TOGETHER FOR RISC-V TECHNOLOGY
& APPLICATIONS

RISC-V Summit Europe 2024

TRISTAN: Free Access Training on EDA tooling for RISC-V

09:00-11:30, Friday, 28.06.2024, Munich

TRISTAN has received funding from Chips Joint Undertaking (CHIPS-JU) under grant agreement nr. 101095947.

CHIPS JU receives support from the European Union's Horizon Europe's research and innovation programme and Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey

Experience on IP Design and Verification with open-source and commercial tools

Sara Bocchio - STMicroelectronics

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

RISC-V RTL cores

- We have seen many RTL code of RISC-V cores in this summit
- OpenHW Group · GitHub has an impressive lists for every embedded application!

- What about RISC-V Ips vendors? What about industry grade cores?

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Agenda

- Introduction to industry grade RISC-V cores
- Open source verification
- The layered approach
- Conclusion

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

What is verification?

- Verification is “the act of checking and proving that something is correct or true”

You just need to prove that your core works correctly!

- Practically it’s a burdensome bug hunting into your core
  - BUG = every defect, fault, flaw, or imperfection inside your code

What does it mean a verified core?

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Verification of cores

- Verification is one of the key challenges of modern processor development.
  - Complexity of the designs (thousands of flip flop, pipeline with loopbacks, FSM that are not FSMs…)
  - Uncertainty of the “functionality”
    - It’s more than a complex function (FFT, control logic, OS)
    - Complexity of the specification
      - ISA (Instructions Set Architecture) specification
      - Privilege Architecture specification (status: user, special event handling, memory privileges)
      - Microarchitectures specifications (Core interfaces, Cache support, Branch predictions, pipeline loops)
      - Plus some specific features (security extensions, safety…)

- On top of that, what does it means to enter in the open source “arena” of RISC-V?

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Verification of cores @ life.augmented

- When you stop the verification?

- We used the following metrics to validate the verification process
  - Code coverage (BET and FSMs) -> 100%
  - Functional coverage -> 100%
  - Qualification by faults injection -> 100%

- Qualification by fault injection
  - Insert faults in your DUT
  - Verification regression should able to detect it

  (= failure)

Fault detail

File name : RTL file path

Fault ID	Fault Type	Fault In Report	Status	Detected By Test
691	OutputPortStuckAt0	✓	Detected	Testname
692	OutputPortStuckAt1	✓	Detected	Testname
→ 693	OutputPortNegated	✓	Detected	Testname

With the fault 693 of type 'OutputPortNegated', the code:

```
16    output reg [12:0] o_ch_err_x,
```

Is changed into:

```
16    output reg [12:0] /* port value negated */,
```

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Vs Open Source Verification

- RISC-V has a rich ecosystems in terms of SW Tools (Compilers, ISS), and verification testbenches
- What about verification or better verified IPs?
  - Almost all open source IPs do not provide coverage numbers
  - Only one of them
  - https://ibex-core.readthedocs.io/en/latest/03_reference/verification.html

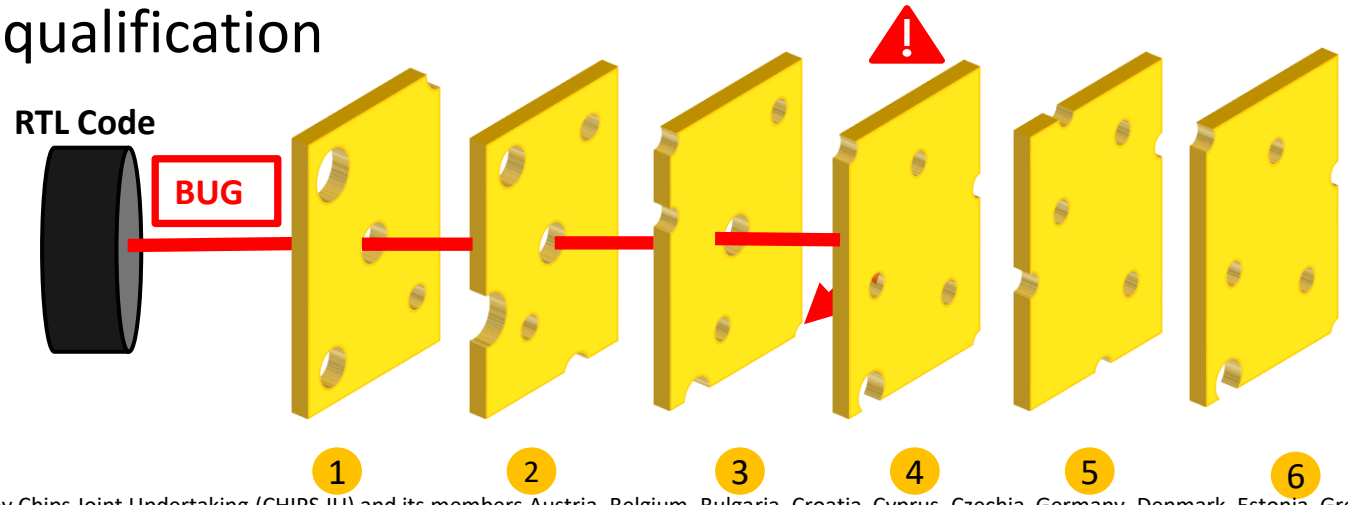
Nightly regression results, including a coverage summary and details of test failures, for the `opentitan` Ibex configuration are published at https://ibex.reports.lowrisc.org/opentitan/latest/report.html. Below is a summary of these results:

Total Tests	1415	Tests Passing	93.5%	Functional Coverage	93.7%	Code Coverage	94.6%

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

The approach in

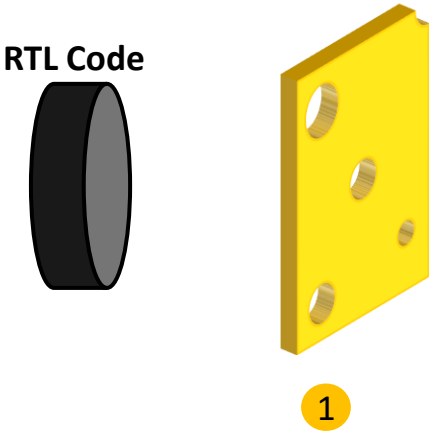
- Layered verification strategy
  1. Code Review (SUPERLINT, XChecks)
  2. Direct Tests
  3. Random Tests
  4. Formal verification
  5. SW benchmarks testing (Coremark, Dhrystone, OS) in FPGA
  6. Coverage and testbench qualification



The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Layer 1: Code review

- Layered verification strategy
  1. **Code Review (SUPERLINT, XChecks)**
  2. Direct tests
  3. Random Tests
  4. Formal verification
  5. SW benchmarks testing (Coremark, Dhrystone, OS) in FPGA
  6. Coverage and testbench qualification



The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Layer 1: Code Review

- Perform basic verification as soon as enough RTL is available
  - Structural lint checks, e.g., naming, coding style, simulation-synthesis mismatch, synthesis
  - DFT checks (both shift and capture mode) to ensure design is testable
  - auto-formal checks, e.g., dead-code, FSM reachability/deadlock/livelock, bus contention, case, arithmetic overflow, and out-of-bounds index

- Open source https://chipsalliance.github.io/verible
  - Only structural lint checks, e.g., naming, coding style, simulation-synthesis mismatch, synthesis

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

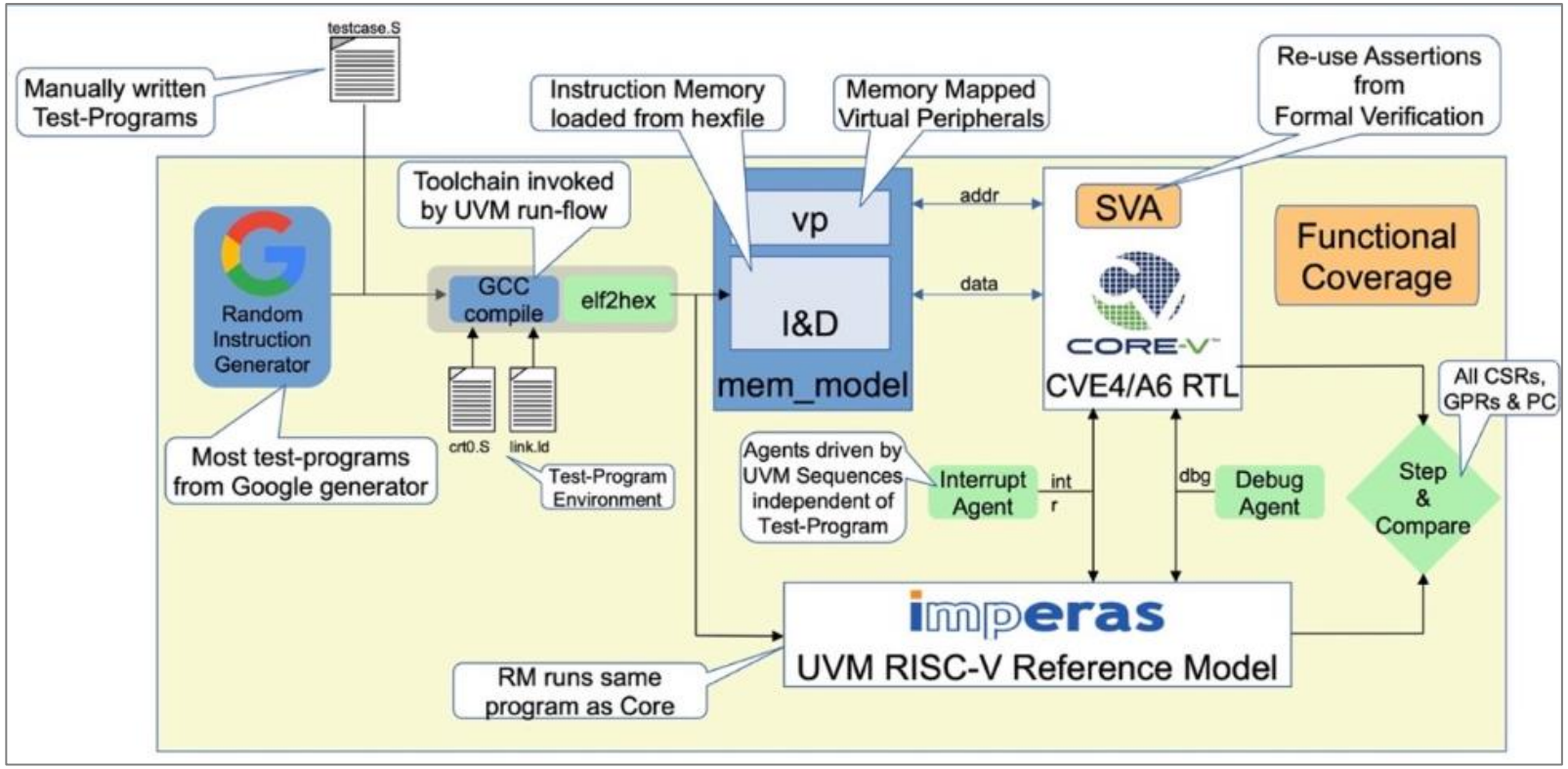
Layer 2 : direct tests

- Standard dynamic verification: DUT is simulated in a testbench that provides stimuli to the DUT and check the expected behavior
- Two type of stimuli
  - Code running on the core
    - Single instruction: basic functionalities + corner cases (e.g arithmetic overflow)
    - Priviledge functionalities: memory protections, interrupts, debug features…
    - Microarchitectures: some specific unit (intensive loads/store), cache particular states….
  - External asynchronous event (interrupt, debug requests) from testbench
    - Core answers to these stimuli by executing code placed in specific location (interrupt handler, debug handler…)
    - Codependency among the two!

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Layer 2: open source scenarios

https://github.com/openhwgroup/core-v-verif



The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Layer 2 open source scenarios

- When we verifying RISC-V we're able to reuse
  - The UVM testbenches
  - riscv-tests https://github.com/riscv-non-isa/riscv-arch-test
    - A simple test framework focused on sanity testing the basic functionality of each RISC-V instruction. It's a very good starting point to find basic implementation issues.
    - No privileged functionalities testing are available in the opensource community (Memory protection)
    - Priviledge are based on
    - No microarchitecture tests
  - The reference model (ISS) spike https://github.com/riscv-software-src/riscv-isa-sim
    - Not so golden (some bugs have been found)
    - Specialization (e.g. adding CSR for cache) not so simple
    - Change some implementation dependency features is even more complicated!

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

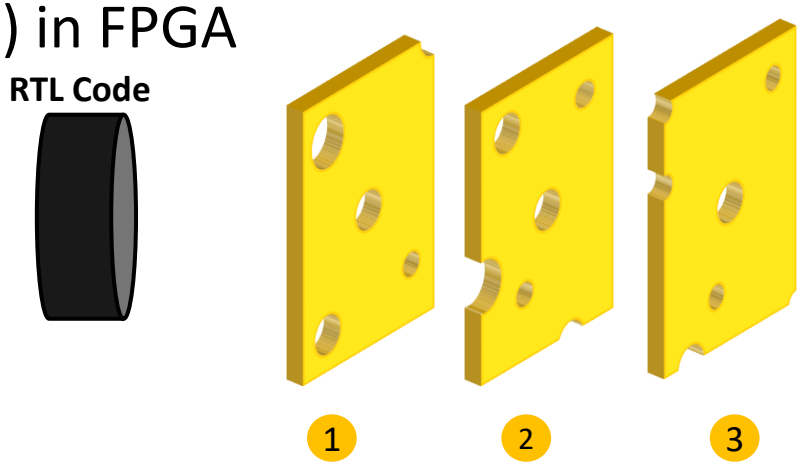
Layer 2: open source scenarios

- Verilator limits:
  - encrypted IP (from third party)
  - # statements (pll and multiclock dmain)
  - mixed Verilog and VHDL designs/ mixed signal
- From verification point of view
  - UVM full support only added recently (2023 – AntMICRO)
  - limited supported of SVA (property, functional coverage)
    - Even open source riscv Ips has to include proprietary compilation/simulation flow for testing and debugging!

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Layer 3: Random tests

- Layered verification strategy
  1. Code Review (SUPERLINT, XChecks)
  2. Direct Tests
  3. **Random Tests**
  4. Formal verification
  5. SW benchmarks testing (Coremark, Dhrystone, OS) in FPGA
  6. Coverage and testbench qualification



The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Random Test generator

- https://github.com/chipsalliance/riscv-dv/
  - Google RISC-V random tests generator

A good random test generator

01 Randomness

Randomize everything: instruction, ordering, program structure, privileged mode setting, exceptions..

02 Architecture-aware

The generated program should be able to hit the corner cases of the processor architectural features.

03 Extendability

Easy to add new instruction sequences, custom instruction extension, custom CSR etc.

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Architecture Aware

- Execution unit could bypass the results to its own inputs

RAW

R3 <- R4 + R5
R1 <- **R3** + R8

WAR

R2 <- **R4** + R5
R4 <- R3 - R8

WAW

R2 <- R4 + R5
R2 <- R3 + R8

- Aim is to generate valid instruction to include sequence WAR like

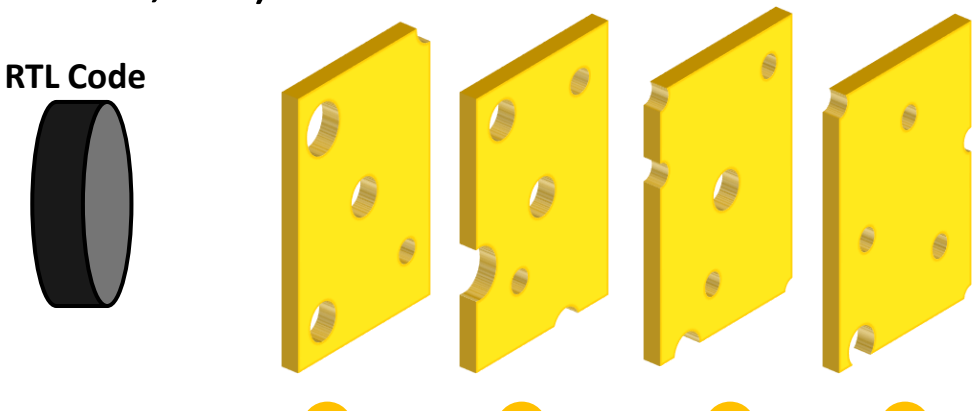
```
lw x1, (x12)
sb x2, (x1)
```

- But the google-dv is not able to generate similar sequence ….

```
lui x10, 0x8000
li x10, 0x30
sw  x2, 28(x10)
```

Layer 4: Formal Verification

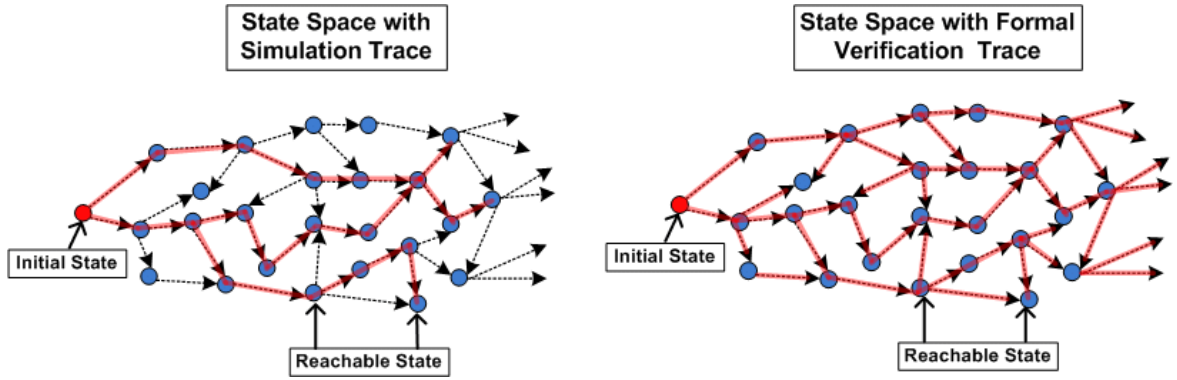
- Layered verification strategy
  1. Code Review (SUPERLINT, XChecks)
  2. Direct Tests
  3. Random Tests
  4. **Formal verification**
  5. SW benchmarks testing (Coremark, Dhrystone, OS) in FPGA
  6. Coverage and testbench qualification



The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Formal Verification

- Formal Verification is the use of tools that mathematically analyze the space of possible behaviors of a design, rather than computing results for particular values.



- During the formal analysis, the tool drives the free signals and explores all the reachable states of the design, trying to prove that the assertions are wrong
- Possible outcomes:

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

What we use

- Assertions in sub-blocks from the designers
- Instruction semantics by model checking
  - An add is an add …
  - Open source is available!
- ABVIP (AMBA VIP) to formally prove the bus protocol.
- Assertions at higher level of abstraction

  PROPERTY: the core must execute instructions i.e.

```
PC_donotstuck: {s_eventually pc_valid}
```

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Our experience

- Most of the work was related to the constraint (assumptions) that define the working conditions
  - Eg example of constraints for for pc do not stuck

```
write_tohost:
                    j write_tohost
```

Formal help us to overcome the limit of dynamic verification that is based on constrained randomization

- Ref: S. Bocchio, T. Majo “How Formal helps RISC-V Processor verification for an effective implementation on silicon.“, CadenceLIVE Europe 2023

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

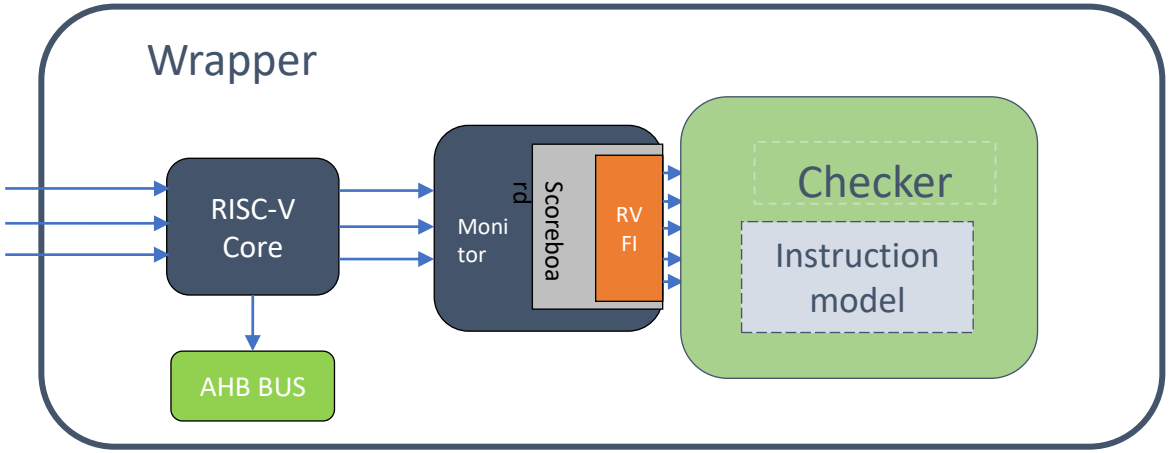
Open sources RISCV-Formal

- **riscv-formal** is an open-source framework for formal verification of RISC-V processors. It consists of the following components: (link to github https://github.com/YosysHQ/riscv-formal)
  - A processor-independent description of the RISC-V ISA
  - A set of checkers, containing the assertions that must be proved during formal analysis
  - The specification for the RISC-V Formal Interface **(RVFI)** that must be implemented by a processor core to interface with riscv-formal.

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Open sources RISCV-Formal

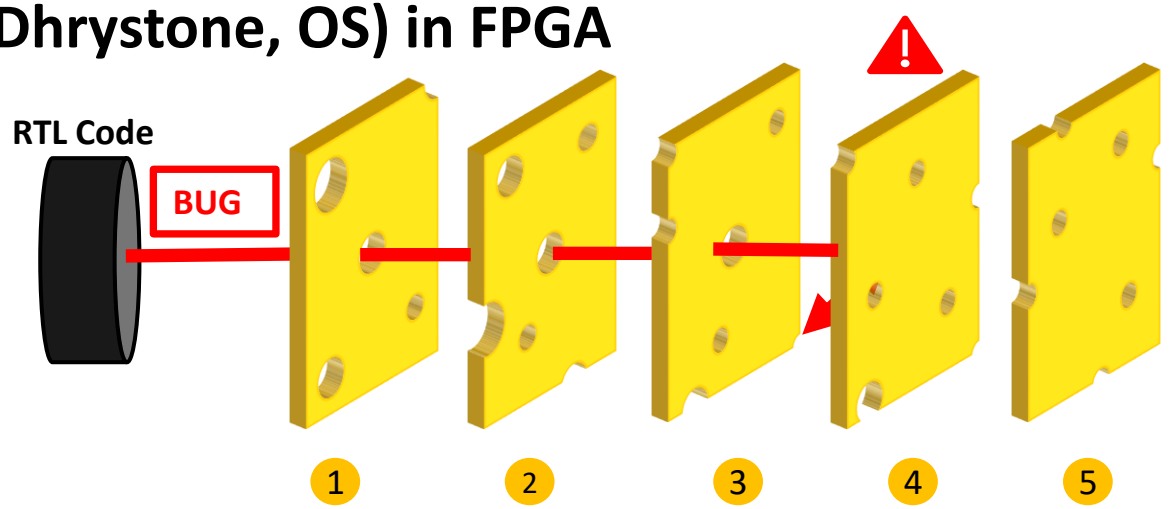
- Ref: “Accelerating Debug and Signoff for RISC-V Processors Using Formal Verification”, Ashish Darbari, Axiomise CDN Europe 2022



The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Layer 5: SW benchmarks

- Layered verification strategy
  1. Code Review (SUPERLINT, XChecks)
  2. Direct Tests
  3. Random Tests
  4. Formal verification
  5. **SW benchmarks testing (Coremark, Dhrystone, OS) in FPGA**
  6. Coverage and testbench qualification



The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

26

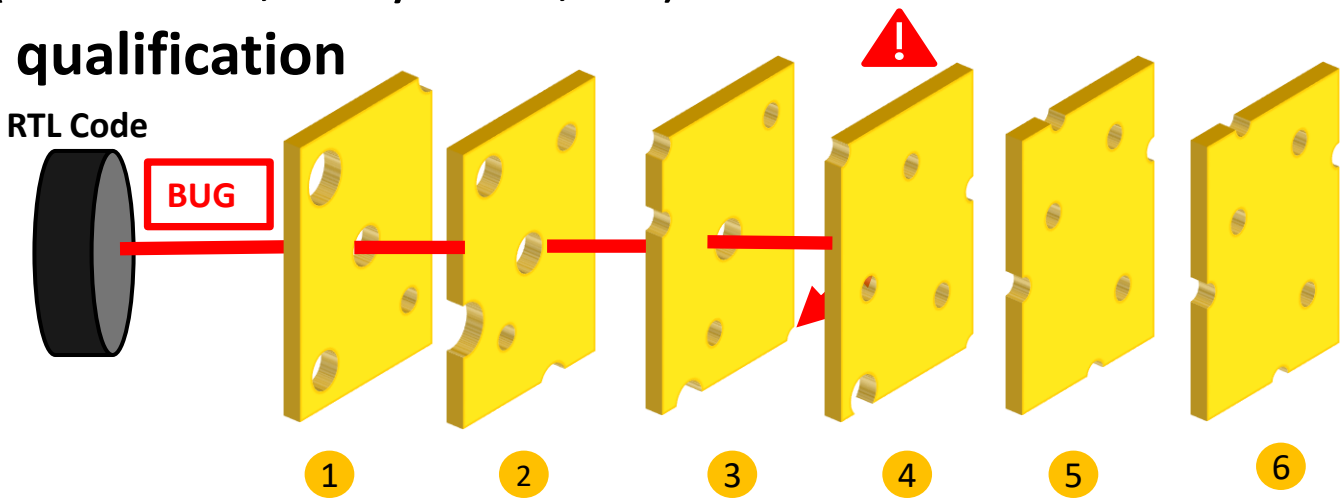
Our experience

- A set of benchmarking applied to our cores in simulation can be found in https://github.com/riscv-software-src/riscv-tests/blob/master/benchmarks/
  - Dhrystones
  - Coremark
  - Embassy …
- FPGA prototyping on RTOS, final application code
- Preliminary results on validations, test on “real” code
  - Performance Validation = verification of “how good” your processor is

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Layer 6: Coverage and testbench qualification

- Layered verification strategy
  1. Code Review (SUPERLINT, XChecks)
  2. Direct Tests
  3. Random Tests
  4. Formal verification
  5. SW benchmarks testing (Coremark, Dhrystone, OS) in FPGA
  6. **Coverage and testbench qualification**



The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Coverage closure

- Before this steps, code coverage reach from 93% to 97%

Nightly regression results, including a coverage summary and details of test failures, for the `opentitan` Ibex configuration are published at https://ibex.reports.lowrisc.org/opentitan/latest/report.html. Below is a summary of these results:

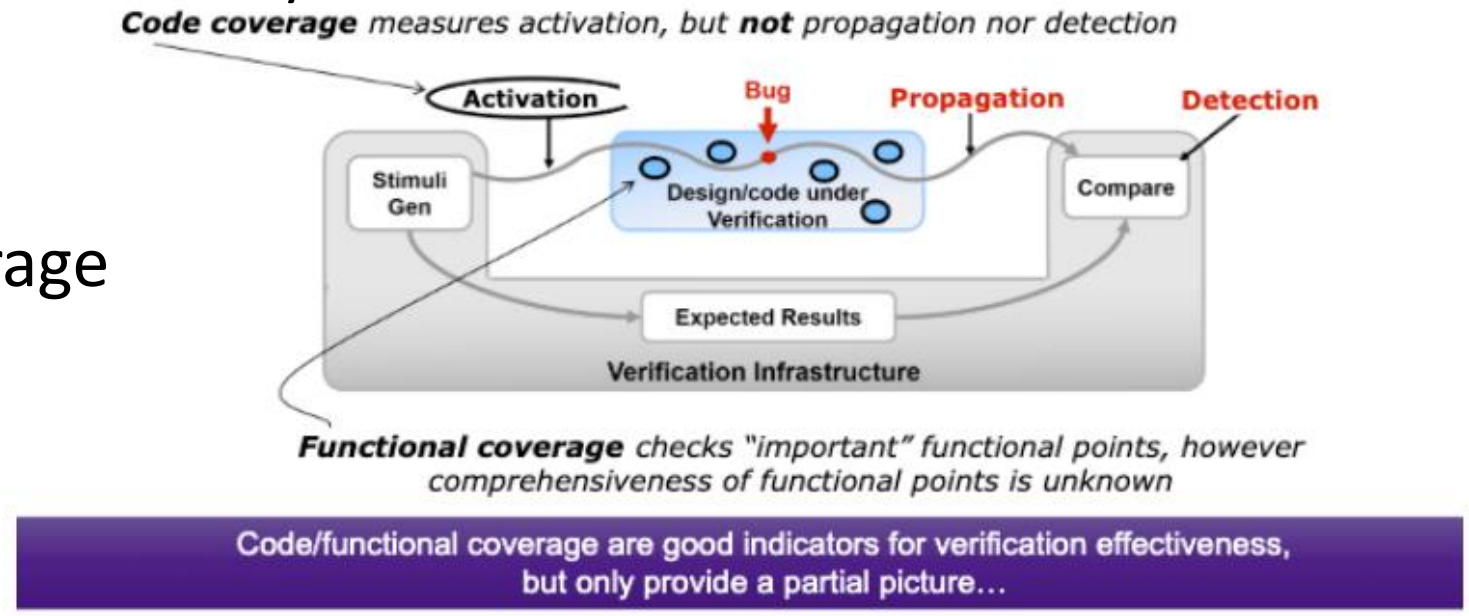
Total Tests	1415	Tests Passing	93.5%	Functional Coverage	93.7%	Code Coverage	94.6%

- Unreachability analysis formally prove to exclude what is not reachable
- Cover the holes!
  - Iterations with designer
  - Formal cover property to extract the scenario
- Manual, tedious and lots of effort
  - Still a few bugs

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Testbench qualification

- Testbench qualification by fault injection
  - Coverage means reachability
  - Fault injections means observability
- Similar process to coverage
  - More tedious ☹
  - Include formal ☺



Source: Synopsys

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

Conclusions

What works for an industry grade verifications is a combination of techniques that evolve as processors become more complex. We develop different approaches as we learn from our experience to refine our verification methodology and offer best-in-class quality IP.

Exposing to open source has helped us

Not a push-the-button verification on the github!	☺ no start from zero (reuse of testbenches, test suites, tools) … but you need to be smart!

The TRISTAN project, nr. 101095947 is supported by Chips Joint Undertaking (CHIPS JU) and its members Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey .

TRISTAN
RISC-V

TOGETHER FOR
RISC-V
TECHNOLOGY
& APPLICATIONS

TRISTAN webpage

TRISTAN LinkedIn

TRISTAN Unified Access Page

TRISTAN has received funding from Chips Joint Undertaking (CHIPS-JU) under grant agreement nr. 101095947.

CHIPS JU receives support from the European Union's Horizon Europe's research and innovation programme and Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia, Turkey