RIS C-V

Big Numbers

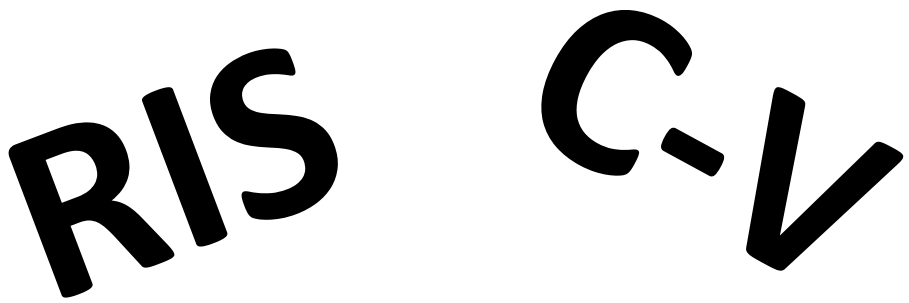
- 131,000,000 circumference of the earth in feet
- 100,000,000,000 the number of stars in the milky way
- 6,000,000,000,000 the number of miles in one light year
- 20,000,000,000,000 the total net worth in the world, in dollars (M2)

- 1,000,000,000,000,000 number of verification cycles Arm applies to one core

The RISC-V Disconnect

RISC-V Core User

Expects core quality to be the same as Arm

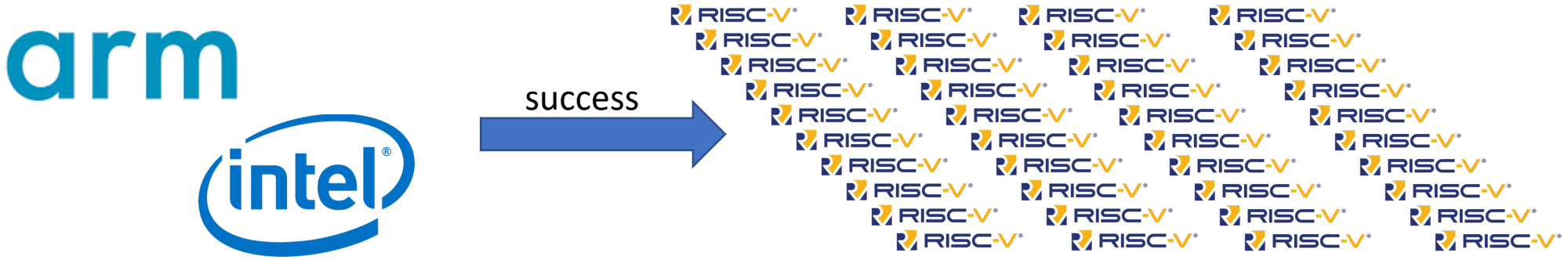


RISC-V Core Developer

Unlikely to have resources needed be able to develop all the technologies required to perform the same level of verification as Arm

Scaling RISC-V Verification

How can we scale in-depth, proprietary processor verification experience, tools and methodology across the RISC-V community?



Improving RISC-V processor quality with verification standards and advanced verification methodologies

Simon Davidmann, CEO, Imperas Software

13-December-2022

Ecosystems

- All electronic products are now defined by their SW
- Not just their operation, but their environments and lifecycles
  - Interconnected, cloud access, over air updates, improvements, …
- What used to be a challenge for devices, is now a systems problem

- Yes we need an ISA, and a focused team for its development
- Yes we need SW, and its ecosystem and partnerships

- But if the devices don't work as needed, then there is nothing…
- And if the cores/processors are not as expected… the systems fail

- So alongside ISA, and SW, we have a third huge challenge: verification

Putting processor verification into context….

1,000,000,000,000,000

The number of verification cycles Arm uses when verifying an Arm core

- SystemVerilog simulator executes 2,000 cycles / second
  => 15,000 SystemVerilog simulators running for 1 year
- HW emulator or FPGA runs at 1,000,000 cycles / second
  ⇒30 years of running needed…

- OK – so this is for a high performance OoO, MP, VM core (full apps processor)
  - Embedded processors will be an order of magnitude less…

How do others do it

- Intel, Arm
  - MIPS, PowerPC, SPARC, …

- Well they developed all needed tools/solutions – internally – proprietary – for their internal usage…

- They had 100s of people involved and spent $100Ms each year on it…

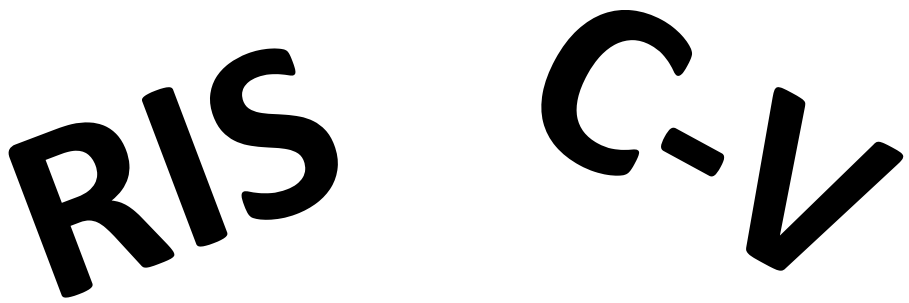
[and they have far less configuration and don't have custom instructions and other extensibility available with RISC-V… and they had the advantage of single source control of the RTL…]

What did they spend their $100Ms/year on…

- Tools
- Verification IP
- Models
- Methodology
- Infrastructure
- Tests
- [and also formal tools and correct-by-construction flows]

And what they did not buy in – they built internally… with 100s of people…

So we have to accept – RISC-V has a problem…



And that problem… is … verification

Some “facts”…

- Almost all ISAs are no longer used…
- Most SoC chips are fabricated first time with bugs (and cost $ to re-spin)
- Most semiconductor start-ups fail to return interesting $ to investors…

Life is already hard!

So what can we do in RISC-V

- In the RISC-V world, it is unlikely that one company can spend the $ or can hire the people to develop all they need…
  - [Arm relies on ISA / design royalty, Intel relies on silicon sale…]

- Partner / Collaborate in non-competitive areas
- Attract players into the verification ecosystem to develop needed solutions
- Build standards to facilitate re-use and efficiency

- If it does not differentiate your product offering / company
  - You can collaborate externally
  - You can license commercial tools

Imperas

- 2008 – developed world class processor modeling & simulation solutions for many ISAs for virtual prototyping and software development
  - A good, growing, and profitable business

- 2016 started looking at RISC-V
- 2018 RISC-V processor developers started using Imperas as reference for their hardware verification
- For last 4 years have been assisting companies with their RISC-V DV needs
- For last 3 years started working collaboratively with free and open source solutions
  - e.g. OpenHW Group open source highly verified industrial quality RISC-V cores
- For last 2 years working on RISC-V verification standards and advanced methodologies

So what have we learnt in last 4 years… There are many methodologies for 'verification' of new processors

imperas

- 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 – 'async-lock-step-compare'

Verification

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

RISC-V verification tooling

- Tools needed to create a good DV environment
  - Test bench / framework / Verification IP
  - Models
  - Coverage recording & reporting tooling
  - HDL Simulators
  - Tests
  - Test generators
  - Cross-compilers
  - Software debuggers
  - Formal tools
  - Regression testing / job run framework
  - …

RISC-V verification tooling

- Tools needed to create a good DV environment
  - Test bench / framework / Verification IP — core bespoke
  - Models — core bespoke
  - Coverage recording & reporting tooling — core bespoke
  - HDL Simulators — industry standards
  - Tests — core bespoke
  - Test generators — ISA bespoke
  - Cross-compilers — ISA bespoke
  - Software debuggers — ISA bespoke
  - Formal tools — ISA bespoke
  - Regression testing / job run framework — industry generic
  - …

RISC-V verification tooling

- Tools needed to create a good DV environment
  - Test bench / framework / Verification IP — core bespoke
  - Models — core bespoke
  - Coverage recording & reporting tooling — core bespoke
  - HDL Simulators — industry standards (SystemVerilog)
  - Tests — core bespoke
  - Test generators — ISA bespoke (buy in…, e.g. Valtrix)
  - Cross-compilers — ISA bespoke (GCC, LLVM projects)
  - Software debuggers — ISA bespoke (GCC, LLVM projects)
  - Formal tools — ISA bespoke
  - Regression testing / job run framework — industry generic (…)
  - …

RISC-V verification tooling

- Tools needed to create a good DV environment
  - Test bench / framework / Verification IP — core bespoke
  - Models — core bespoke
  - Coverage recording & reporting tooling — core bespoke
  - HDL Simulators — industry standards (SystemVerilog)
  - Tests — core bespoke
  - Test generators — ISA bespoke (buy in…, e.g. Valtrix)
  - Cross-compilers — ISA bespoke (GCC, LLVM projects)
  - Software debuggers — ISA bespoke (GCC, LLVM projects)
  - Formal tools — ISA bespoke (SystemVerilog)
  - Regression testing / job run framework — industry generic (…)
  - …

RISC-V verification tooling

- Tools needed to create a good DV environment
  - Test bench / framework / Verification IP — core bespoke
  - Models — core bespoke
  - Coverage recording & reporting tooling — core bespoke
  - HDL Simulators — industry standards (SystemVerilog)
  - Tests — core bespoke (standard .S etc.)
  - Test generators — ISA bespoke (buy in…, e.g. Valtrix)
  - Cross-compilers — ISA bespoke (GCC, LLVM projects)
  - Software debuggers — ISA bespoke (GCC, LLVM projects)
  - Formal tools — ISA bespoke (SystemVerilog)
  - Regression testing / job run framework — industry generic (…)
  - …

RISC-V verification tooling

- Tools needed to create a good DV environment – for specific ISA extensions, customizations, and implementations
  - Test bench / framework / Verification IP
  - Models
  - Coverage recording & reporting tooling

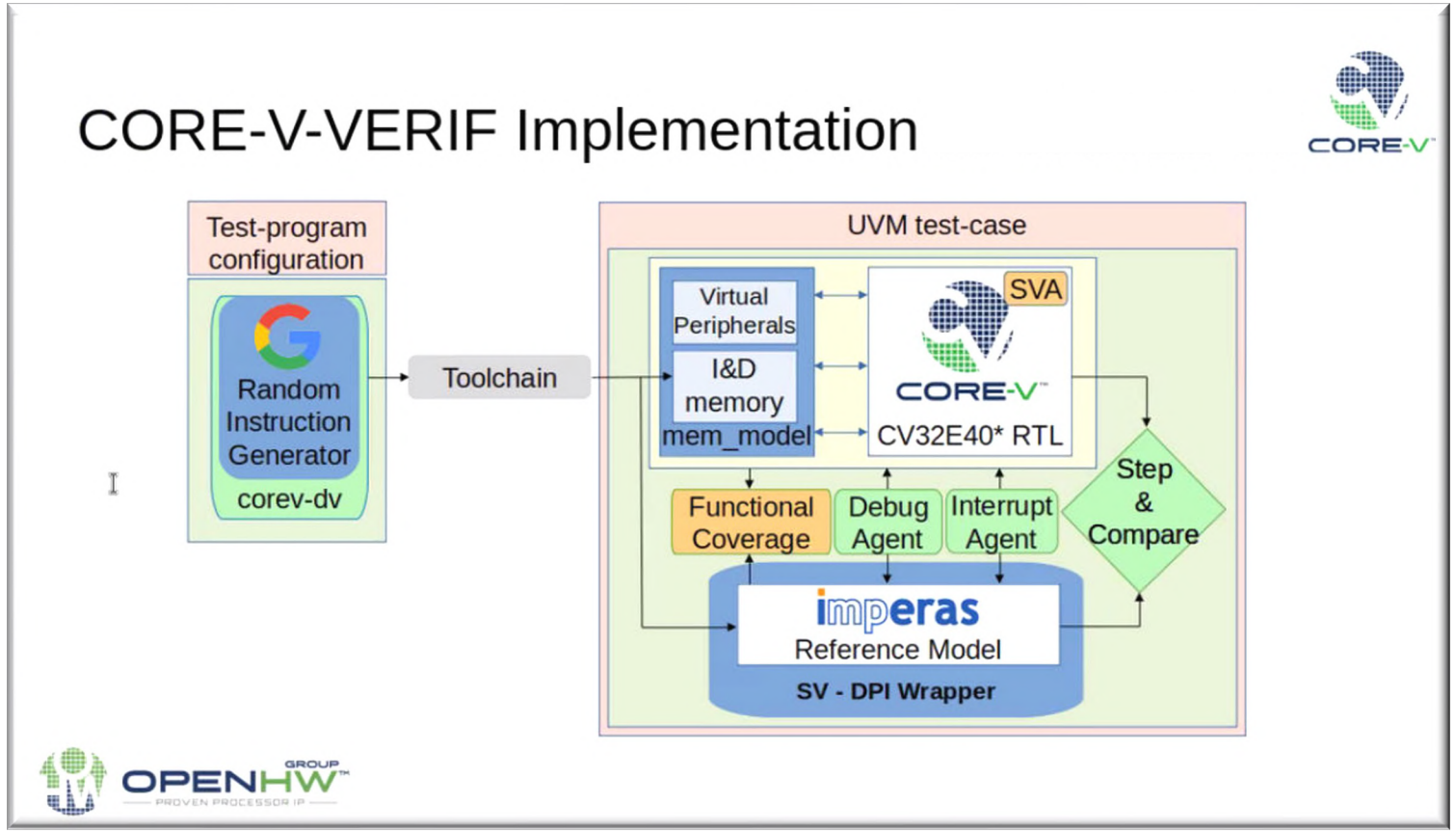
- These need creating and adapting and focusing on the core being verified
- AND – we don't want to have to re-create them for each core… by each team, by each company

- We need to develop these in such a way that they are
  - Configurable without (too many) source edits
  - Extendable to exploit freedoms of RISC-V
  - Develop them to be re-used

- And that is where standards come in…

Example of (ad hoc) DV environment

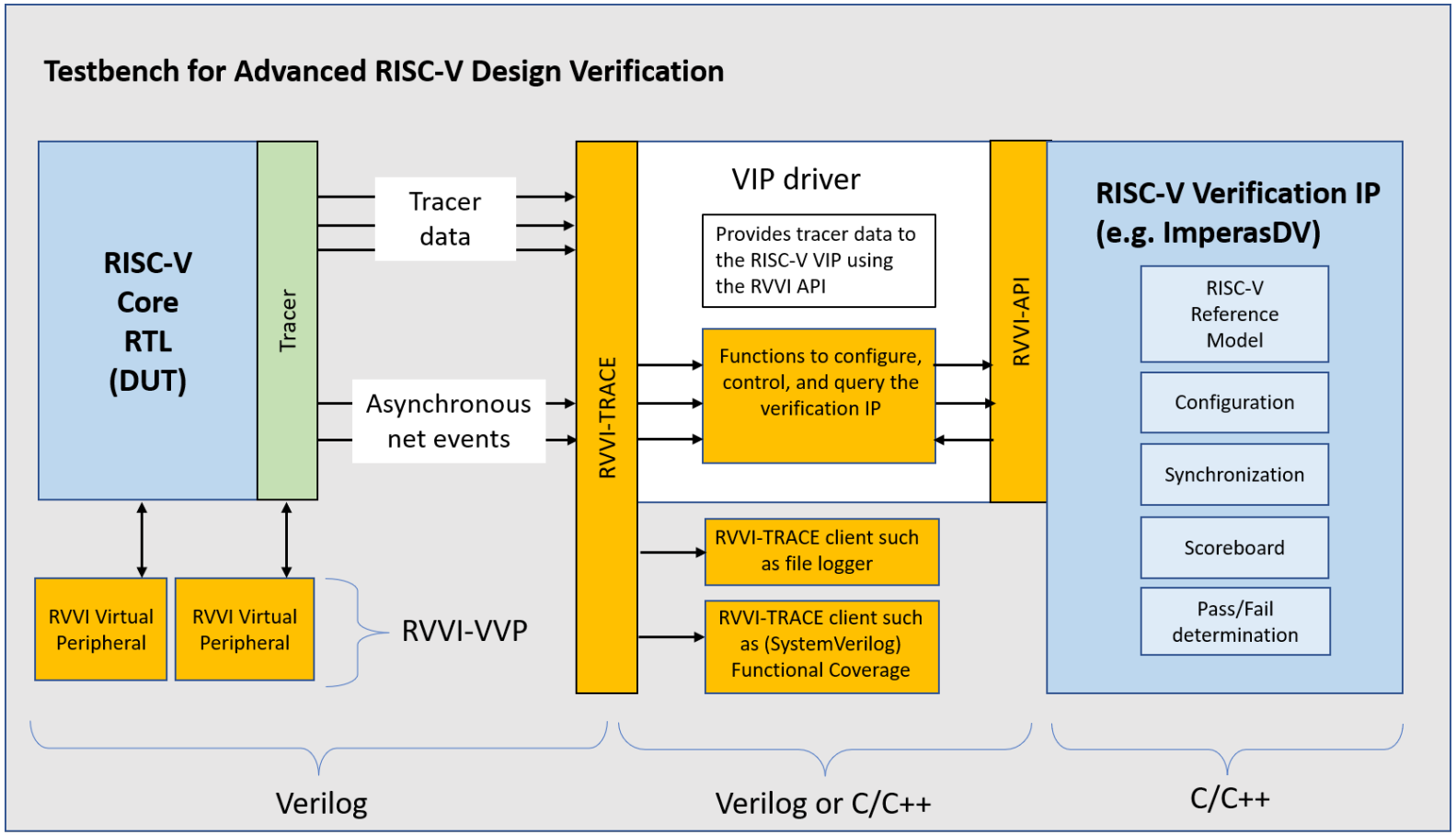
- With focus on:
  - Testbench
  - Reference model
  - Functional coverage
- Challenge for OpenHW is now 8+ cores in roadmap…
- All needing efficient verification



Developed and in use c.2020

Adopting RVVI allows much re-use

RISC-V Verification Interface (https://github.com/riscv-verification/RVVI)

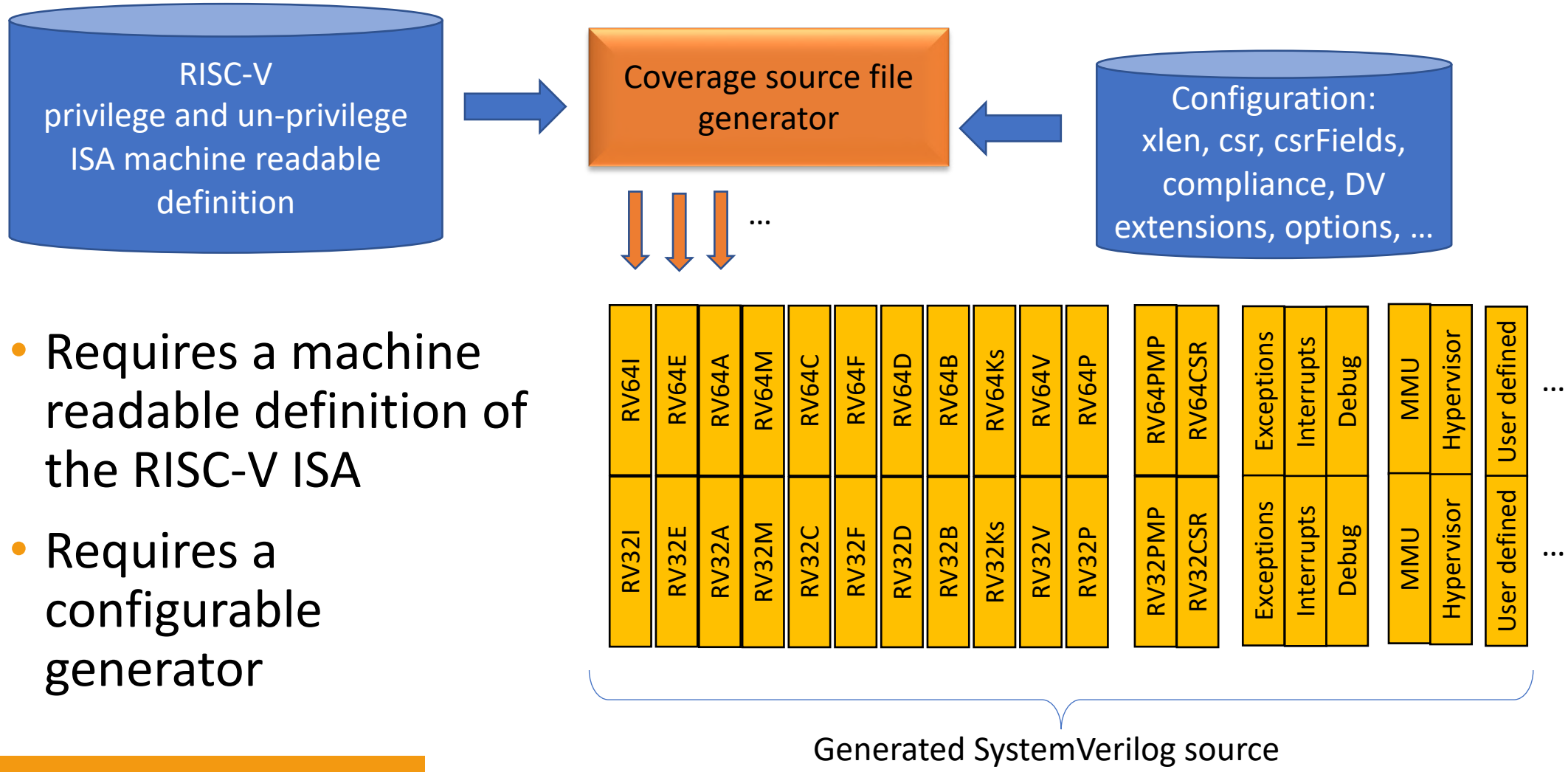


- Uses open standard RVVI
- Allows re-use of test bench VIP components
- Allows re-use of configured reference models
- Allows use of external components such as Functional Coverage

=> Just requires a core specific 'tracer' to interface DUT into RVVI

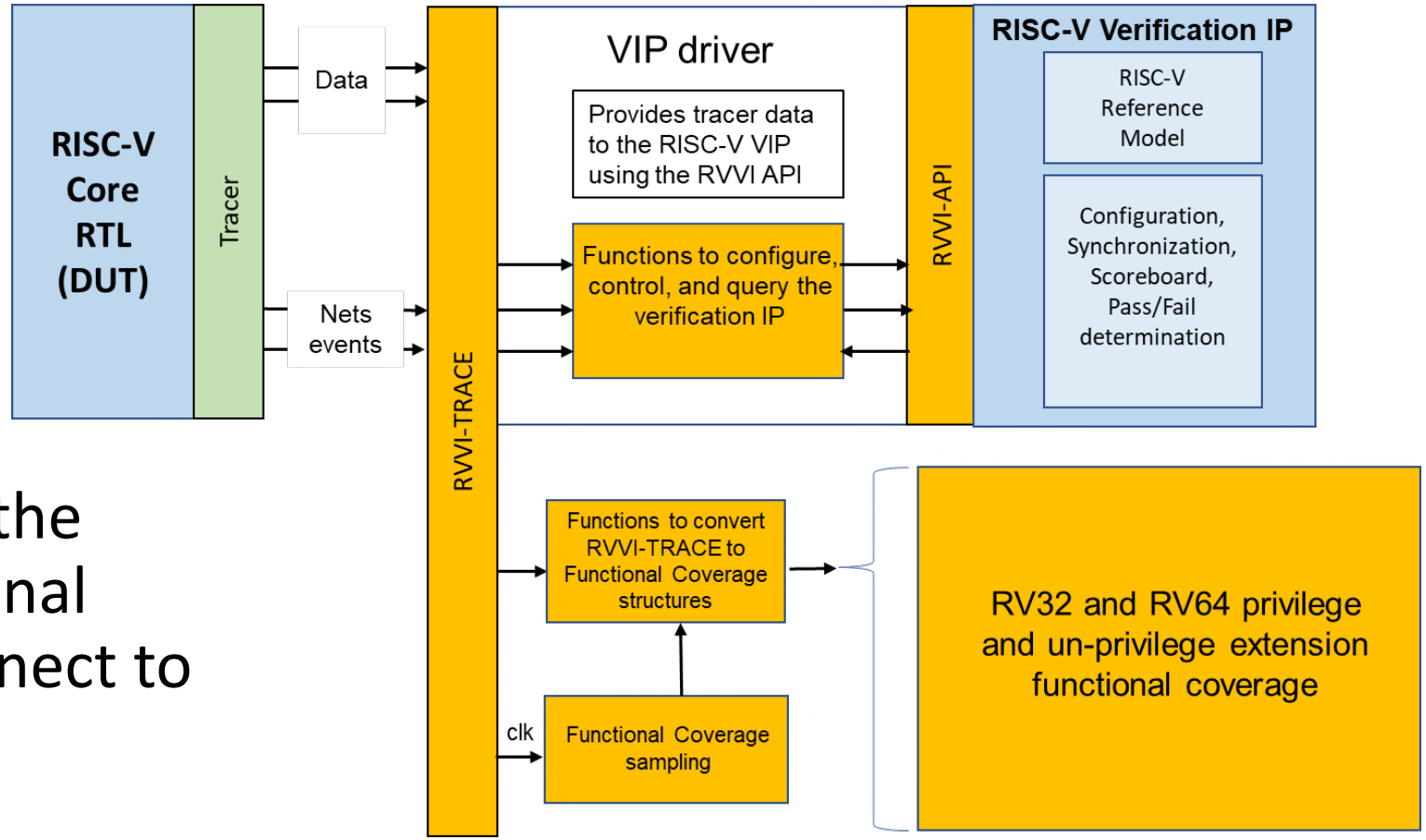
OpenHW Functional Coverage requirements project

riscvISACOV (https://github.com/riscv-verification/riscvISACOV)



- Requires a machine readable definition of the RISC-V ISA
- Requires a configurable generator

OpenHW Functional Coverage - using RVVI standard & generated SystemVerilog source



- Can just 'plug-in' the generated functional coverage and connect to the RVVI

Summary

- RISC-V processors need to be high quality
- High quality means a lot of verification resources
- High quality needs advanced verification methodologies
- Efficiency needs standards and re-use

- RISC-V industry needs to invest in verification ecosystem to complement ISA and SW ecosystems

Conclusions…

- If we fail in this verification challenge, RISC-V will fail to achieve its potential
- Collaboration works well in RISC-V
  - In ISA
  - In SW
- Verification for RISC-V processors needs to become an industry focus
  - Verification collaboration is starting
  - e.g. new projects in OpenHW Group Verification Task Group starting

- For more information on the new verification standards and methodologies come and talk
- There are presentations on the OpenHW stand on RVVI
- And for re-usable Verification IP come and listen to Imperas presentations
- And the verification tutorial on Thursday

Thank you

For more information please visit www.imperas.com/riscv