Men Su, Ph.D., Andes Technology CTO*



# High Quality RISC-V Models Are Required for RISC-V Success



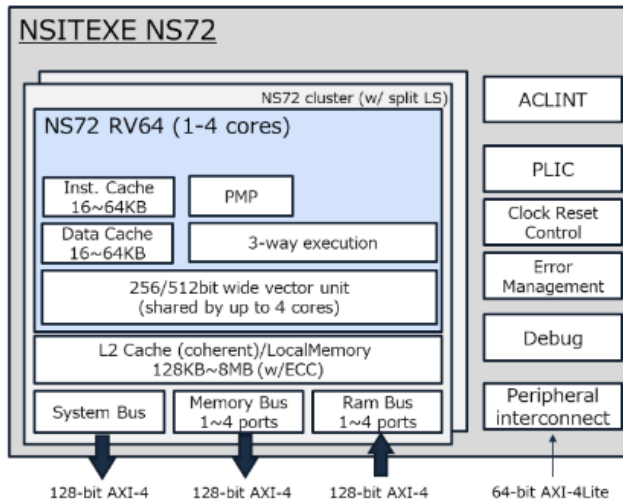
- Use cases
  - Architecture analysis, including (especially) custom instructions
  - Software development, debug and test
  - Processor and SoC verification
- Anyone can build an Instruction Set Simulator (ISS)
  - ISS is viewed as a commodity item
  - “Everyone” has built an ISS
- More than an ISS is needed

“For the automotive market our customers expect the highest standards of quality and design assurance. NSITEXE selected the Imperas Vector Extensions Compliance test cases and RISC-V Reference Model as a foundation for our simulation-based design verification (DV) plans.”

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

# Simulation at scale

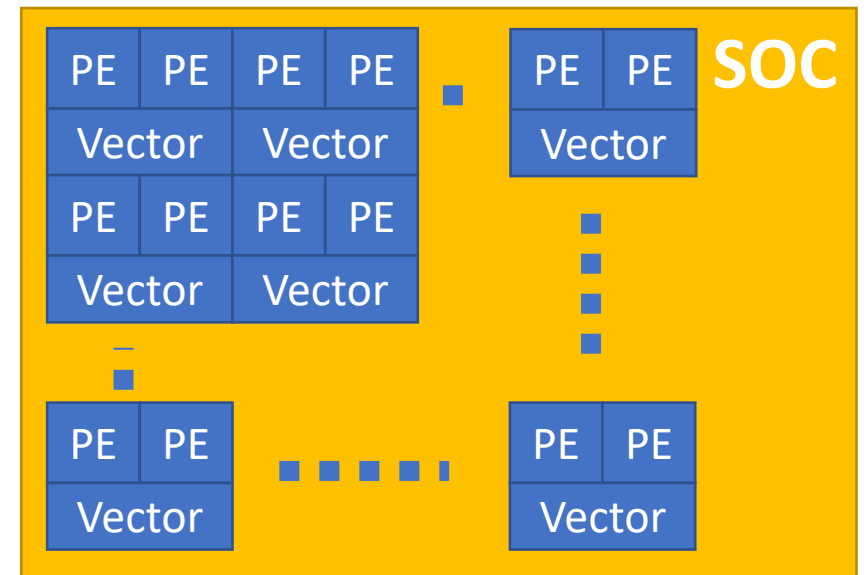
- Having a fast single thread, single core simulation isn't enough for HPC
- Imperas simulators scale to thousands of harts, wide vectors and more
- Customer A: NSITEXE



## NS72 highlight features

- Efficient Out of Order Superscalar w/ hardware-multithreading
  - 10+-stage pipeline
  - 128-bit instruction fetch
  - 3-wide instruction decode & renaming
  - 2 hw-threads per core
- High Performance Vector Processing Unit (Optional)
  - 256/512-bit wide execution pipelines
  - 4 pipelines (Arithmetic x2, Special x1, Store x1)
  - Support RVV1.0 full features
  - Vector Out-of-Order execution
- FuSa
  - ECC memory/bus protection
  - Split Lockstep mechanism
  - Physical Memory Protection
- High Performance Vector Processing Unit
  - 128-bit AXI-4 system bus
  - 128-bit AXI-4 memory bus (up to 4)
  - 128-bit AXI-4 ram bus slave (up to 4)

## Customer B: Hyperscaler

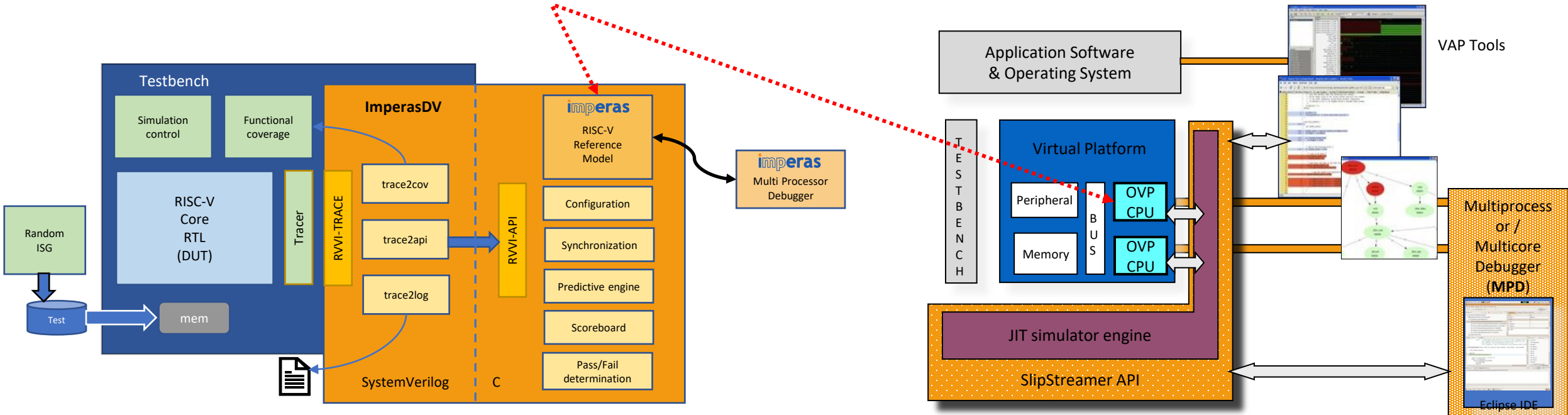


<https://www.imperas.com/articles/nsitexe-selects-imperas-risc-v-and-vectors-reference-model>

# Imperas OVP RISC-V Models are used for Processor DV & SW Development



- Base model implements RISC-V specification in full
- Fully user configurable to select which ISA extensions/versions
- Imperas provides methodology to easily extend base model



# Linear scaling with more cores



- Fast single core performance

```
Info -----  
Info CPU 'iss/cpu0' STATISTICS  
Info   Type                : riscv (RV64GC)  
Info   Nominal MIPS         : 100  
Info   Final program counter : 0x1bece  
Info   Simulated instructions: 5,600,066,547  
Info   Simulated MIPS       : 3135.9  
Info -----
```

(Dhrystone speed test of a single core)

- Linear performance with many cores

```
IMPERAS Instruction Set Simulator (ISS)  
Platform has 1024 separate processors  
...  
Info CPU 'iss/cpu<x>' STATISTICS  
Info   Type                : riscv (RV64GC)  
Info   Nominal MIPS         : 100  
Info   Final program counter : 0x1bece  
Info   Simulated instructions: 5,600,066,547  
Info   Simulated MIPS       : 2.5  
Info -----  
...  
Info TOTAL  
Info   Simulated instructions: 5,734,468,144,128  
Info   Simulated MIPS       : 2576.3  
Info -----
```

(Dhrystone test with 1024 cores)



# Host parallelisation



- Single thread host

IMPERAS Instruction Set Simulator (ISS)  
Platform has 1024 separate processors

```
...
Info CPU 'iss/cpu<x>' STATISTICS
Info   Type                : riscv (RV64GC)
Info   Nominal MIPS        : 100
Info   Final program counter : 0x1bece
Info   Simulated instructions: 5,600,066,547
Info   Simulated MIPS      : 2.5
Info -----
...
Info TOTAL
Info   Simulated instructions: 5,734,468,144,128
Info   Simulated MIPS        : 2576.3
Info -----
```

(Dhrystone test with 1024 cores)

- 4-core host (-parallelmax)

IMPERAS Instruction Set Simulator (ISS)  
Platform has 1024 separate processors

```
Info -----
Info CPU 'iss/cpu<x>' STATISTICS
Info   Type                : riscv (RV64GC)
Info   Nominal MIPS        : 100
Info   Final program counter : 0x1bece
Info   Simulated instructions: 5,600,066,547
Info   Simulated MIPS      : 9.8
Info -----
...
Info -----
Info TOTAL
Info   Simulated instructions: 5,734,468,144,128
Info   Simulated MIPS        : 10004.9
Info -----
```



# Agenda

- Imperas introduction
- **RISC-V design verification (DV) challenges**
- Standardization: RISC-V Verification Interface (RVVI)
- Summary



# 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
- ImperasDV is the first commercial product to support RISC-V CPU DV

# Agenda

- Imperas introduction
- RISC-V design verification (DV) challenges
- **Standardization: RISC-V Verification Interface (RVVI)**
- Summary



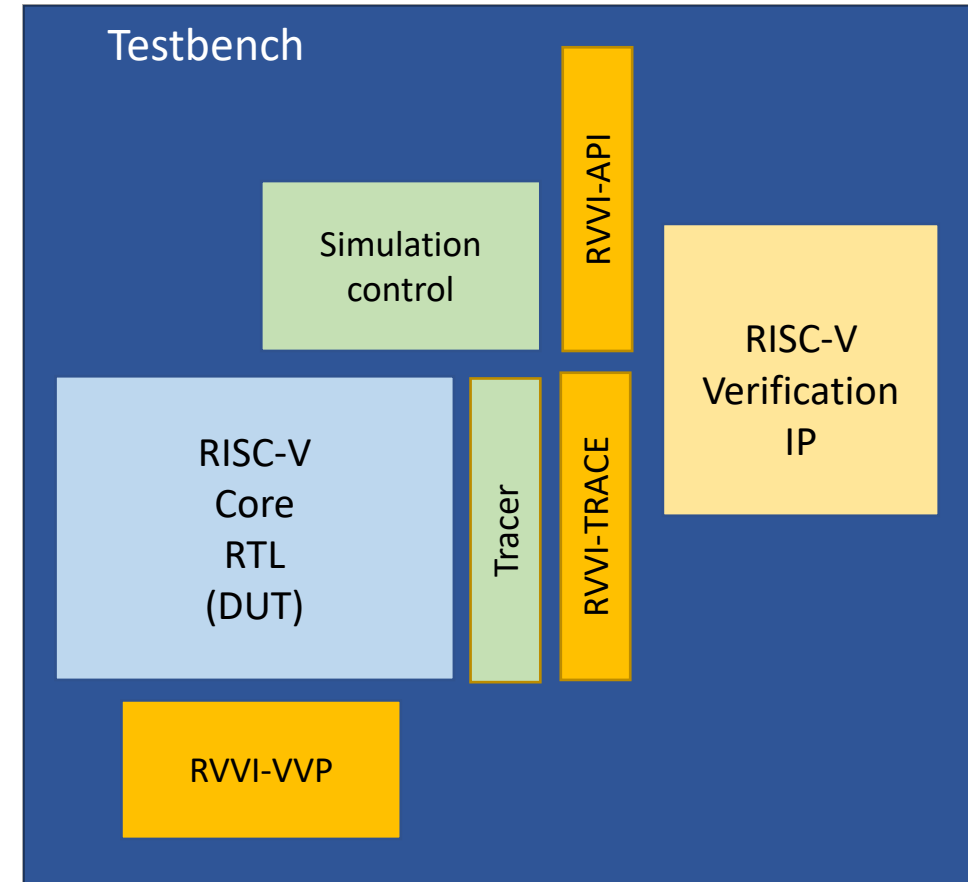
# Standardization



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

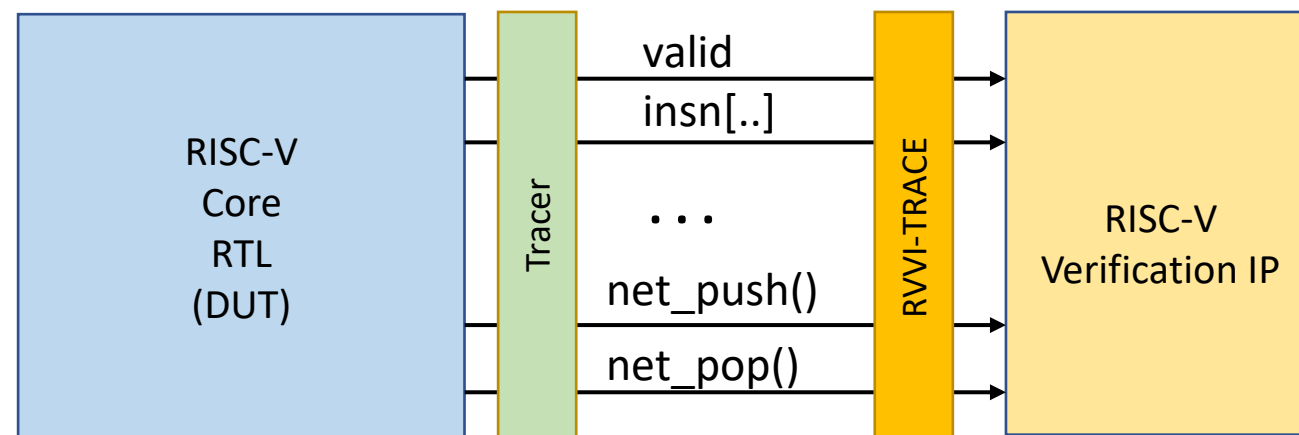
# Standardization: RVVI

- 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



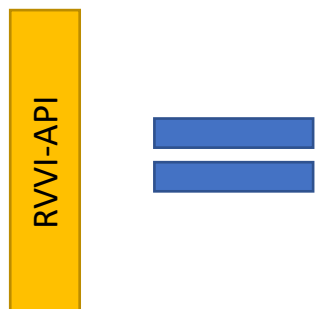
# RVVI-TRACE

- 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



- rvviRefEventStep()
- rvviRefGprsCompare()
- rvviRefPcCompare()
- rvviRefCsrsCompare()
- ⋮
- rvviRefGprGet()
- rvviRefPcGet()
- rvviRefInsBinGet()
- rvviRefCsrGet()



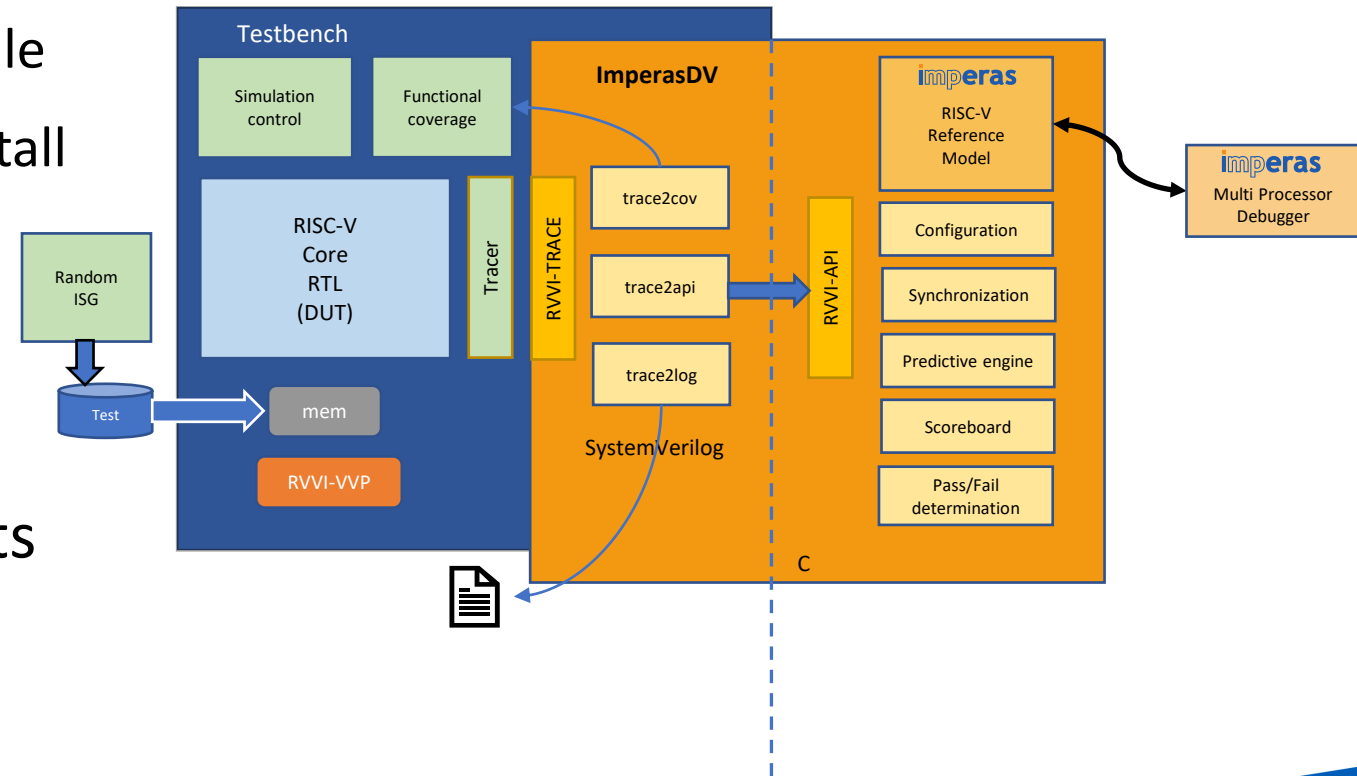
- Standard functions that RISC-V processor VIPs need to implement
- Supports a lockstep co-simulation methodology
- C and SystemVerilog versions available

<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 of privilege mode (inc interrupts)
- Work-in-progress



# Why a standard, 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
  - Lockstep co-simulation comparison, asynchronous events

# 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

# Comparison of methodologies



	Log compare	Sync lockstep co-simulation	Async lockstep co-simulation
Instruction trace	Yes	Yes	Yes
Data trace	Yes	Yes	Yes
Runtime checking	No	Yes	Yes
Stop on error (to allow debug)	No	Yes	Yes
All processor state checked every cycle?	No	Yes	Yes
Data storage requirements	Large	Small	Small
Interrupts?	No	No	Yes
Debug mode	No	No	Yes
uArch details (eg PMU)	No	No	Yes
Multi-issue, OoO, multi-hart	No	No	Yes

# 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
- HPC adds the challenge of scale and complexity – not all tools can do this



**imperas**

**Thank you!**

Jon Taylor

[jont@imperas.com](mailto:jont@imperas.com)

[www.imperas.com](http://www.imperas.com)

[www.OVPworld.org](http://www.OVPworld.org)