2021 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION CHINA
SHANGHAI | MAY 26, 2021

accellera SYSTEMS INITIATIVE

onespin assuring IC integrity

A Methodology to Verify Functionality, Security, and Trust for RISC-V Cores

TUSINSCHI Nicolae | CHEN Wei Wei | ANDERSON Tom
OneSpin Solutions

The Rise of RISC-V

RISC-V Fundamentals

- Open-source ISA
- Support a wide variety of applications
- Many possible configurations
- Custom extensions – Domain Specific Architectures
- Number of members is increasing continuously
- Ecosystem maturing quickly – toolchain, simulators, verification, …
- Not-for-profit commercial-grade cores – OpenHW Group

Instruction Set Architecture

- “I” base integer instruction set
- “M” extension for integer multiplication/division
- “A” extension for atomic read-modify-write memory accesses
- “F” extension for single-precision (32-bit) floating point
- 32 registers (32-bit, 64-bit, 128-bit)
- 3 privilege levels
- 4096 CSRs
- Interrupts and exceptions

RTL Verification with Formal

RTL Verification Challenges

- Checking compliance with ISA is a significant task …
- … ensuring functional correctness is a very complex task
- Pipelined implementation optimized for power, performance, area
- Many pipeline-based corner cases are impossible to foresee
- Corner-cases related to interrupts, exceptions, privileged modes
- Risk of security vulnerabilities and hardware Trojans

Formal Verification

- It's great …
  - Systematic detection of corner-cases bugs
  - The only technology that can provide exhaustive verification
    - Proof of bug absence
    - Simulation/emulation explore a fraction of the state space
- … but
  - Requires expertise to write good quality assertions
  - Difficult to assess quality of assertions, detect gaps
  - Complexity issues—inconclusive proofs

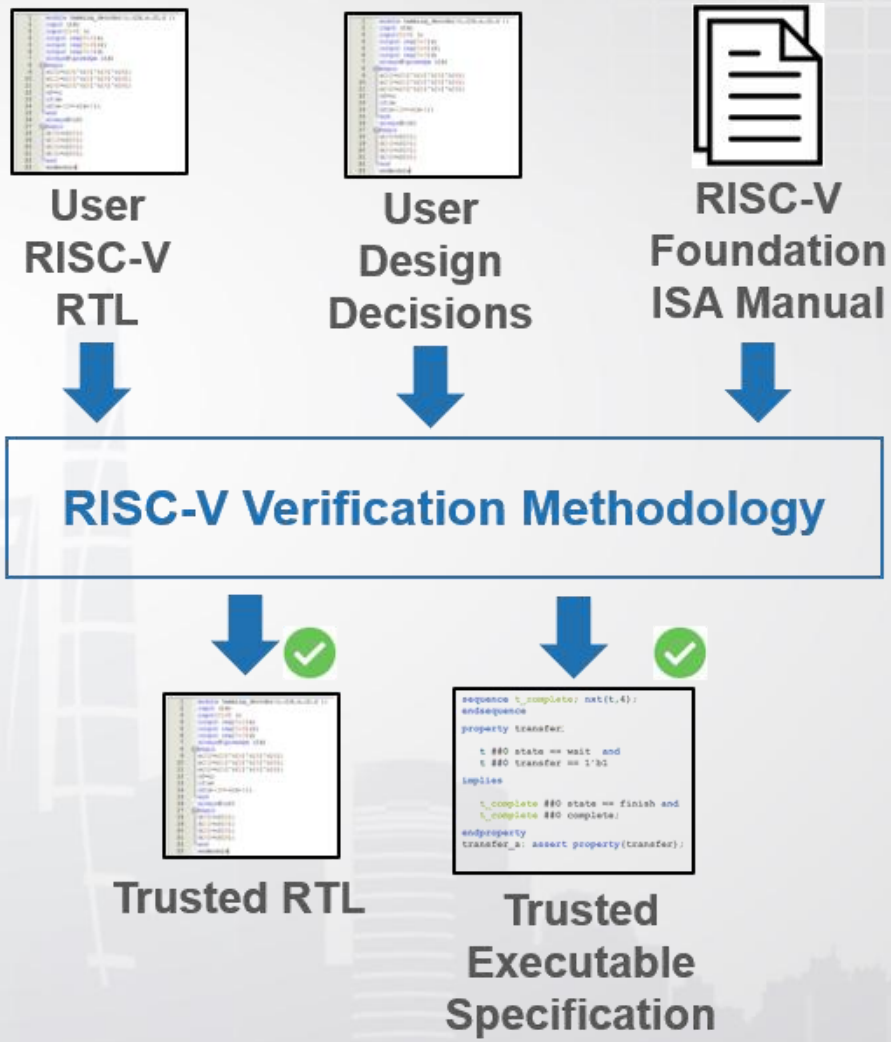
RISC-V Verification Methodology

Inputs

- Core's RTL
- RISC-V ISA (Spec)
- Design implementation decisions (e.g., number of pipeline stages)

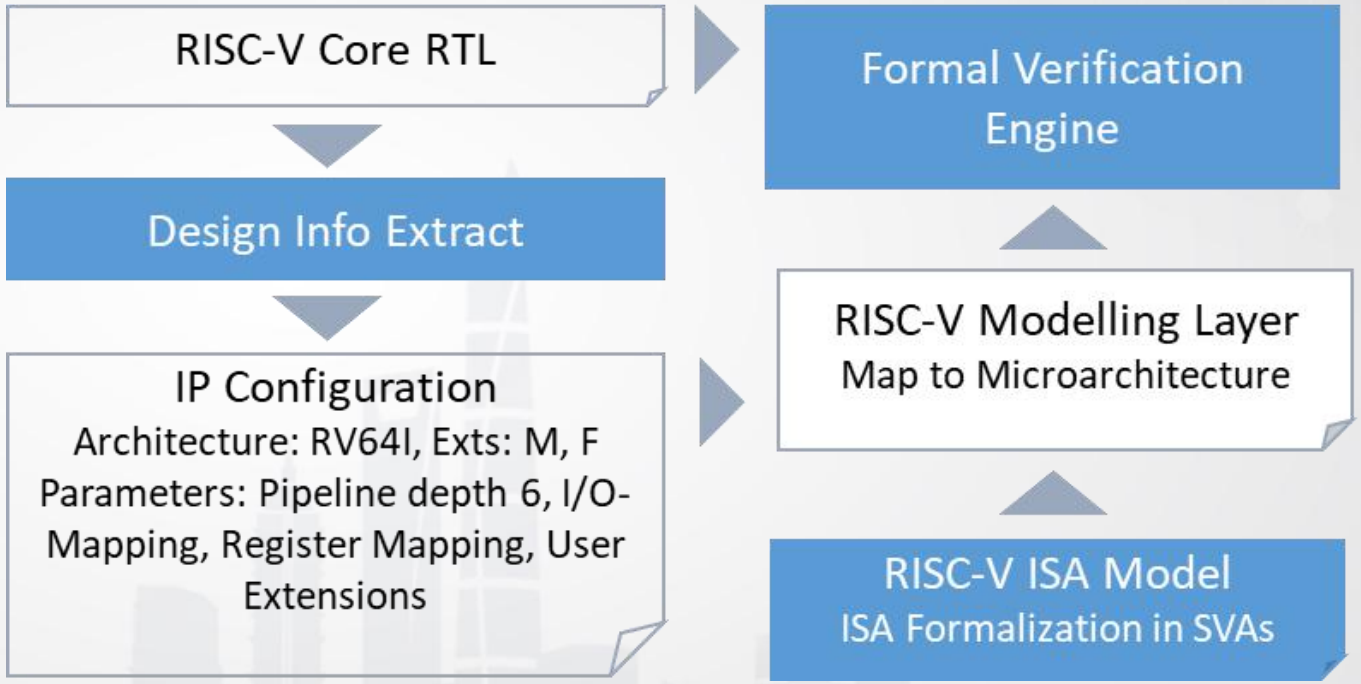
Outputs

- Trusted executable spec
- Proof that RTL is equivalent to executable spec



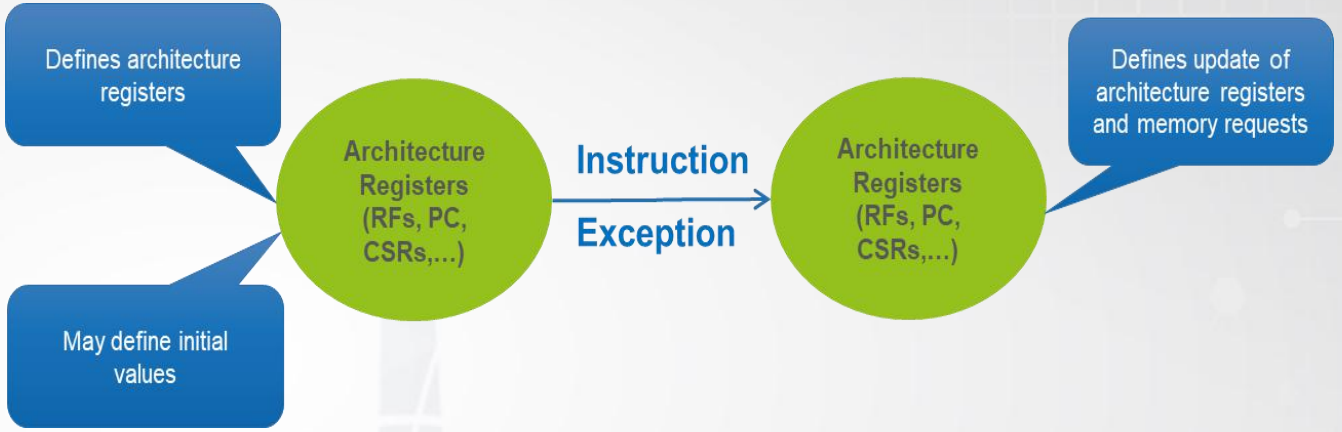
OneSpin's Processor Integrity Solution

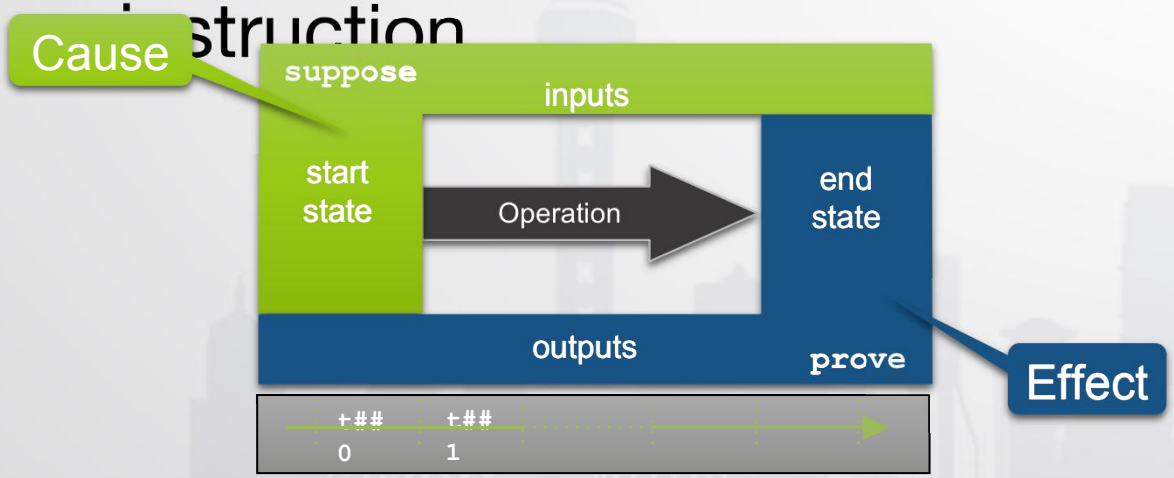
- design info
- Built-in, proven RISC-V ISA formalization in SVAs
- Optimized for exhaustive, unbounded proofs
- Proof that SVAs achieve 100% coverage – no gaps
- Integrated debug features



Operational Assertions and Formalizing ISA

- SVAs use library of Operational Assertions
- Strict coding style to express the expected behaviour of each instruction

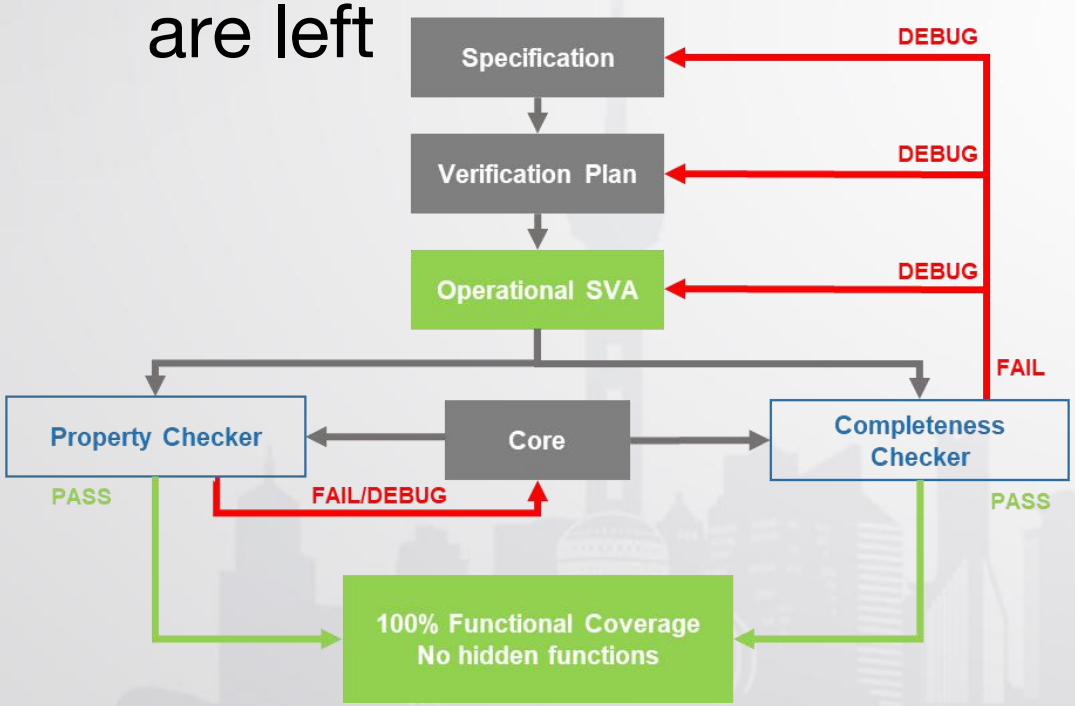




- Capture effects of instruction and exceptions on the architectural state
- Decoupled from micro-architectural details

GapFreeVerification

- Systematic process to cover 100% of functionality
- Formal proof that no gaps are left

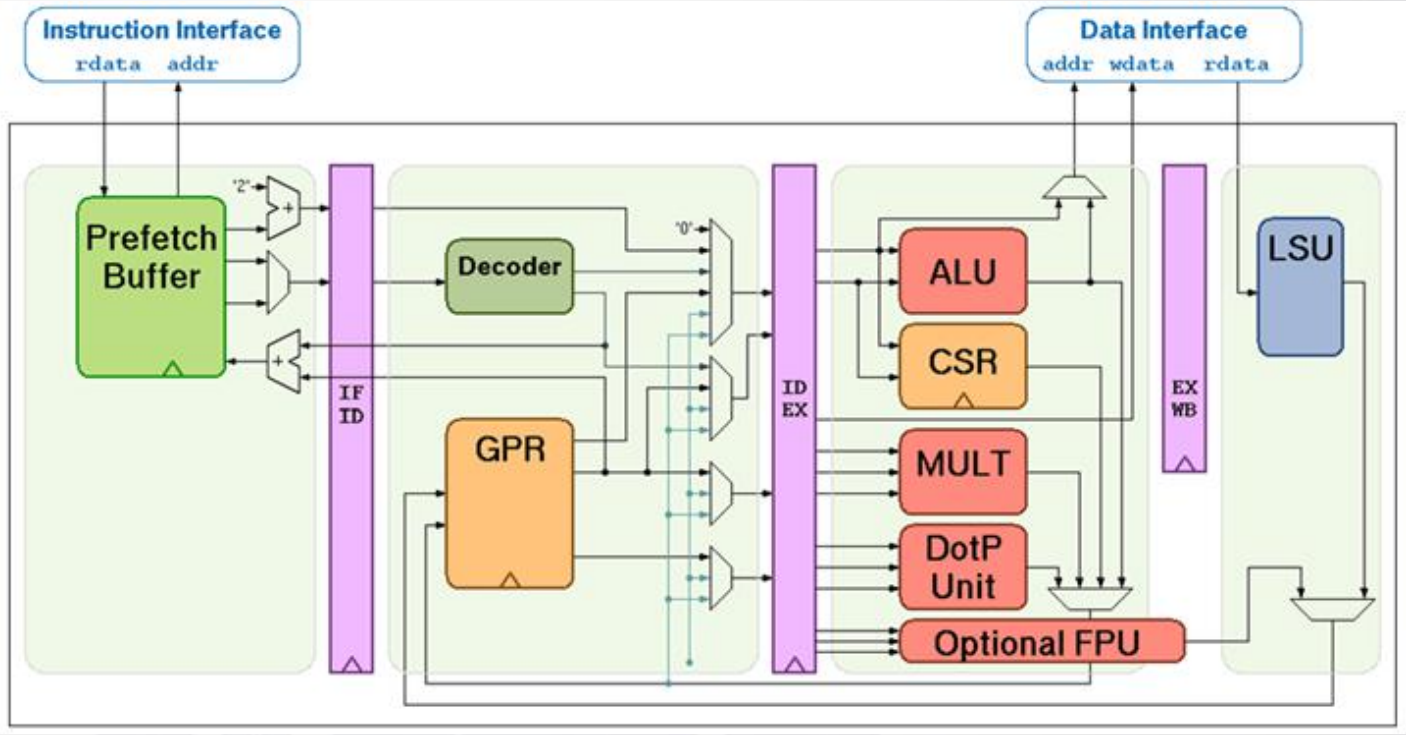


Outcome:

- Proof that ISA's executable model (SVAs) and RTL are equivalent
  - For any input trace the two models produce the same output trace
- Any undocumented or deliberately hidden function is detected

Results – RI5CY (CV32E40P)

- 4 stages, 32-bit
- Core now curated by OpenHW Group
- Target is commercial-grade quality
- Solution applied to bring core's quality to the level of most advanced IP providers

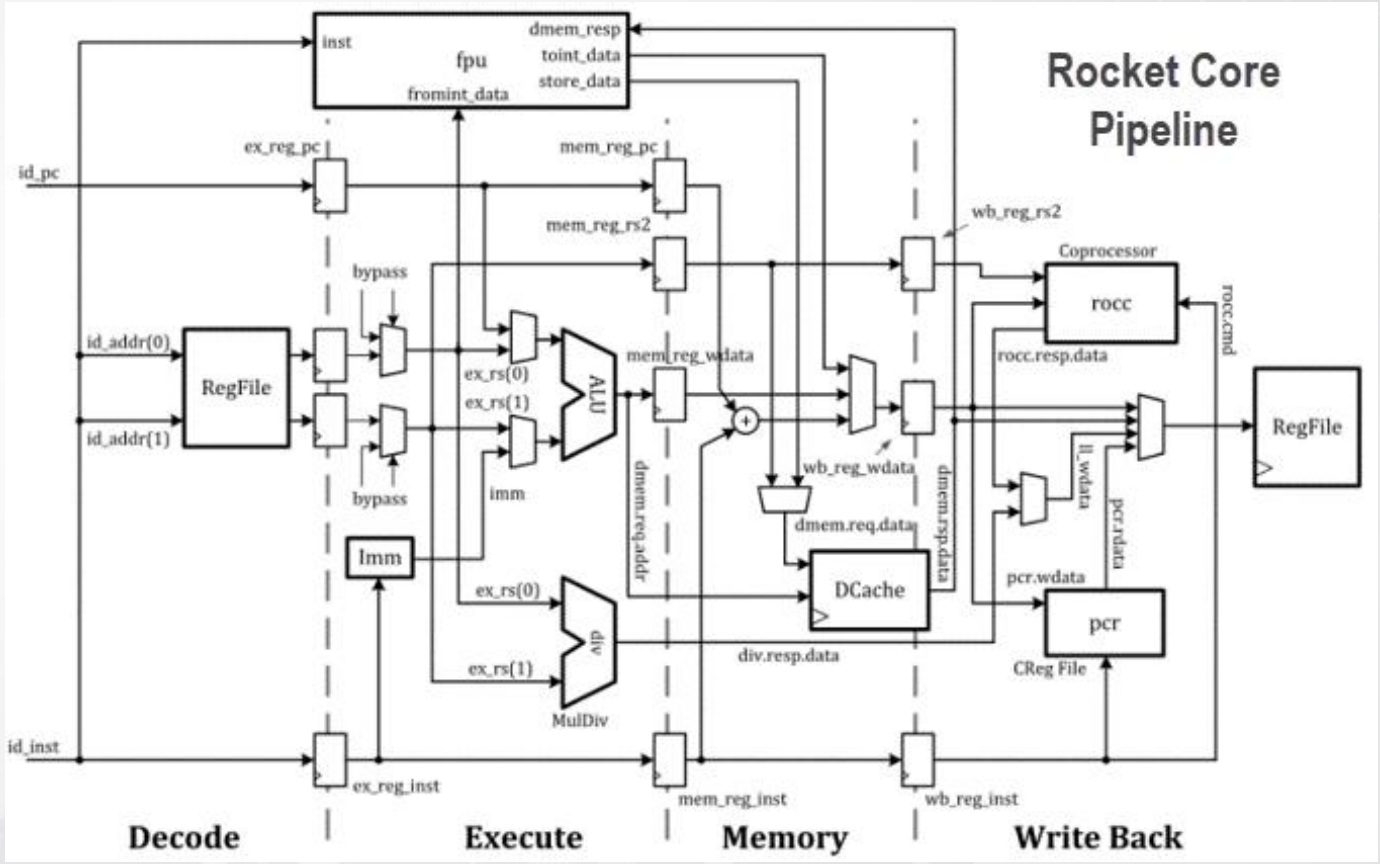


Results – RI5CY (CV32E40P)

- github.com/openhwgroup/cv32e40p/issues
- #157: Exception Handling Violation - dcsr
- #159: Exception Raising Violation - Fetch/Store/Load Access
- #169: Exception Raising Violation - Illegal Instruction - dynamic rounding mode
- #170: Exception Raising Violation - Illegal Instruction - FS field
- #174: F extension - Dynamic Rounding Mode Violation
- #175: F extension - Wrong Result Calculation
- #182: Trap Return Handling Violation - mstatus' MIE
- #185: Debug Mode Violation - Exceptions Update CSRs
- #438: Illegal Instruction Exception not Raised - URET
- #439: Illegal Instruction Exception Raised Incorrectly - C.EBREAK
- #440: Illegal Instruction Exception Raised Incorrectly - CSRs
- #441: Illegal Instruction Exception Raised Incorrectly – MRET
- #442: Illegal Instruction Exception Raised Incorrectly – FENCE
- #443: Incorrect DCSR value read/ written
- #509: Core executes wrong instruction

Results – RocketCore

- 5 stages, 64-bit
- Chisel
- Mostly in-order
- Long latency instruction DIV completes out of order



Results – RocketCore

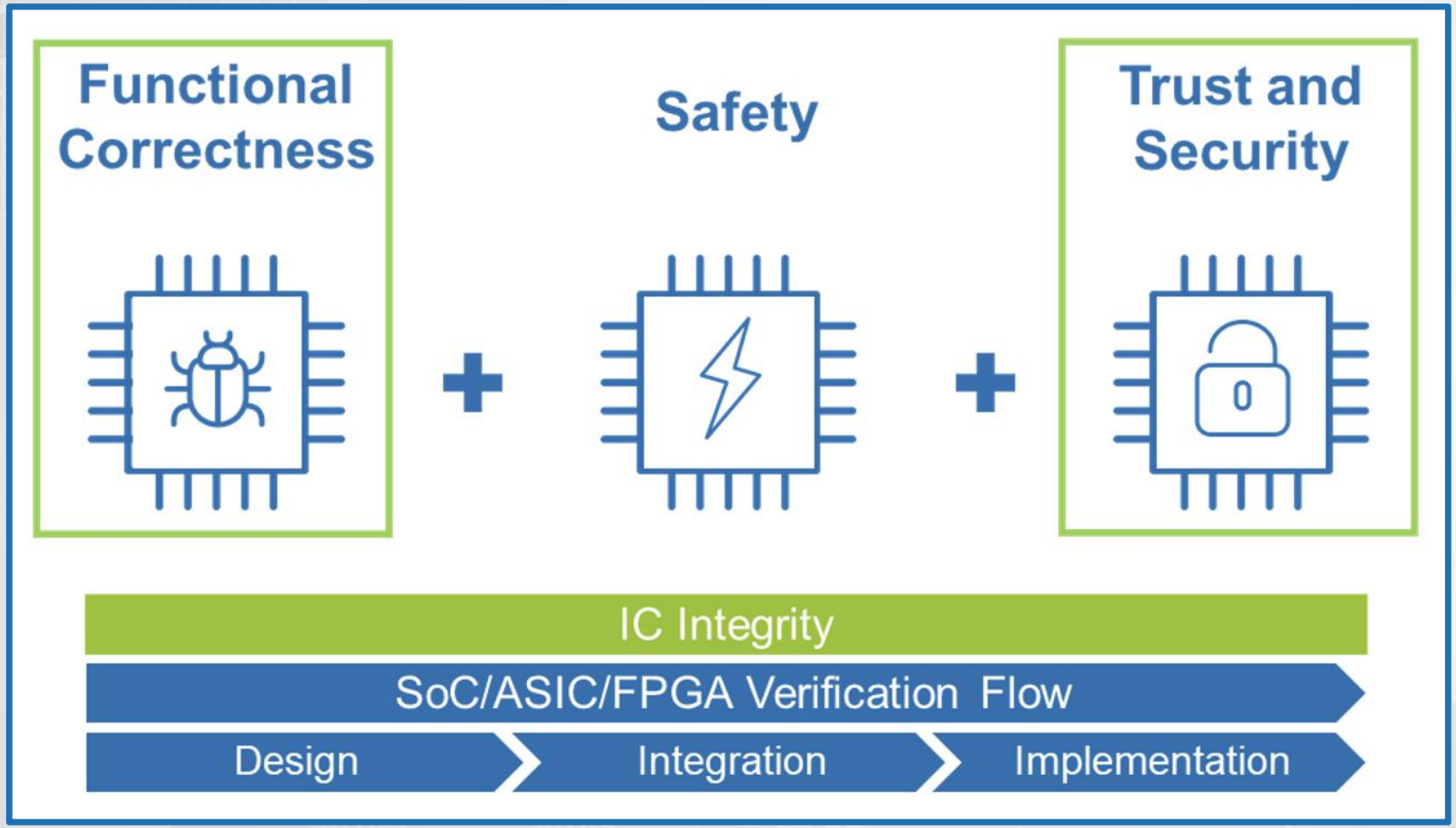
- github.com/chipsalliance/rocket-chip/issues
- #1752: DIV result not written back to register file
- #1757: JAL and JALR jump instructions store different return PC – instruction fetch unit responsible to prevent this issue
- #1861: replay of illegal opcode instruction or instruction with fetch exception
- #1868: undocumented non-standard instruction (opcode 32'h30500073) detected - CEASE
- #1868: presence of non-standard instruction (opcode 32'h30500073) not declared in misa register
- #1949: access to non-existent CSR does not raise illegal instruction exception – open
- #2022: DRET instruction outside of Debug mode does not cause illegal exception
- #2043: DRET instruction illegal exception tied to M mode status

Summary

- RISC-V pre-silicon functional verification is challenging
- Complex implementations – pipeline, performance optimizations
- Many configurations and custom extensions possible
- Many cores – open-source, in-house, third-party

- Formal verification using automated solution allows the user to:
  - Prove that the core complies with RISC-V ISA
  - Detect all corner-case bugs, including in custom extensions
  - Identify security weaknesses, vulnerabilities, and hardware Trojans
  - Applicable during core's RTL development and IP integration into a SoC

OneSpin: Assuring IC Integrity



OneSpin provides certified
IC Integrity
Verification Solutions
to develop
functionally correct,
safe, secure, and trusted
integrated circuits