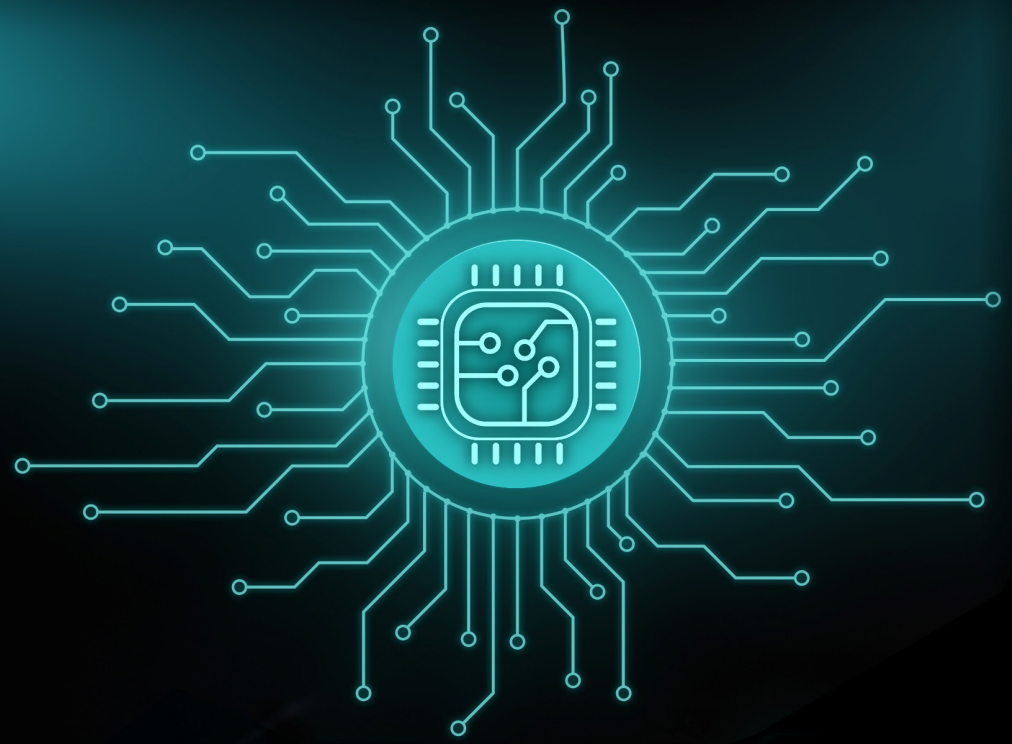

Axiomise Formal Verification Update
CVE4

2 DECEMBER 2020

Agile formal verification for RISC-V

GROUP OPENHW™ PROVEN PROCESSOR IP

axiomise®
predictable formal verification

Summary

End-to-end, vendor-neutral, formal verification for CVE4

About 27,000 formal properties including coverage targets

- ISA (User & Privileged)
- No inconclusive proofs
- Run time of six hours
- End-to-end proofs of compliance
- No cut-pointing, no black-boxing

Weekly regressions were provided: 16 July - 1 Oct

Several issues identified (ISA, RTL, TB, Doc)

Proofs run against multiple commercial formal verification tools

Pass/Fail outcome on all assertions and covers from two different tools must agree

Coverage results reviewed

Key verification activities

Axiomise contributions

ISA checks formally prove that all user-instructions within RV32IC work correctly

Detailed plan for exception handling created and checks were verified exhaustively

Verification plan was provided to designers and Mike Thompson for review

Regular weekly status updates were provided

Verification carried out with both PULP extensions enabled as well as disabled

- We do not check if PULP extension instruction works correctly
- We check that other ISA instructions do not break in presence of PULP extensions

ISIDE and DSIDE formal for protocol compliance with respect to the OBI spec

X-checks establish that X is not visible when appropriate valid conditions are set

Deadlock checks establish that FSMs are never stuck

Methodology overview

Brief description of how our formal verification is set up

All properties are coded in the *formalISA®* app

Each instruction is issued arbitrarily any number of times

Any number of stalls, debug and interrupt events can intervene between the issue and commit

ISA checks establish that upon commit, the instruction execution yields the expected result

Expected result semantics is obtained by reconciling the user ISA and the CVE4 micro-arch spec

All checks are verified with debug and interrupt enabled as well as disabled

We prove that between any two instructions the pipeline can be stalled for arbitrary time

Memory model is randomized to not return the data always within the same bound

- LSU checks are verified with memory randomization

Coverage

Six steps towards sign-off

1. Processor coverage

   Reviewed coverage target list is complete against published ISA

2. Assertion coverage

   Obtained from multiple tools – list of Pass/Fails

3. Checker completeness
   a) All checkers are activated arbitrarily many times, and produce expected results
   b) Bugs introduced in design trigger failures
   c) Bugs introduced in properties trigger failures

4. Over-constraint detection

   Ensured healthy TB. 99.36% of TB was reachable

5. Property-driven design coverage
   a) For every checker, the checker coverage (stimulus & proof core) reported by JasperGold is as expected
   b) Code coverage reported by Cadence JasperGold® and Mentor's QuestaPropcheck® is as expected
   c) Deadcode and unreachable code reviewed
   d) Statement, branch, and expression coverage reviewed

6. Scenario coverage provides evidence of specific scenarios through proofs and waveforms

Contact info@axiomise.com for more information