onespin

Complete Formal Verification of RISC-V Cores for Trojan-Free Trusted ICs

Sergio Marchese
Technical Marketing Manager
OneSpin Solutions
sergio.marchese@onespin.com

October 1 - 2
2nd RISC-V Meeting

cea NANOELEC

assuring IC integrity

Abstract

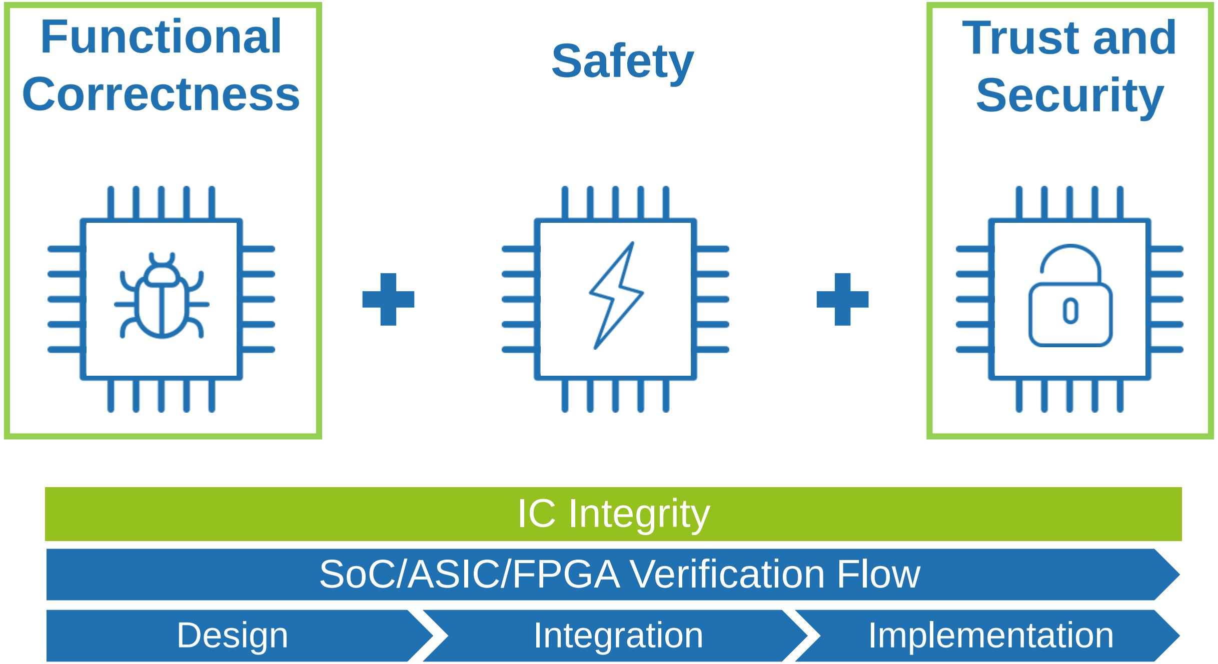
RISC-V processor IPs are increasingly being integrated into system-on-chip designs for a variety of applications. However, there is still a lack of dedicated functional verification solutions supporting high-integrity, trusted integrated circuits.

This presentation examines an efficient, novel, formal-based RISC-V processor verification methodology. The RISC-V ISA is formalized in a set of Operational SystemVerilog assertions. Each assertion is formally verified against the processor's RTL model. Crucially, the set of assertions is mathematically proven to be complete and free from gaps, thus ensuring that all possible RTL behaviors have been examined. This systematic verification process detects both hardware Trojans and genuine functional errors present in the RTL code.

The solution is demonstrated on an open-source RISC-V implementation using a commercially available formal tool, and is arguably a significant improvement to previously published RISC-V ISA verification approaches, advancing hardware assurance and trust of RISC-V designs.

IC Integrity

Functionally correct, safe, secure, and trusted SoCs/ASICs/FPGAs



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

Agenda

RISC-V verification and trust assurance challenges

RISC-V Integrity Verification Solution

Case study: Rocket Core

RISC-V Background

"The Free and Open RISC ISA"

Developed at the University of California, Berkeley
Instruction set architecture (ISA) designed for flexibility
Free, open-source and royalty-free

Supported by the RISC-V Foundation

- More than 200 members, including OneSpin

OpenHW Group

- Not-for-profit organization
- Provides high-quality open-source HW
- CORE-V: family of RISC-V cores
- OneSpin is a sponsor

Functional Verification of RISC-V Cores

Does the RTL correctly implement the RISC-V ISA spec?

Processor cores are hard to verify

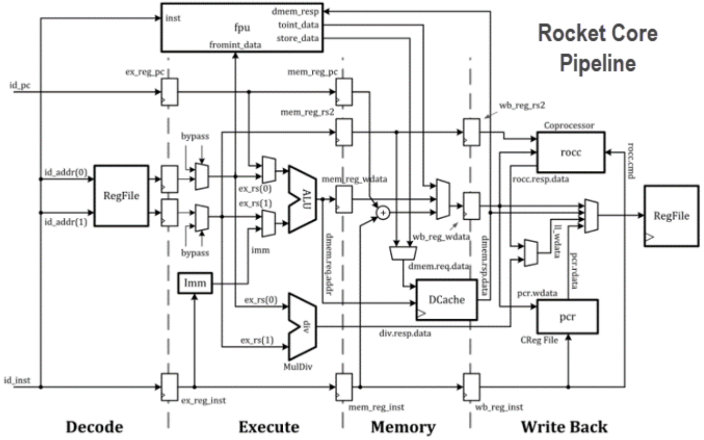
- Complex microarchitecture to achieve PPA targets
- Branch prediction, forwarding, out-of-order execution …

Formal verification

- Exhaustive analysis finds corner-case bugs
- The only technology with potential to prove absence of bugs

Challenges

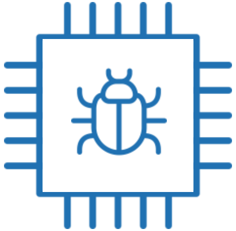
- Complexity issues lead to bounded proofs
- Hard to write good quality, reusable assertions



Trust Assurance and Security Verification

Require new solutions, metrics, processes

Functional Correctness

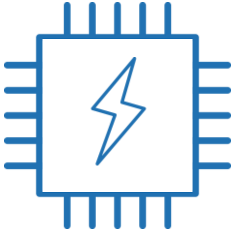


Does the IC do what it is supposed to do?

Use Cases

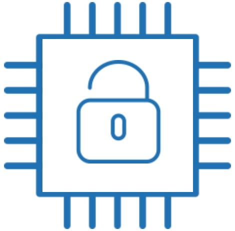
+

Safety



+

Trust and Security

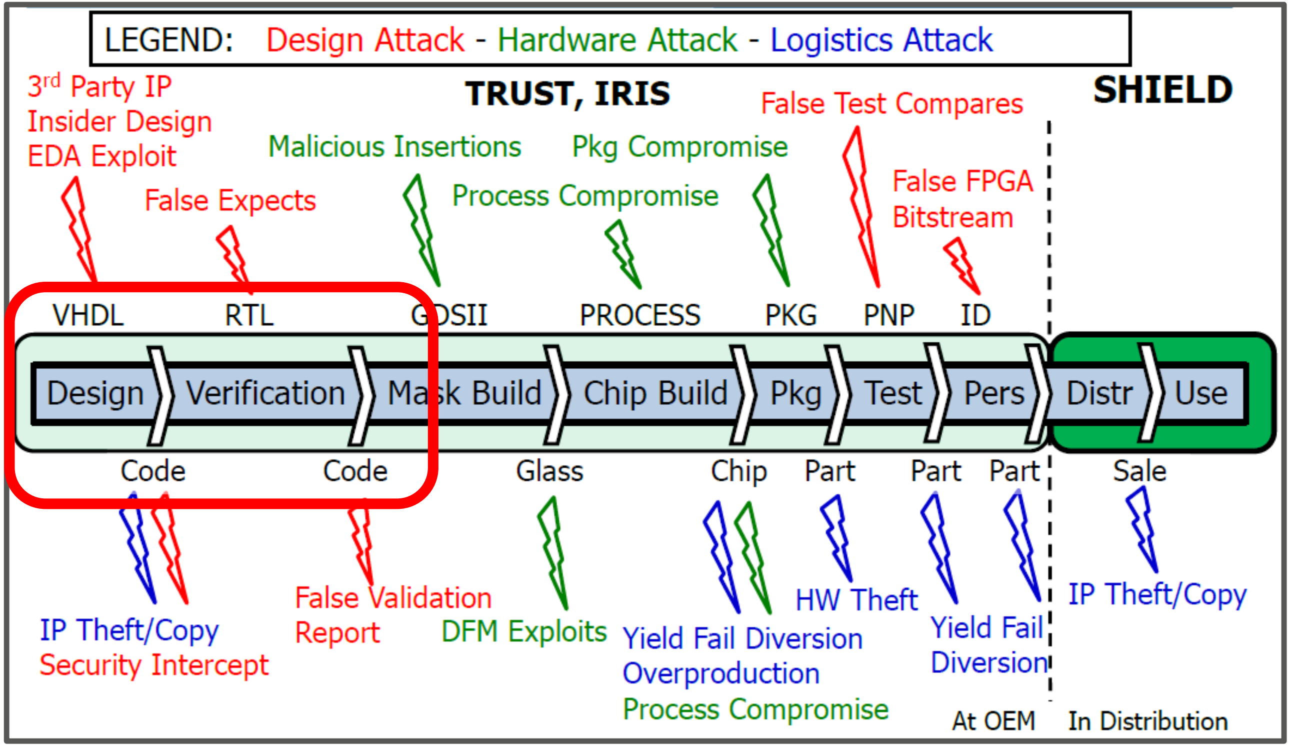


Does the IC do anything that it is not supposed to do?

Trojans, Misuse Cases

Trust Assurance for Integrated Circuit (IC)

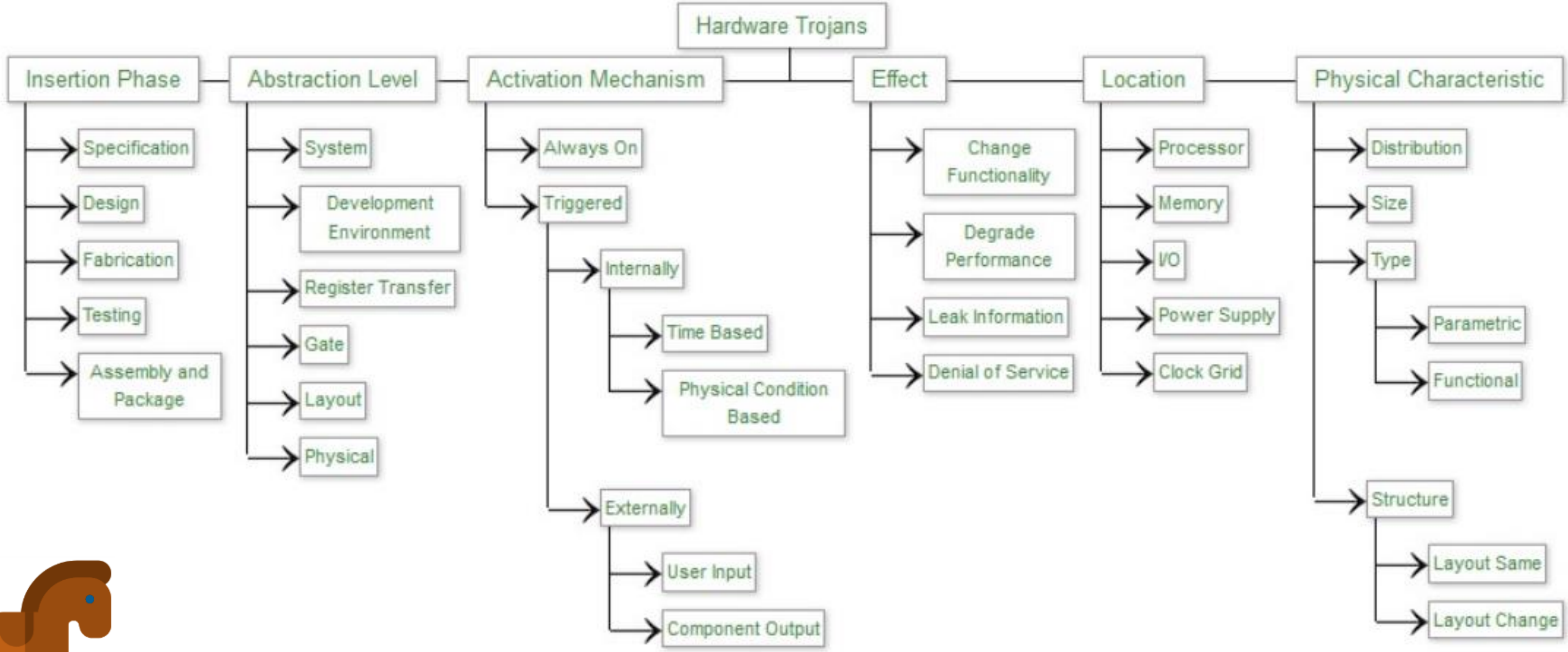
All stages of the supply chain are vulnerable



Source: DARPA

Hardware Trojans Taxonomy

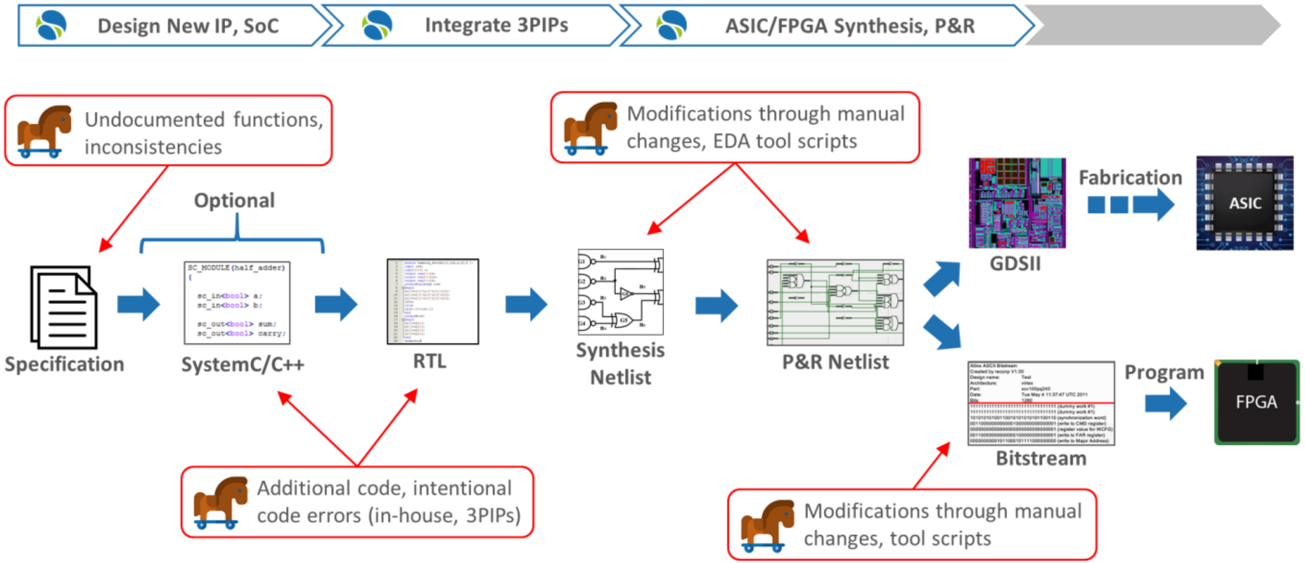
Anatomy: **triggers** on rare, hidden condition - delivers damaging **payload**



Source: Trust-Hub

Trust Assurance Challenges

Functional hardware Trojans are NOT bugs



IPs are complex and support a variety of configurations

Code review unlikely to spot malicious code

Functional verification targets bugs, not deliberately stealthy Trojans

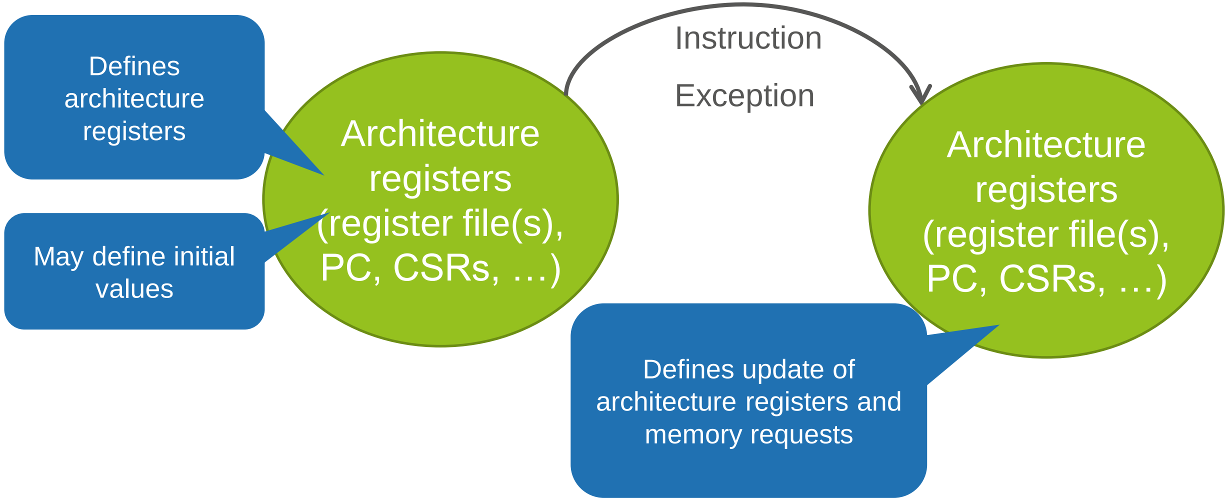
ASIC/FPGA/SoC developers need automated processes to increase confidence in IP trustworthiness

RISC-V ISA Specification

LW

31 … 20	19 … 15	14 … 12	11 … 7	6 … 0
imm	RS1	010	RD	0000011

```
op_a   = R[RS1]
addr   = op_a + imm
result = M[addr]
R[RD]  = result
```



Formalized User-Level ISA

- Captures effect of instructions on architecture state and output to memory
- Formalized in SystemVerilog Assertions (SVA)

ISA formalization excerpt for LW

```
32'bXXXXXXXXXXXXXXXXX010XXXXX0000011:
  decode.instr      = LW;
  decode.RS1.valid = 1'b1;
  decode.RD.valid  = 1'b1;
  decode.imm        = $signed(iw[31:20]);
  decode.mem        = 1'b1;
...
```

Pipelined Microarchitecture Verification

onespin

Various implementation choices for microarchitecture

- Specific pipeline length
- Forwarding paths to decode state and other stages
- Separate ICache/ DCache units with specific protocols
- Branch prediction for instruction fetch unit
- Stalling of pipeline stages or replay mechanism
- Out-of-order termination for long-latency instructions (like DIV, DCache miss)

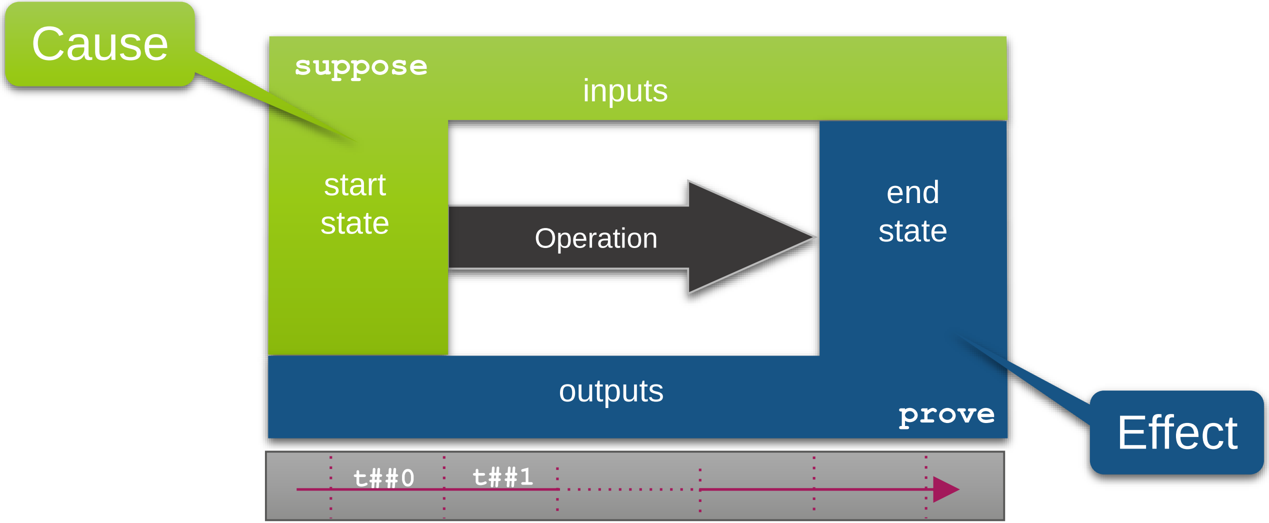
Verification links pipeline to sequential execution of instruction

- Capture full effect of one instruction/exception in one property
- Independent of preceding or succeeding instructions
- Next sequential instruction “starts” when leaving decode

Operational Assertions

Formally captures single DUV operation

- Suppose part describes cause – when does assertion apply
- Prove part specifies effect - intended behavior in that case



OneSpin's GapFreeVerification™

Proof that RISC-V assertions cover all possible core behaviors

RISC-V ISA expressed using OneSpin's Operational Assertions

- Standard SystemVerilog assertions following a strict template
- Assertions define results for each instruction
- Assertions cover instruction decode to completion

Enables automated unbounded proof of all assertions

Trojans or other unexamined logic cause failure of completeness proof

- Formal check of core's RTL against the RISC-V ISA
- Reveals any hidden logic that impacts core's functionality

GapFreeVerification

onespin

Achieving 100% functional coverage with SystemVerilog assertions (SVA)

Efficient Methodology | Industrial-Scale Technology

Rigorous Mathematical Foundation

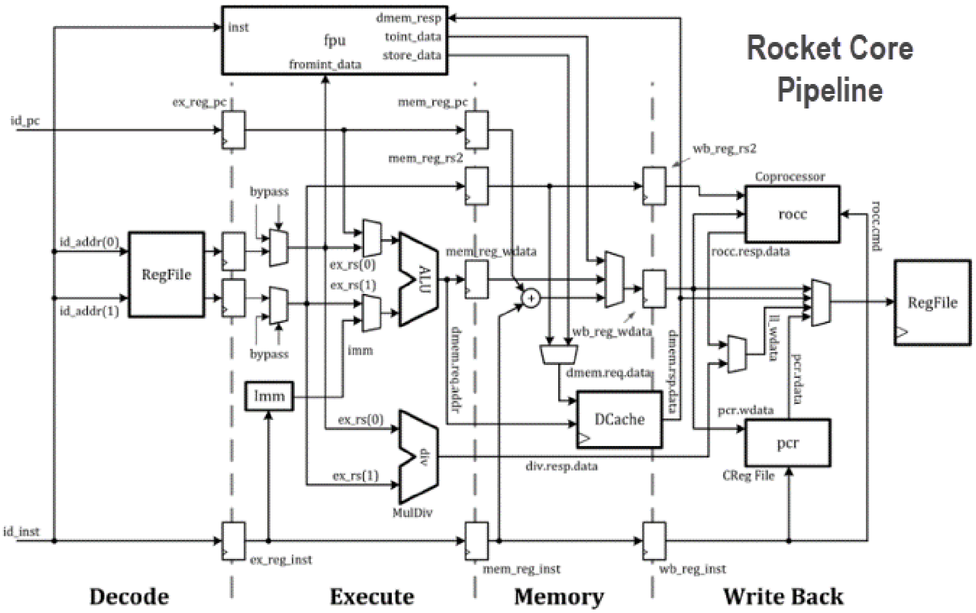
GapFreeVerification™ rigorous *completeness* definition

- A set of assertions *P* (formal testbench) is complete if every two designs *C1*, *C2* satisfying the assertions in *P* are sequentially equivalent (for every, arbitrarily long input trace, *C1* and *C2* produce the same output trace)

Many hardware trust issues are very hard-to-find bugs

- GapFreeVerification makes no distinction between *“malicious”* and *“naturally occurring”* bugs

Case Study: Rocket Core
Pipelined implementation



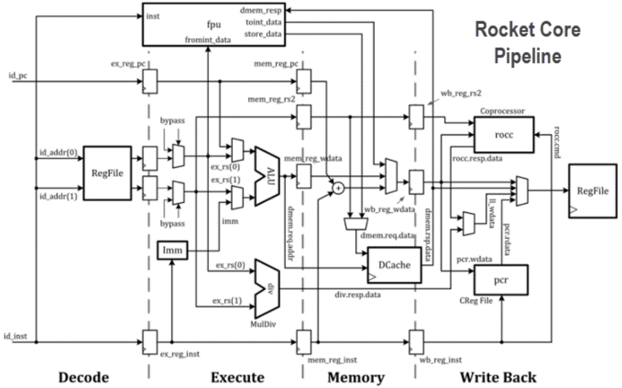
- 64-bit core
- 5-stage pipeline
- Single-issue, in-order pipeline
- Out-of-order completion of long latency instructions (e.g., DIV)
- Branch prediction
- Instruction replay

- Verified
- Taped-out multiple times

Case Study: Rocket Core

Project Scope

- Verification of core pipeline and CSRs
- Integer, compressed, atomics instructions
- Multiplication/division unit (control logic only)
- Exception handling and interrupt events
- Consistency of special instructions (LR, SC, fence)
- FPU excluded



Case Study: Rocket Core
Issues

Design issues reported*:

- **Issue 1757**: Jump instructions store different return PC - instruction fetch unit responsible to prevent this issue — **Acknowledged**
- **Issue 1752**: DIV result not written in register file — **Confirmed, fixed, closed**
- **Issue 1861**: Replay of illegal opcodes / generating memory accesses - Illegal opcodes not throwing an exception — **Under investigation**
- **Issue 1868:** Undocumented non-standard instruction - opcode 32'h30500073 / CEASE instruction — **Document update pending**
- **Issue 1949:** Undocumented CSR that reads back 0 — **Under investigation**

Highlights

- Each property returns a result in less than 10 minutes with helper assertions
- Each property returns a result in max. 5 hours w/o helper assertions
- Two hours runtime
- Unbounded proofs
- Low effort (few days) to set up

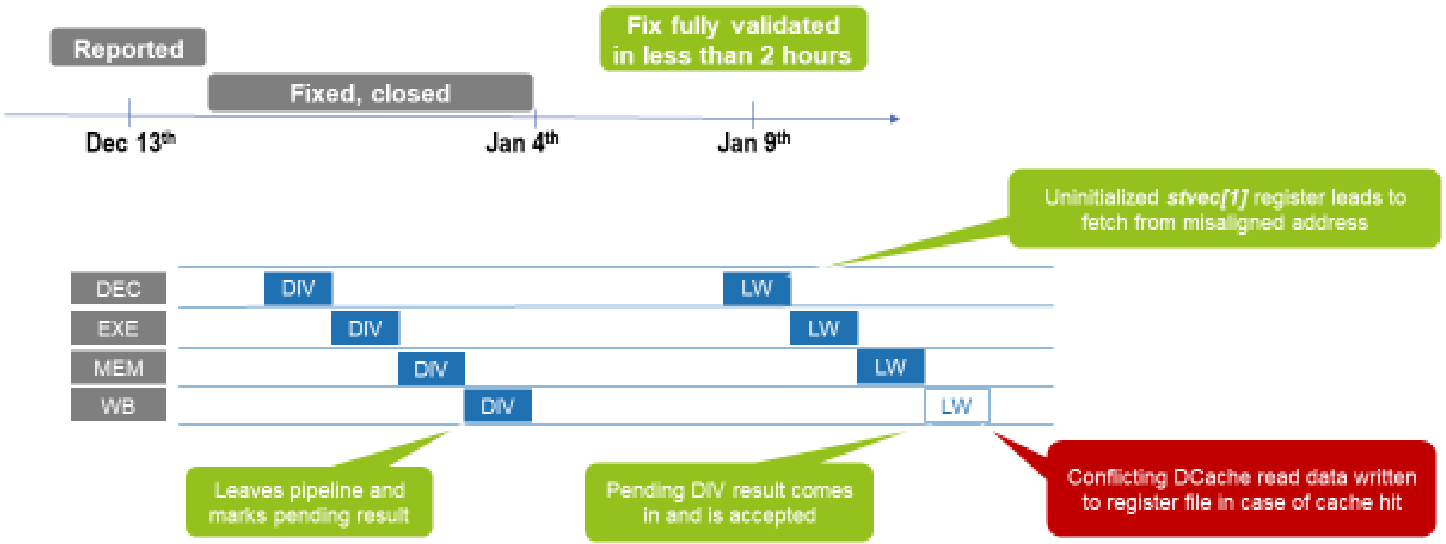
* https://github.com/freechipsproject/rocket-chip/issues/

Case Study: Rocket Core - GOMACTech 2019

Issues Found: DIV Issue (#1752)

Instruction result not written in register file

- Corner-case scenario – impossible to foresee – unrelated to any use-case scenario



2019 GOMACTech | 28 MAR 2019

DISTRIBUTION STATEMENT A. Approved for public release: distribution is unlimited. Approval ID: 88ABW-2019-2609. 13

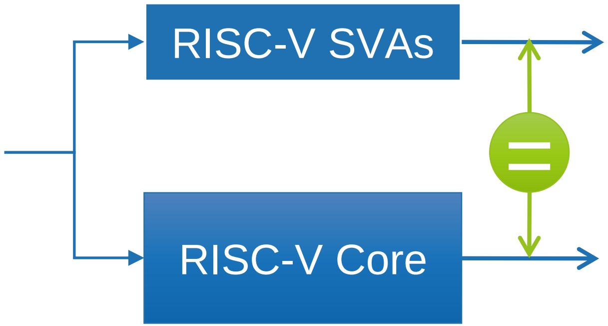
Summary
Industry's first RISC-V Integrity Verification Solution

RISC-V ISA formalized as SystemVerilog Assertions (SVAs)

- Decoupled from microarchitectural details

Enables 100% unbounded formal proof

- Proves that core is functionally correct
- Match or exceed quality of established ISA implementations
- Applies to 3PIP cores, open-source cores
- Applies to in-house custom extensions, optimized implementation



Detects hidden functions and hardware Trojans

- Achieves trust assurance

Additional Information

Learn more about RISC-V integrity verification

onespin.com/solutions/risc-v

Formal Verification of RISC-V Cores
onespin.com/blog

Complete Formal Verification of RISC-V Processor IPs for Trojan-Free Trusted ICs
Government Microcircuit Applications & Critical Technology (GOMACTech) Conference
Albuquerque, NM, USA, 2019

sergio.marchese@onespin.com
Drop me an email to request a copy of the GOMACTech paper and for additional information