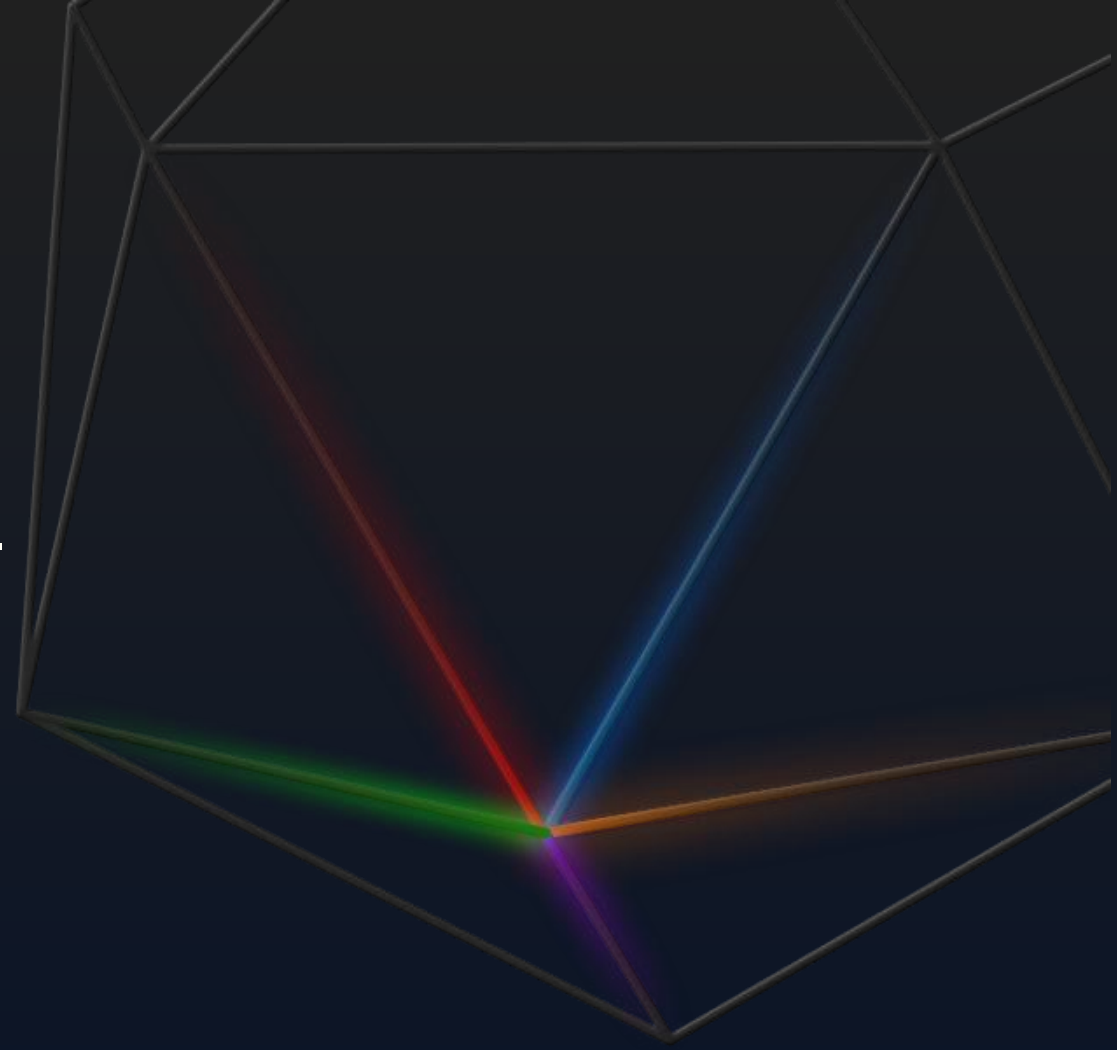
Industrial-Strength Formal Verification of RISC-V Processors

Dr. Ashish Darbari
Founder & CEO
Axiomise



axiomise®
predictable formal verification

Verification Trends

Wilson research reports 2024

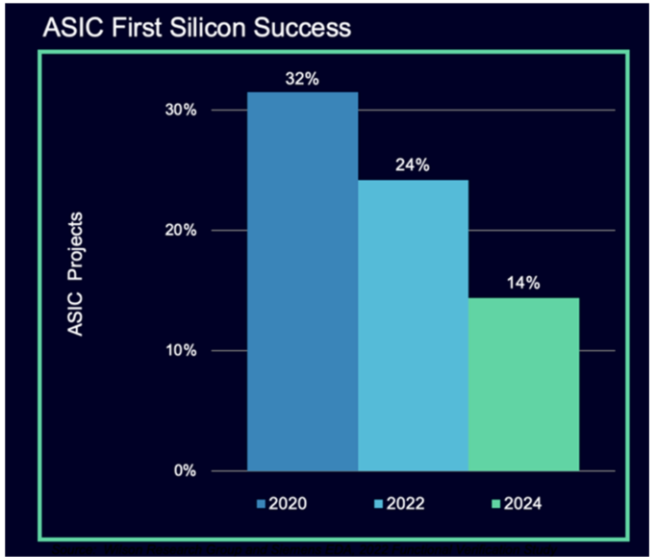


Fig. 1: Number of designs that are functionally correct and manufacturable is declining. Source: Siemens EDA/Wilson Research Group 2024 Functional Verification Study/DVCon

SYSTEMS & DESIGN

First-Time Silicon Success Plummets

564 Shares | 10 | 109 | 161

Number of designs that are late increases. Rapidly rising complexity is the leading cause, but tools, training, and workflows need to improve.

MARCH 27TH, 2025 - BY: **ED SPERLING**

First-time silicon success is falling sharply due to rising complexity, the need for more iterations as chipmakers shift from monolithic chips to multi-die assemblies, and an increasing amount of customization that makes design and verification more time-consuming.

TECHNICAL PAPERS

Scalable And Energy Efficient Solution For Hardware-Based ANNs (KAUST, NUS)

MARCH 30, 2025 **BY TECHNICAL PAPER LINK**

GPU Analysis Identifying Performance Bottlenecks That Cause Throughput Plateaus In Large-Batch Inference

MARCH 30, 2025 **BY TECHNICAL PAPER LINK**

Strategies For Reducing The Effective GaN/Diamond TBR

SPONSORS

SIEMENS | cadence | SYNOPSYS | KEYSIGHT | MOVELLUS | ARTERIS IP | ALPHAWAVE SEMI | axiomise | eliyan | BLUE CHEETAH ANALOG DESIGN

NEWSLETTER SIGNUP

Axiomise Solutions

Making formal normal by building a tool agnostic layer of solutions

Training

1-2-3-4 days

Instructor-led

On-demand

Primer

Comprehensive

Methodology

Tool independent

Consulting

Any duration

Training follow up

Methodology

Strategy

Planning

Review

Mentoring

Services

Any duration

Verification Strategy

Verification Planning

Execution

Sign-off

Weekly updates

Agile workflow

Apps

formalISA®, footprint™

Tool vendor independent

Push button, easy set-up

Find arch & uArch bugs

Functional verification

Safety, security, PPA

Bug presence & absence

axiomise®

predictable formal verification

Consulting & Services

Formal verification at scale – turnkey services delivered on some of the projects

We have been carrying out functional verification of designs with over 1 billion gates with formal.

Our abstraction-powered methodologies work can find bugs in new and existing designs.

We also help with customers' post-silicon issues on designs previously verified by others.

AI/ML accelerator

NoC

Ethernet Switch

Mixed-signal, low-power chip

Power controller

DMA controller

Multi-threaded processor

Bus bridges (AXI/CHI/OCP/TileLink)

Cache sub-systems

GPU blocks

I2C/USB/HDMI/I2S

Why is processor verification hard?

Why bugs escape to silicon?

A Holistic Approach is Missing

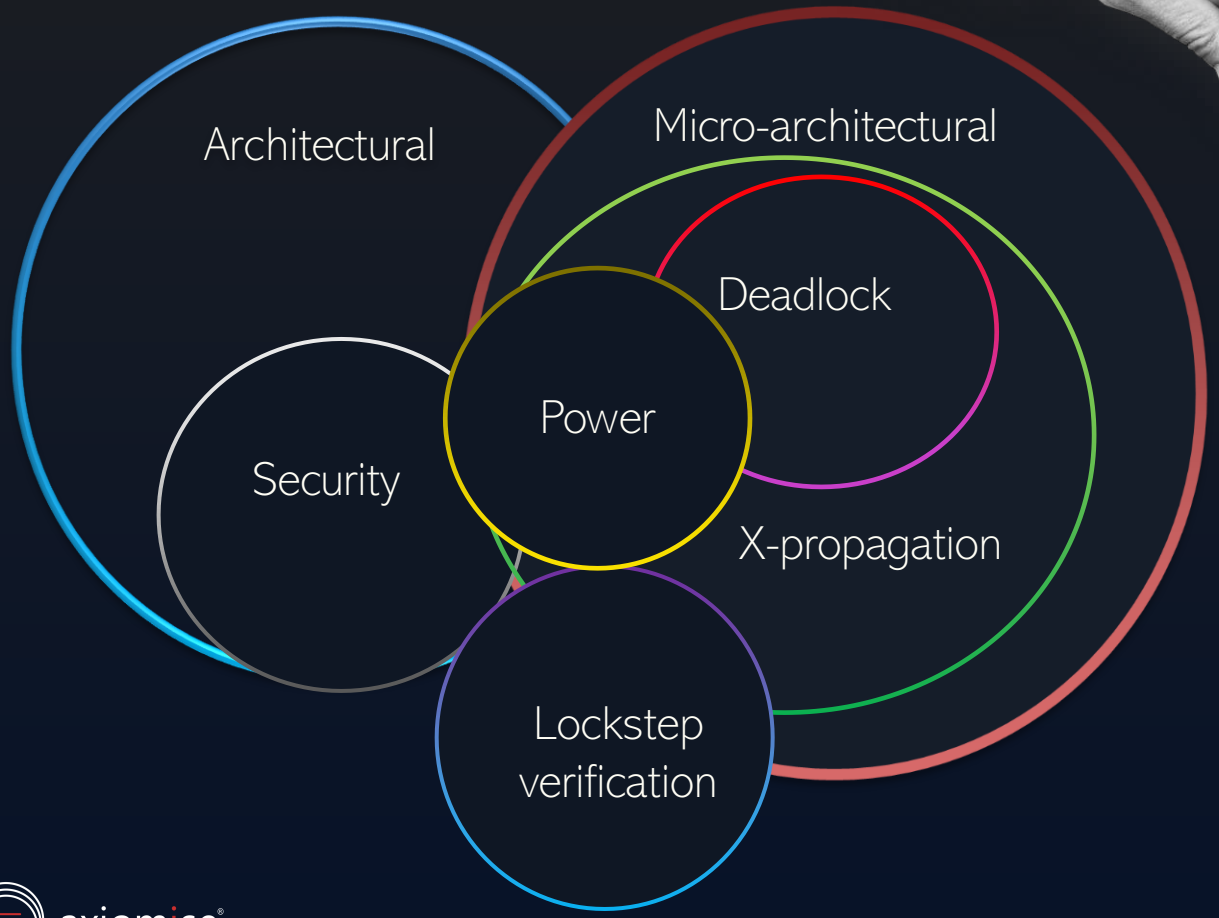
A unifying perspective is missing

ARCHITECTURE

DESIGN/MICROARCHITECTURE

NETLIST

SILICON



Modern-day Processors

Massively optimised

Pipelining	Interlocking	Forwarding
Branches	Jumps	Exceptions
Stalls	Interrupts	Debug
Extensions	Clock gating	Arithmetic
Power	Safety	Security

Complex Control and Data Dependencies

And the cores have in-order or out-of-order behaviour?

Branches:

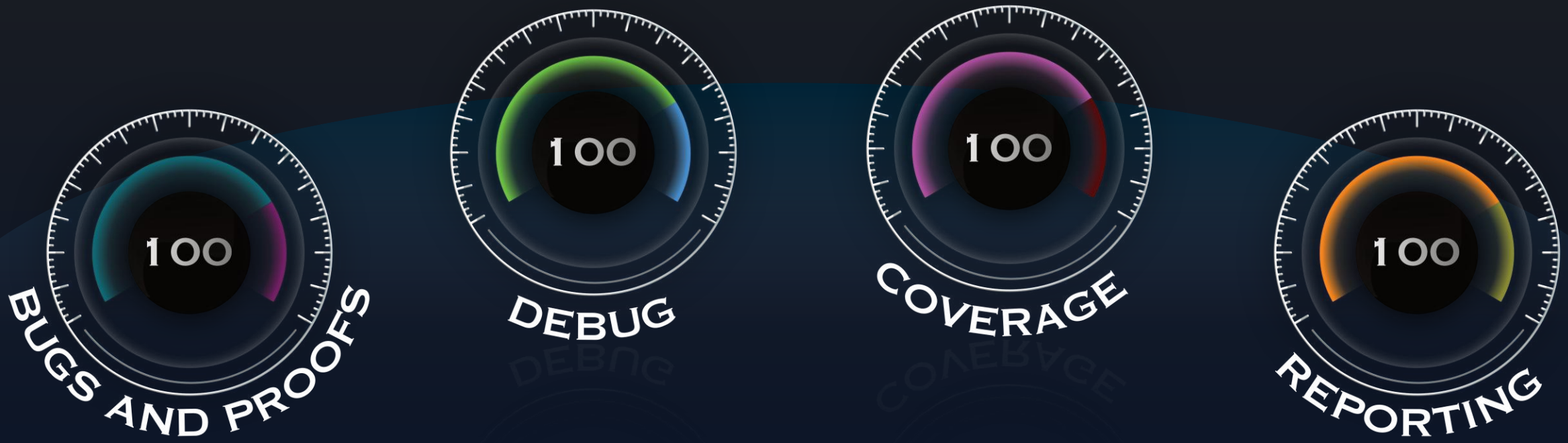
- Speculative branches
- Forward jumps, Backward jumps, Page size jumps, Page boundary jumps, Jumps across pages (same or different pages)

Back-to-back memory operations:

- Cache hit & cache misses
- Write-through stores
- Cache bypasses, atomics and cache coherency

formalISA®

Making formal normal for RISC-V



Accelerating debug and sign-off for custom designs using exhaustive formal

Our Formal RISC-V Solution

Enables adoption of formal methods more widely

1. No test case to write
2. No manual checker to write
3. No verification code to be written
4. Exhaustively prove that all ISA instructions work as expected under all conditions

What goes in our APP?

1. Your RISC-V core
2. Set up file
3. Coverage specification

What comes out?

Exhaustive proofs that "mathematically" prove under all conditions:

- ✓ Each instruction in the ISA works always as expected
- ✓ Scenarios specified in the coverage specification can "always" happen
- ✓ Visualize that scenarios in the coverage specification "can happen"

axiomise®
predictable formal verification

formalISA

How different pieces connect?

ULTRA formalISA® Predictable RISC-V formal verification

RISC-V®

ISA

RISC-V®

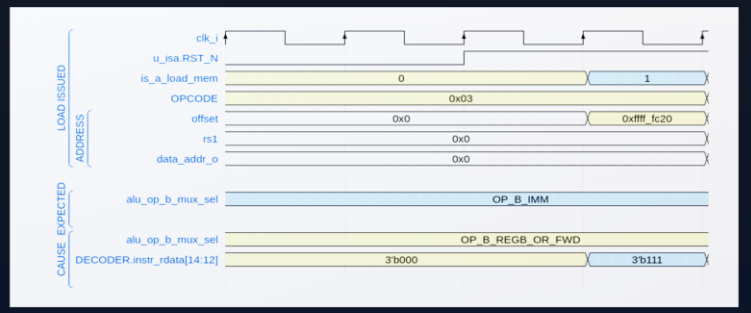
RTL

VENDOR NEUTRAL

USE ANY FORMAL TOOL YOU LIKE

ULTRA formalISA® Predictable RISC-V formal verification

Formal Tool

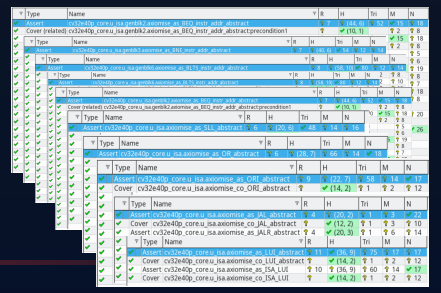


Setup

FAIL

OR

PASS



ibex

Complete democracy – use any tool you like

✓		ibex_core.u_isa.axiomise__ISA_JALR
✓		ibex_core.u_isa.axiomise__ISA_LUI
✓		ibex_core.u_isa.axiomise__ISA_OR
✓		ibex_core.u_isa.axiomise__ISA_ORI
✓		ibex_core.u_isa.axiomise__ISA_SLL
✓		ibex_core.u_isa.axiomise__ISA_SLLI
✓		ibex_core.u_isa.axiomise__ISA_SLTSI_SET_TO_0
✓		ibex_core.u_isa.axiomise__ISA_SLTSI_SET_TO_1
✓		ibex_core.u_isa.axiomise__ISA_SLTS_SET_TO_0
✓		ibex_core.u_isa.axiomise__ISA_SLTS_SET_TO_1
✓		ibex_core.u_isa.axiomise__ISA_SLTUI_SET_TO_0
✓		ibex_core.u_isa.axiomise__ISA_SLTUI_SET_TO_1
✓		ibex_core.u_isa.axiomise__ISA_SLTU_SET_TO_0
✓		ibex_core.u_isa.axiomise__ISA_SLTU_SET_TO_1
✓		ibex_core.u_isa.axiomise__ISA_SRA
✓		ibex_core.u_isa.axiomise__ISA_SRAI
✓		ibex_core.u_isa.axiomise__ISA_SRL
✓		ibex_core.u_isa.axiomise__ISA_SRLI
✓		ibex_core.u_isa.axiomise__ISA_SUB
✓		ibex_core.u_isa.axiomise__ISA_XOR
✓		ibex_core.u_isa.axiomise__ISA_XORI

☐	⊙	u_isa.axiomise__ISA_SRLI
☐	⊙	u_isa.axiomise__ISA_ADDI
☐	⊙	u_isa.axiomise__ISA_SRL
☐	⊙	u_isa.axiomise__ISA_JALR
☐	⊙	u_isa.axiomise__ISA_XORI
☐	⊙	u_isa.axiomise__ISA_BEQ
☐	⊙	u_isa.axiomise__ISA_BNE
☐	⊙	u_isa.axiomise__ISA_SLTSI_SET_TO_1
☐	⊙	u_isa.axiomise__ISA_BGEU
☐	⊙	u_isa.axiomise__ISA_SLTSI_SET_TO_0
☐	⊙	u_isa.axiomise__ISA_XOR
☐	⊙	u_isa.axiomise__ISA_SLTUI_SET_TO_1
☐	⊙	u_isa.axiomise__ISA_OR
☐	⊙	u_isa.axiomise__ISA_ORI
☐	⊙	u_isa.axiomise__ISA_SLTUI_SET_TO_0
☐	⊙	u_isa.axiomise__ISA_ANDI
☐	⊙	u_isa.axiomise__ISA_SLLI
☐	⊙	u_isa.axiomise__ISA_SLTS_SET_TO_1
☐	⊙	u_isa.axiomise__ISA_SLTU_SET_TO_1
☐	⊙	u_isa.axiomise__ISA_BLTS
☐	⊙	u_isa.axiomise__ISA_SRAI
☐	⊙	u_isa.axiomise__ISA_SLTS_SET_TO_0
☐	⊙	u_isa.axiomise__ISA_SLTU_SET_TO_0
☐	⊙	u_isa.axiomise__ISA_JAL
☐	⊙	u_isa.axiomise__ISA_LUI
☐	⊙	u_isa.axiomise__ISA_BGES
☐	⊙	u_isa.axiomise__ISA_SRA
☐	⊙	u_isa.axiomise__ISA_BLTU
☐	⊙	u_isa.axiomise__ISA_ADD
☐	⊙	u_isa.axiomise__ISA_AND
☐	⊙	u_isa.axiomise__ISA_SUB
☐	⊙	u_isa.axiomise__ISA_SLL
☐	⊙	u_isa.axiomise__ISA_JAL_ret_address

✓	Assert	ibex_core.u_isa.inv_block_jal[4].axiomise_inv_isa_jal
✓	Assert	ibex_core.u_isa.inv_block_jalr[0].axiomise_inv_isa_jalr
✓	Assert	ibex_core.u_isa.inv_block_auipc[0].axiomise_inv_isa_auipc
✓	Assert	ibex_core.u_isa.inv_block_auipc[4].axiomise_inv_isa_auipc
✓	Assert	ibex_core.u_isa.inv_block_auipc[8].axiomise_inv_isa_auipc
✓	Assert	ibex_core.u_isa.inv_block_auipc[12].axiomise_inv_isa_auipc
✓	Assert	ibex_core.u_isa.inv_block_auipc[20].axiomise_inv_isa_auipc
✓	Assert	ibex_core.u_isa.inv_block_auipc[24].axiomise_inv_isa_auipc
✓	Assert	ibex_core.u_isa.inv_block_auipc[28].axiomise_inv_isa_auipc
✓	Assert	ibex_core.u_isa.axiomise__ISA_JAL
✓	Assert	ibex_core.u_isa.axiomise__ISA_JAL_ret_address
✓	Assert	ibex_core.u_isa.axiomise__ISA_JALR
✓	Assert	ibex_core.u_isa.axiomise__ISA_BEQ
✓	Assert	ibex_core.u_isa.axiomise__ISA_BNE
✓	Assert	ibex_core.u_isa.axiomise__ISA_BGEU
✓	Assert	ibex_core.u_isa.axiomise__ISA_BLTU
✓	Assert	ibex_core.u_isa.axiomise__ISA_LUI
✓	Assert	ibex_core.u_isa.axiomise__ISA_ADDI
✓	Assert	ibex_core.u_isa.axiomise__ISA_XORI
✓	Assert	ibex_core.u_isa.axiomise__ISA_ORI
✓	Assert	ibex_core.u_isa.axiomise__ISA_ANDI
✓	Assert	ibex_core.u_isa.axiomise__ISA_BLTS
✓	Assert	ibex_core.u_isa.axiomise__ISA_BGES
✓	Assert	ibex_core.u_isa.axiomise__ISA_SLTSI_SET_TO_1
✓	Assert	ibex_core.u_isa.axiomise__ISA_SLTSI_SET_TO_0
✓	Assert	ibex_core.u_isa.axiomise__ISA_SLTUI_SET_TO_1
✓	Assert	ibex_core.u_isa.axiomise__ISA_SLTUI_SET_TO_0
✓	Assert	ibex_core.u_isa.axiomise__ISA_SLLI
✓	Assert	ibex_core.u_isa.axiomise__ISA_SRLI
✓	Assert	ibex_core.u_isa.axiomise__ISA_SRAI
✓	Assert	ibex_core.u_isa.axiomise__ISA_SLL
✓	Assert	ibex_core.u_isa.axiomise__ISA_SRL

sva/u_isa/axiomise__ISA_ADDI	pass (4)	hold
sva/u_isa/axiomise__ISA_AND	pass (4)	hold
sva/u_isa/axiomise__ISA_ANDI	pass (4)	hold
sva/u_isa/axiomise__ISA_BEQ	pass (4)	hold
sva/u_isa/axiomise__ISA_BGES	pass (4)	hold
sva/u_isa/axiomise__ISA_BGEU	pass (4)	hold
sva/u_isa/axiomise__ISA_BLTS	pass (5)	hold
sva/u_isa/axiomise__ISA_BLTU	pass (5)	hold
sva/u_isa/axiomise__ISA_BNE	pass (5)	hold
sva/u_isa/axiomise__ISA_JAL	pass (4)	hold
sva/u_isa/axiomise__ISA_JAL_ret_address	pass (5)	hold
sva/u_isa/axiomise__ISA_JALR	pass (4)	hold
sva/u_isa/axiomise__ISA_LUI	pass (4)	hold
sva/u_isa/axiomise__ISA_OR	pass (4)	hold
sva/u_isa/axiomise__ISA_ORI	pass (4)	hold
sva/u_isa/axiomise__ISA_SLL	pass (4)	hold
sva/u_isa/axiomise__ISA_SLLI	pass (4)	hold
sva/u_isa/axiomise__ISA_SLTS_SET_TO_0	pass (4)	hold
sva/u_isa/axiomise__ISA_SLTS_SET_TO_1	pass (5)	hold
sva/u_isa/axiomise__ISA_SLTSI_SET_TO_0	pass (4)	hold
sva/u_isa/axiomise__ISA_SLTSI_SET_TO_1	pass (4)	hold
sva/u_isa/axiomise__ISA_SLTU_SET_TO_0	pass (4)	hold
sva/u_isa/axiomise__ISA_SLTU_SET_TO_1	pass (5)	hold
sva/u_isa/axiomise__ISA_SLTUI_SET_TO_0	pass (4)	hold
sva/u_isa/axiomise__ISA_SLTUI_SET_TO_1	pass (4)	hold
sva/u_isa/axiomise__ISA_SRA	pass (4)	hold
sva/u_isa/axiomise__ISA_SRAI	pass (4)	hold
sva/u_isa/axiomise__ISA_SRL	pass (4)	hold
sva/u_isa/axiomise__ISA_SRLI	pass (4)	hold
sva/u_isa/axiomise__ISA_SUB	pass (4)	hold
sva/u_isa/axiomise__ISA_XOR	pass (4)	hold
sva/u_isa/axiomise__ISA_XORI	pass (4)	hold
sva/u_isa/axiomiseH__ISA_AUIPC	pass (4)	hold
sva/u_isa/axiomiseH__ISA_JALR2	pass (4)	hold

SYNOPSYS®

Mentor® A Siemens Business

cādence

onespin

axiomise® predictable formal verification

Formal verification

Bugs and Proofs

axiomise®

predictable formal verification

formalISA®

Making formal normal for RISC-V

	ibex	zeroriscy	cv32e40p	WARP-V			Cheriot-ibex
Pipeline stages	2-stage	2-stage	4-stage	6-stage	4-stage	2-stage	2
No. of issues	65	77	5	30	30	30	6
Previously verified	Yes	Yes	No	Yes	Yes	Yes	Yes
How was it previously verified?	Simulation	Simulation	Simulation & Formal	Formal	Formal	Formal	Simulation & Formal
Time taken to find issues	< 30 seconds	< 30 seconds	< 30 seconds	< 30 seconds	< 30 seconds	< 30 seconds	<1 min
Nature of analysis and issues	Microarchitectural Deadlocks and Architectural	Microarchitectural Deadlocks and Architectural	Architectural	Architectural	Architectural	Architectural	Corner-case bugs
When was the issue found?	2019	2019	2020	2021	2021	2021	2024

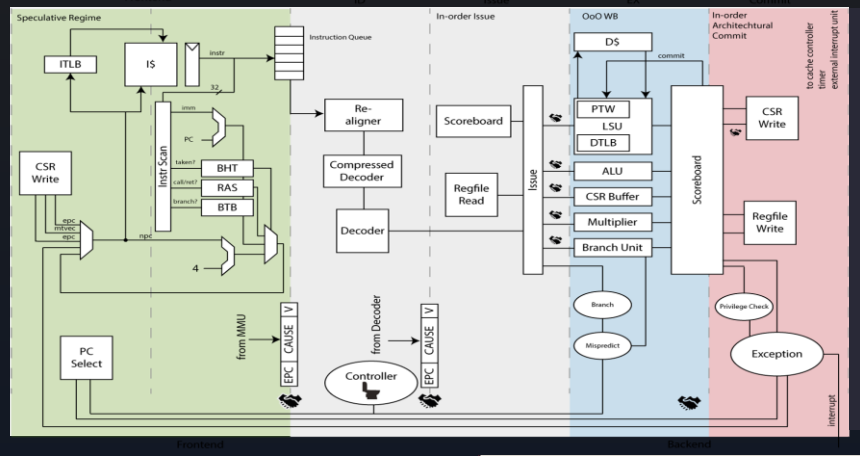
cv32e40p

32-bit, 4-stage in-order pipeline

Type	Name	R	H	Tri	M	N
Assert	cv32e40p_core.u_isa.genblk2.axiomise_as_BEQ_instr_addr_abstract	7	(44, 6)	52	15	18
Cover (related)	cv32e40p_core.u_isa.genblk2.axiomise_as_BEQ_instr_addr_abstract:precondition1		(10, 1)		2	8
Assert	cv32e40p_core.u_isa.genblk3.axiomise_as_BNE_instr_addr_abstract	7	(40, 6)	54	12	14
Assert	cv32e40p_core.u_isa.genblk6.axiomise_as_BLTS_instr_addr_abstract	8	(58, 10)	80	12	14
Cover (related)	cv32e40p_core.u_isa.genblk6.axiomise_as_BLTS_instr_addr_abstract:precondition1		(12, 2)	1	2	8
Cover	cv32e40p_core.u_isa.genblk6.axiomise_co_BLTS_instr_addr_abstract		(14, 2)	1	2	10
Assert	cv32e40p_core.u_isa.genblk6.axiomise_as_ISA_BLTS_instr_addr	9	(66, 24)	76	8	16
Cover (related)	cv32e40p_core.u_isa.genblk6.axiomise_as_ISA_BLTS_instr_addr:precondition1		(12, 2)			
Assert	cv32e40p_core.u_isa.genblk6.axiomise_as_BLTS_instr_addr_abstract	8	(58, 10)	80	12	14
Cover (related)	cv32e40p_core.u_isa.genblk6.axiomise_as_BLTS_instr_addr_abstract:precondition1		(12, 2)	1	2	8
Cover	cv32e40p_core.u_isa.genblk6.axiomise_co_BLTS_instr_addr_abstract		(14, 2)	1	2	10
Assert	cv32e40p_core.u_isa.genblk6.axiomise_as_ISA_BLTS_instr_addr	9	(66, 24)	76	8	16
Cover (related)	cv32e40p_core.u_isa.genblk6.axiomise_as_ISA_BLTS_instr_addr:precondition1		(12, 2)			
Cover	cv32e40p_core.u_isa.genblk6.axiomise_co_ISA_BLTS_instr_addr		(14, 2)	1		5
Assert	cv32e40p_core.u_isa.axiomise_as_BLTS_abstract	8	(66, 11)	80	14	16
Cover (related)	cv32e40p_core.u_isa.axiomise_as_BLTS_abstract:precondition1		(12, 2)	1	2	8
Cover	cv32e40p_core.u_isa.axiomise_co_BLTS_abstract	2	(16, 3)	1	2	12
Assert	cv32e40p_core.u_isa.axiomise_as_ORI_abstract	9	(22, 7)	58	14	17
Cover	cv32e40p_core.u_isa.axiomise_co_ORI_abstract		(14, 2)	1	2	12
Assert	cv32e40p_core.u_isa.axiomise_as_JAL_abstract	4	(20, 2)	1	3	22
Cover	cv32e40p_core.u_isa.axiomise_co_JAL_abstract		(12, 2)	1	3	10
Assert	cv32e40p_core.u_isa.axiomise_as_JALR_abstract	4	(20, 3)	1	6	14
Cover	cv32e40p_core.u_isa.axiomise_co_JALR_abstract		(12, 2)	1	3	10
Assert	cv32e40p_core.u_isa.axiomise_as_LUI_abstract	11	(36, 9)	75	17	17
Cover	cv32e40p_core.u_isa.axiomise_co_LUI_abstract		(14, 2)	1	2	12
Assert	cv32e40p_core.u_isa.axiomise_as_ISA_LUI	10	(36, 9)	60	14	17
Cover	cv32e40p_core.u_isa.axiomise_co_ISA_LUI		(14, 2)	1	2	12

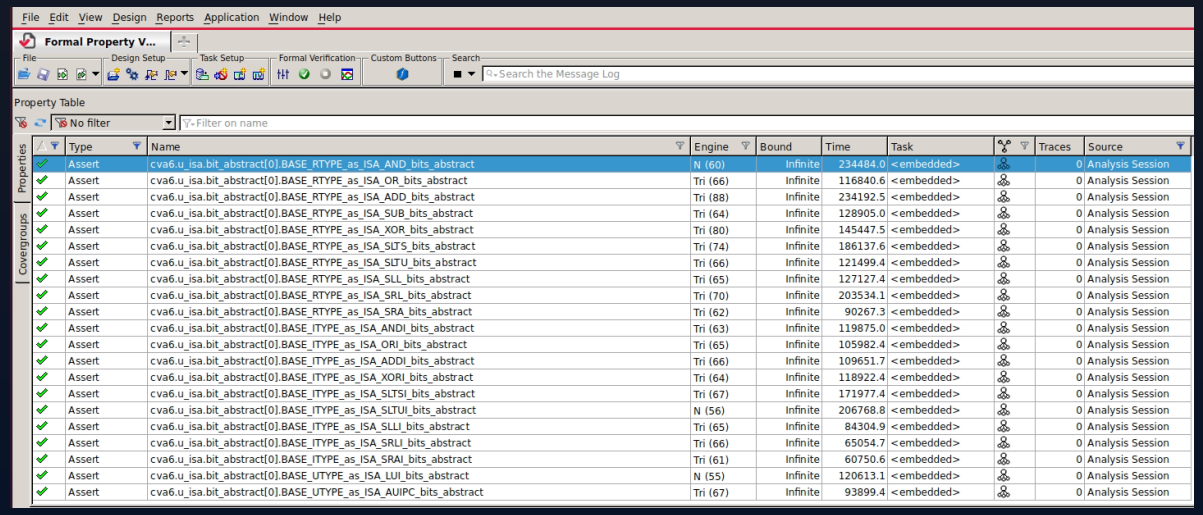
CVA6

64-bit six-stage, in-order issue, out-of-order execution, in-order commit



From the OPENHW Group Page

CVA6 is a 6-stage, single issue, in-order CPU which implements the 64-bit RISC-V instruction set. It fully implements I, M, A and C extensions as specified in Volume I: User-Level ISA V 2.3 as well as the draft privilege extension 1.10. It implements three privilege levels M, S, U to fully support a Unix-like operating system. Furthermore, it is compliant to the draft external debug spec 0.13. It has configurable size, separate TLBs, a hardware PTW and branch-prediction (branch target buffer and branch history table). The primary design goal was on reducing critical path length.



Type	Name	Engine	Bound	Time	Task	Traces	Source
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_AND_bits_abstract	N (60)	Infinite	234484.0	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_OR_bits_abstract	Tri (66)	Infinite	116840.6	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_ADD_bits_abstract	Tri (88)	Infinite	234192.5	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_SUB_bits_abstract	Tri (64)	Infinite	128905.0	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_XOR_bits_abstract	Tri (80)	Infinite	145447.5	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_SLTS_bits_abstract	Tri (74)	Infinite	186137.6	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_SLTU_bits_abstract	Tri (66)	Infinite	121499.4	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_SLL_bits_abstract	Tri (65)	Infinite	127127.4	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_SRL_bits_abstract	Tri (70)	Infinite	203534.1	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_SRA_bits_abstract	Tri (62)	Infinite	90267.3	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_ANDI_bits_abstract	Tri (63)	Infinite	119875.0	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_ORI_bits_abstract	Tri (65)	Infinite	105982.4	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_ADDI_bits_abstract	Tri (66)	Infinite	109651.7	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_XORI_bits_abstract	Tri (64)	Infinite	118922.4	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_SLTSI_bits_abstract	Tri (67)	Infinite	171977.4	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_SLTUI_bits_abstract	N (56)	Infinite	206768.8	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_SLLI_bits_abstract	Tri (65)	Infinite	84304.9	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_SRLI_bits_abstract	Tri (66)	Infinite	65054.7	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_SRAI_bits_abstract	Tri (61)	Infinite	60750.6	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_UTYPE_as_ISA_LUI_bits_abstract	N (55)	Infinite	120613.1	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_UTYPE_as_ISA_AUIPC_bits_abstract	Tri (67)	Infinite	93899.4	<embedded>	0	Analysis Session

axiomise®
predictable formal verification

CVA6

64-bit six-stage, in-order issue, out-of-order execution, in-order commit

File Edit View Design Reports Application Window Help

Formal Property V...

File | Design Setup | Task Setup | Formal Verification | Custom Buttons | Search

Search the Message Log

Property Table

No filter | Filter on name

Properties | Covergroups

Type	Name	Engine	Bound	Time	Task	Traces	Source
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_AND_bits_abstract	N (60)	Infinite	234484.0	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_OR_bits_abstract	Tri (66)	Infinite	116840.6	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_ADD_bits_abstract	Tri (88)	Infinite	234192.5	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_SUB_bits_abstract	Tri (64)	Infinite	128905.0	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_XOR_bits_abstract	Tri (80)	Infinite	145447.5	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_SLTS_bits_abstract	Tri (74)	Infinite	186137.6	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_SLTU_bits_abstract	Tri (66)	Infinite	121499.4	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_SLL_bits_abstract	Tri (65)	Infinite	127127.4	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_SRL_bits_abstract	Tri (70)	Infinite	203534.1	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_RTYPE_as_ISA_SRA_bits_abstract	Tri (62)	Infinite	90267.3	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_ANDI_bits_abstract	Tri (63)	Infinite	119875.0	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_ORI_bits_abstract	Tri (65)	Infinite	105982.4	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_ADDI_bits_abstract	Tri (66)	Infinite	109651.7	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_XORI_bits_abstract	Tri (64)	Infinite	118922.4	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_SLTSI_bits_abstract	Tri (67)	Infinite	171977.4	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_SLTUI_bits_abstract	N (56)	Infinite	206768.8	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_SLLI_bits_abstract	Tri (65)	Infinite	84304.9	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_SRLI_bits_abstract	Tri (66)	Infinite	65054.7	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_ITYPE_as_ISA_SRAI_bits_abstract	Tri (61)	Infinite	60750.6	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_UTYPE_as_ISA_LUI_bits_abstract	N (55)	Infinite	120613.1	<embedded>	0	Analysis Session
Assert	cva6.u_isa.bit_abstract[0].BASE_UTYPE_as_ISA_AUIPC_bits_abstract	Tri (67)	Infinite	93899.4	<embedded>	0	Analysis Session

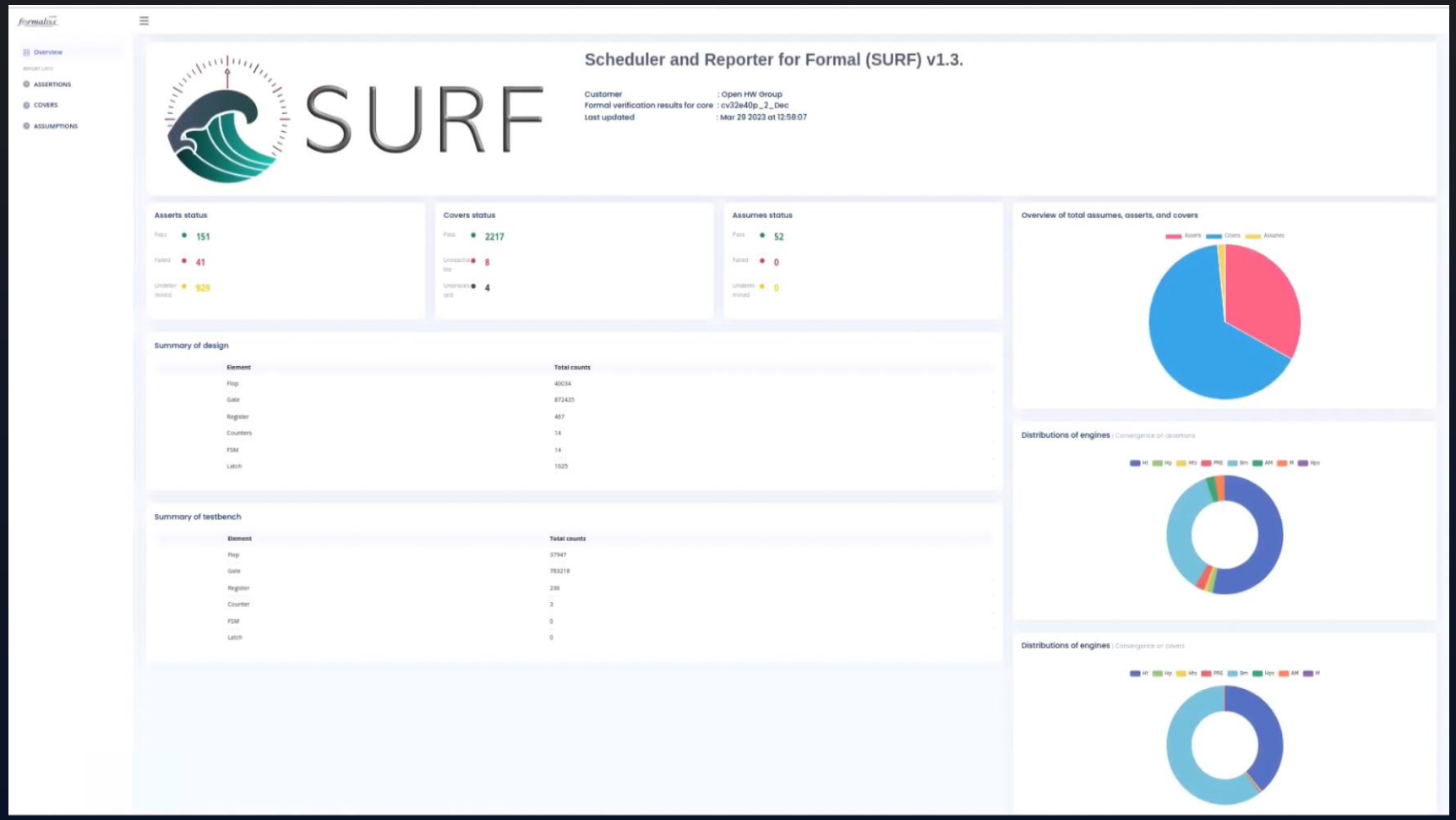
Reporting

Scheduler and Reporter for Formal SURF

axiomise®
predictable formal verification

SURF Dashboard

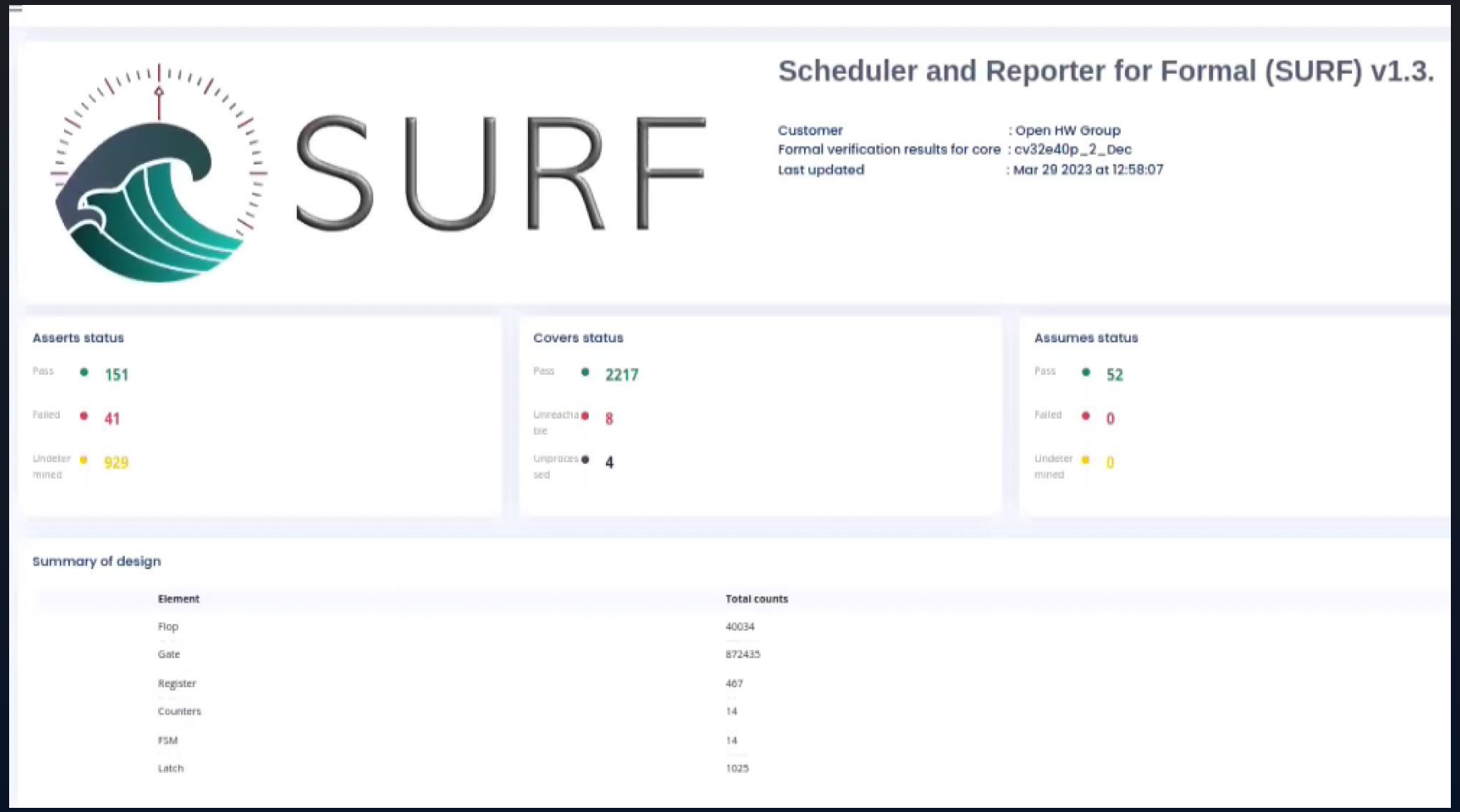
RISC-V



axiomise®
predictable formal verification

SURF Dashboard

RISC-V



SURF Dashboard

Example reporting bugs

formalise

Overview

REPORT LISTS

ASSERTIONS

COVERS

ASSUMPTIONS

Covers

No.	Instruction type	Property label	Cover status	Proof time	Engine
1.	BASE ITYPE	co_ISA_ADDI_abstract	COVERED	0.08	Hts
2.	BASE ITYPE	co_ISA_ADDI_abstract_JG	COVERED	0.38	Hts
3.	BASE ITYPE	co_ISA_ADDI	COVERED	0.07	Bm
4.	BASE ITYPE	co_ISA_ADDI_JG	COVERED	0.68	Ht
5.	BASE ITYPE	co_ISA_XORI_abstract	COVERED	0.07	Bm
6.	BASE ITYPE	co_ISA_XORI_abstract_JG	COVERED	0.72	Ht
7.	BASE ITYPE	co_ISA_XORI	COVERED	0.07	Bm
8.	BASE ITYPE	co_ISA_XORI_JG	COVERED	0.72	Ht
9.	BASE ITYPE	co_ISA_ORI_abstract	COVERED	0.07	Bm
10.	BASE ITYPE	co_ISA_ORI_abstract_JG	COVERED	0.76	Ht
11.	BASE ITYPE	co_ISA_ORI	COVERED	0.07	Bm
12.	BASE ITYPE	co_ISA_ORI_JG	COVERED	0.76	Ht
13.	BASE ITYPE	co_ISA_ANDI_abstract	COVERED	0.07	Bm
14.	BASE ITYPE	co_ISA_ANDI_abstract_JG	COVERED	0.80	Ht
15.	BASE ITYPE	co_ISA_ANDI	COVERED	0.07	Bm
16.	BASE ITYPE	co_ISA_ANDI_JG	COVERED	0.80	Ht
17.	BASE ITYPE	co_ISA_SLTI_SET_TO_1_abstract	COVERED	0.07	Bm
18.	BASE ITYPE	co_ISA_SLTI_SET_TO_1_abstract_JG	COVERED	0.83	Ht
19.	BASE ITYPE	co_ISA_SLTI_SET_TO_1	COVERED	0.07	Bm
20.	BASE ITYPE	co_ISA_SLTI_SET_TO_1_JG	COVERED	0.83	Ht
21.	BASE ITYPE	co_ISA_SLTI_SET_TO_0_abstract_JG	COVERED	0.87	Ht
22.	BASE ITYPE	co_ISA_SLTI_SET_TO_0_JG	COVERED	0.87	Ht
23.	BASE ITYPE	co_ISA_SLTI_SET_TO_0_abstract	COVERED	0.07	Bm
24.	BASE ITYPE	co_ISA_SLTI_SET_TO_0	COVERED	0.07	Bm
25.	BASE ITYPE	co_ISA_SLTIU_SET_TO_1_abstract	COVERED	0.07	Bm
26.	BASE ITYPE	co_ISA_SLTIU_SET_TO_1_abstract_JG	COVERED	0.91	Ht
27.	BASE ITYPE	co_ISA_SLTIU_SET_TO_1	COVERED	0.07	Bm
28.	BASE ITYPE	co_ISA_SLTIU_SET_TO_1_JG	COVERED	0.91	Ht
29.	BASE ITYPE	co_ISA_SLTIU_SET_TO_0_abstract	COVERED	0.07	Bm
30.	BASE ITYPE	co_ISA_SLTIU_SET_TO_0_abstract_JG	COVERED	0.95	Ht

SURF Dashboard

Example reporting bugs

formalisa

Overview

REPORT LISTS

ASSERTIONS

COVERS

ASSUMPTIONS

Asserts

No.	Instruction type	Property label	Assert status	Preconditions	Proof time	Engine	Bug	Mnemonic	Specifications
1.	BASE ITYPE	as_ISA_ADDI_abstract	UNDETERMINED	COVERED	42783.41	Tri	Maybe	add rd rs1 imm12	x[rd] = x[rs1] + imm12. Adds imm12 to register x[rs1] and writes the result to x[rd], arithmetic overflow is ignored.
2.	BASE ITYPE	as_ISA_ADDI	UNDETERMINED	COVERED	86063.33	Bm	Maybe	add rd rs1 imm12	x[rd] = x[rs1] + imm12. Adds imm12 to register x[rs1] and writes the result to x[rd], arithmetic overflow is ignored.
3.	BASE ITYPE	as_ISA_XORI_abstract	PROVEN	COVERED	90.19	M	No	xori rd rs1 imm12	x[rd] = x[rs1] ^ imm12. Computes the bitwise XOR of registers x[rs1] and imm12 and writes the result to x[rd].
4.	BASE ITYPE	as_ISA_XORI	PROVEN	COVERED	73.07	M	No	xori rd rs1 imm12	x[rd] = x[rs1] ^ imm12. Computes the bitwise XOR of registers x[rs1] and imm12 and writes the result to x[rd].
5.	BASE ITYPE	as_ISA_ORI_abstract	PROVEN	COVERED	67.32	M	No	ori rd rs1 imm12	x[rd] = x[rs1] | imm12. Computes the bitwise OR of registers x[rs1] and imm12 and writes the result to x[rd].
6.	BASE ITYPE	as_ISA_ORI	PROVEN	COVERED	86.84	M	No	ori rd rs1 imm12	x[rd] = x[rs1] | imm12. Computes the bitwise OR of registers x[rs1] and imm12 and writes the result to x[rd].
7.	BASE ITYPE	as_ISA_ANDI_abstract	PROVEN	COVERED	99.02	M	No	andi rd rs1 imm12	x[rd] = x[rs1] & imm12. Computes the bitwise OR of registers x[rs1] and imm12 and writes the result to x[rd].
8.	BASE ITYPE	as_ISA_ANDI	PROVEN	COVERED	66.73	M	No	andi rd rs1 imm12	x[rd] = x[rs1] & imm12. Computes the bitwise OR of registers x[rs1] and imm12 and writes the result to x[rd].
9.	BASE ITYPE	as_ISA_SLTI_SET_TO_1_abstract	PROVEN	COVERED	79.05	Tri	No	slti rd rs1 imm12	x[rd] = x[rs1] <s imm12. Compares x[rs1] and x[rs2] as signed numbers, and writes 1 to x[rd] if x[rs1] is smaller, and 0 if not.
10.	BASE ITYPE	as_ISA_SLTI_SET_TO_1	PROVEN	COVERED	35.67	Tri	No	slti rd rs1 imm12	x[rd] = x[rs1] <s imm12. Compares x[rs1] and x[rs2] as signed numbers, and writes 1 to x[rd] if x[rs1] is smaller, and 0 if not.

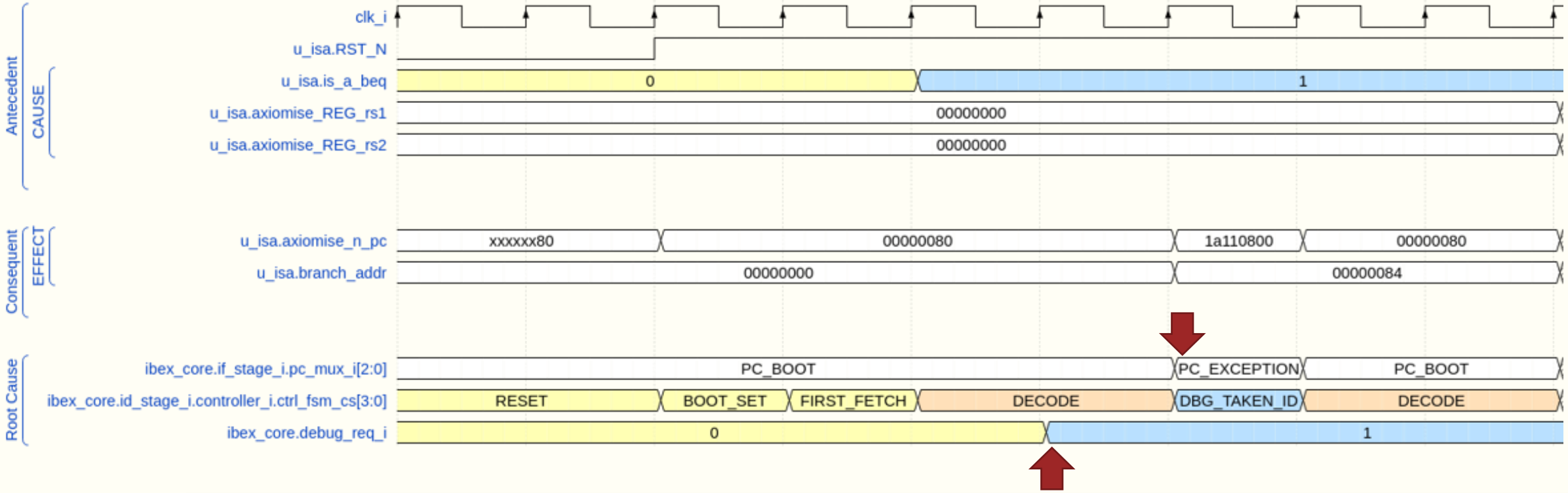
Anatomy of bugs

Processor bugs caught by *formalISA*

axiomise®
predictable formal verification

BEQ Failure

Functional verification - ibex



Bug caused due to incoming debug request on the debug interface when the controller is in the DECODE state.

Nothing in the design to take care of such requests, causing the PC to be not updated correctly.

BEQ Failure

Functional verification - ibex

Only seen when debug arrives and the controller FSM is in the DECODE state.

Precise timing of arrival of debug makes this bug really hard to catch in dynamic simulation.

Formal catches it in seconds in 7 cycles!

Communication on ibex

Corner-case bugs confirmed

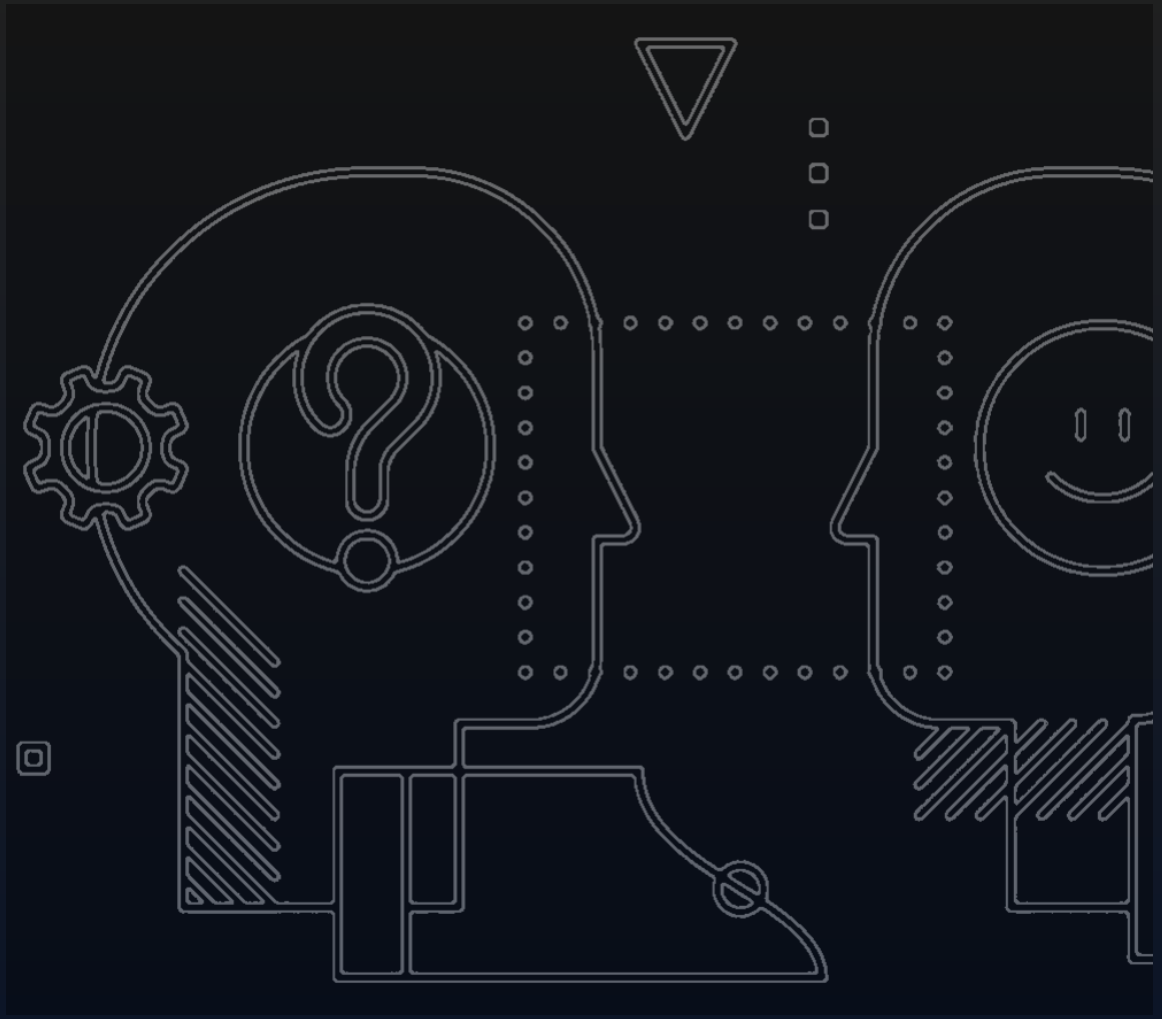
Hi Philip

Further, on my analysis on enabling full debug to see which arch checks pass/fail, it appears that the extent of damage goes beyond the six instructions I initially reported. Actually, other than SLLI, SRAI, and SRLI, all the checks fail.

The pattern is similar to what I reported in the six VCDs that I already assigned to you. If you want I can send you the tarball but don't want to overwhelm you. We're talking about investigating 51 additional property failures besides the six initially reported on branch instructions.

Instructions that fail are:

- BEQ, BGES, BLTS, BLTU, BNE (all reported before)
- ADDI, ADD, AND, ANDI, AUIPC, LUI, JAL, JALR, OR, ORI, XOR, XORI, SUB, SLTUI, STUI, SRL, SLL, SLTI, SLTS, SLTSI
- All checks that establish correctness for different variants (BYTE, WORD, HALF WORD) of LOAD/STORES (aligned/misaligned)



Open **BEQ not working as expected** #2

darbaria opened this issue on 26 Jun · 3 comments

imphil commented 2 hours ago

Hi **@darbaria** we have recently reworked the controller in this area and I'd expect this bug to be fixed. Can you test if it is still present?

axiomise®
predictable formal verification

https://github.com/darbaria/axiomise-warpv-formal-6-stage/issues

WARP-V

Six stage pipelined processor with a range of bugs

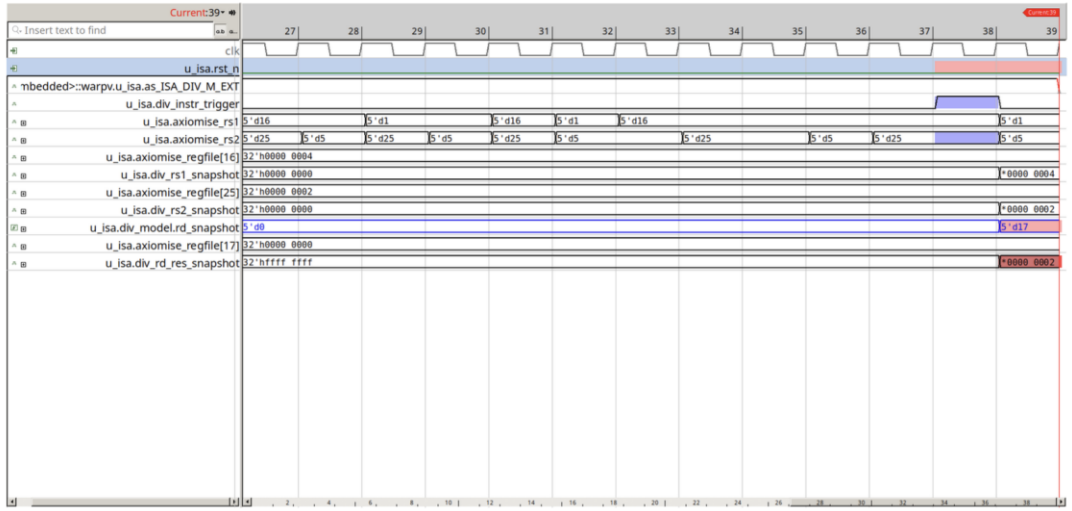
DIV Instruction not working correctly #29

Open **shivanishah269** opened this issue on 6 Jun 2021 · 0 comments

shivanishah269 commented on 6 Jun 2021 Collaborator

Page 44 of RISC-V ISA mandates "DIV: Divides x[rs1] by x[rs2] rounding towards 0, treating the values as signed numbers and writes the quotient to x[rd]"

Our checker fails showing that the updates did not happen in cycle 38 to the register 17 in response to a prior **div** instruction detected in cycle 37. x[16] is divided by x[25] and rd is 17. We expect x[17] to be 2 as x[16] is 4 and x[25] is 2, but it isn't.



Assignees
stevehoover
shivanishah269

Labels
bug

Projects
None yet

Milestone
No milestone

Development
No branches or pull requests

2 participants

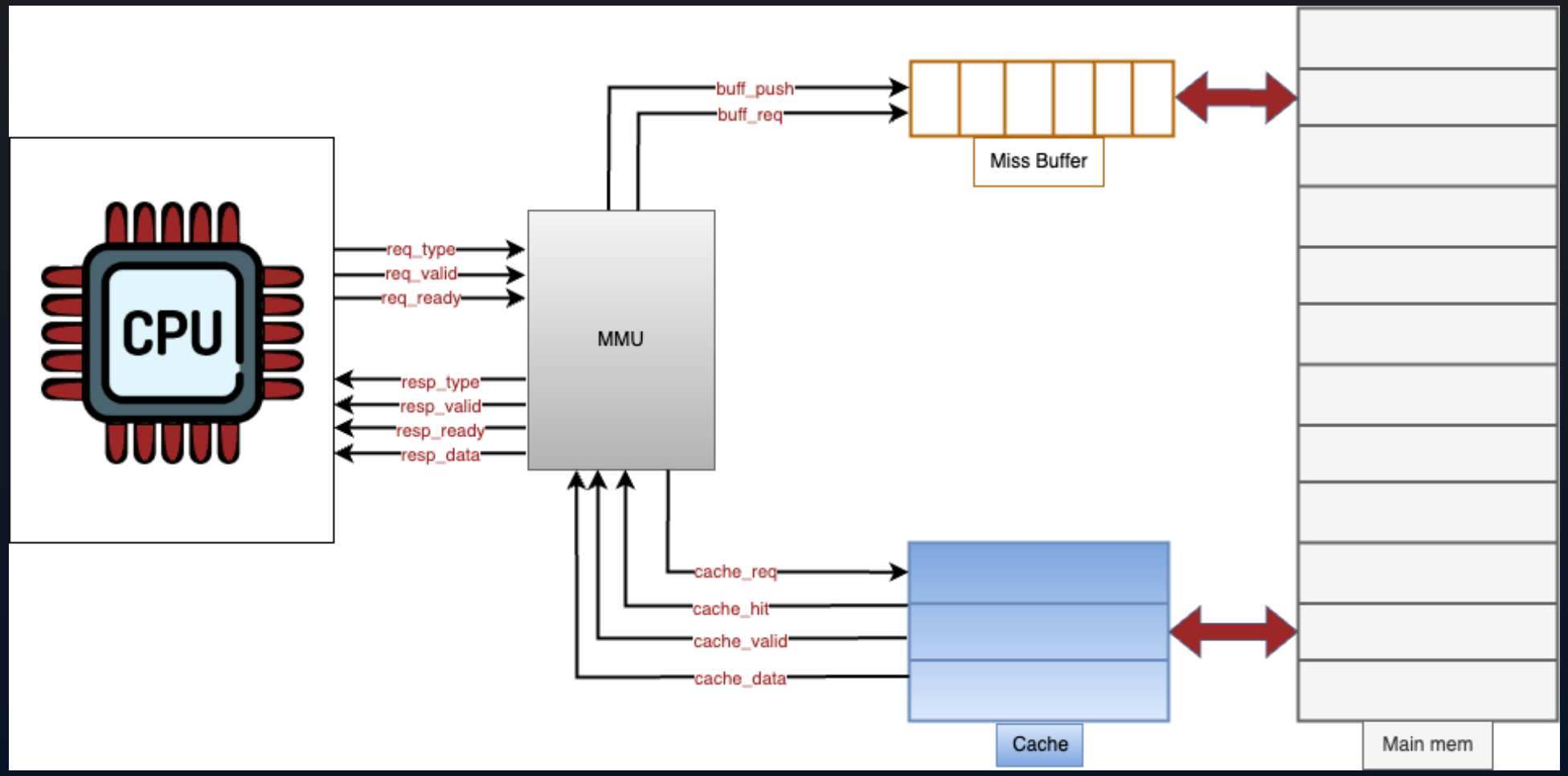
axiomise
predictable formal verification

Memory subsystem

Caught by our *formalISA*

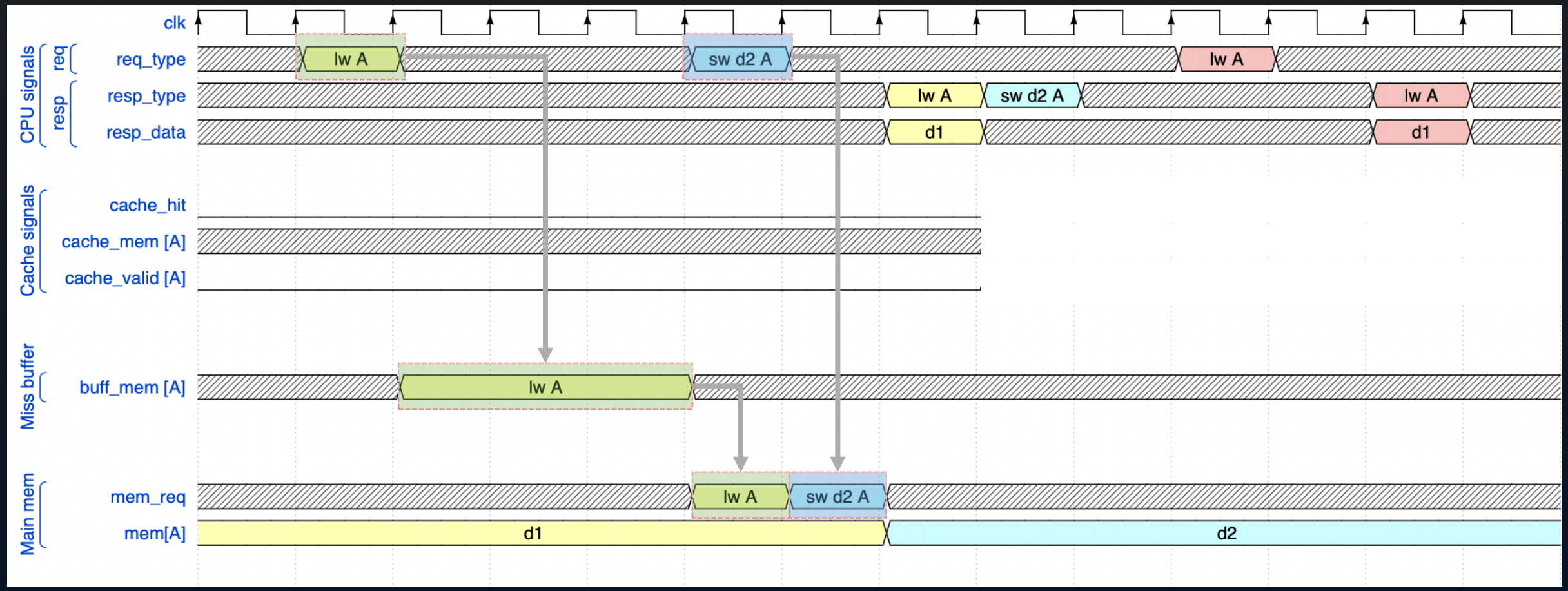
Cache Issues

Incorrect validation of cache line due to bypass store



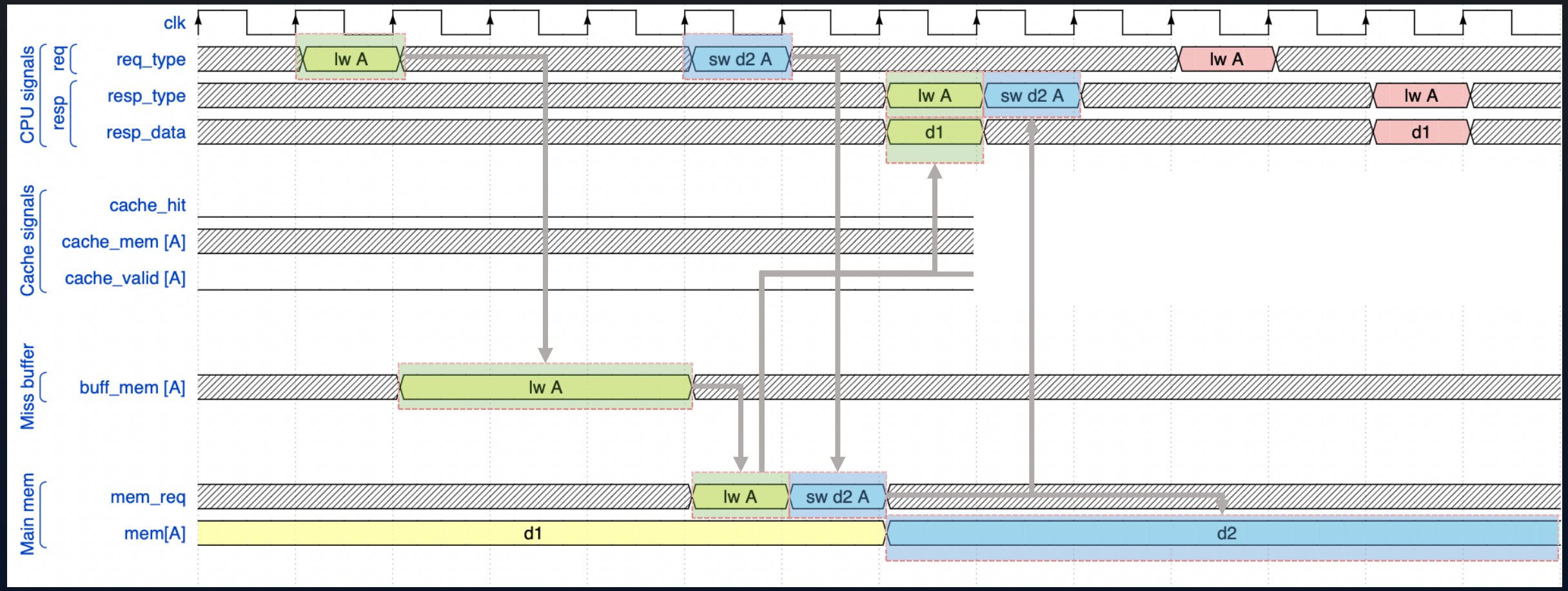
Cache Issues

Incorrect validation of cache line due to bypass store



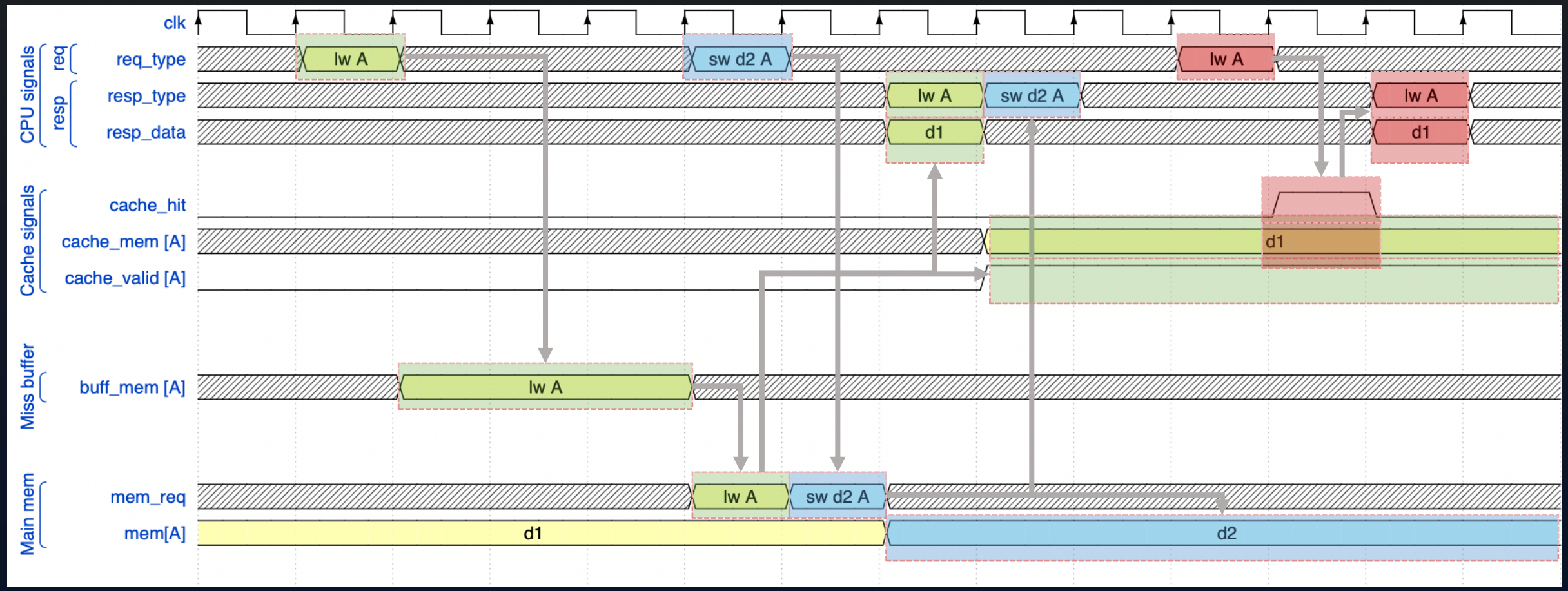
Cache Issues

Incorrect validation of cache line due to bypass store



Cache Issues

Incorrect validation of cache line due to bypass store



CHERIOT-IBEX

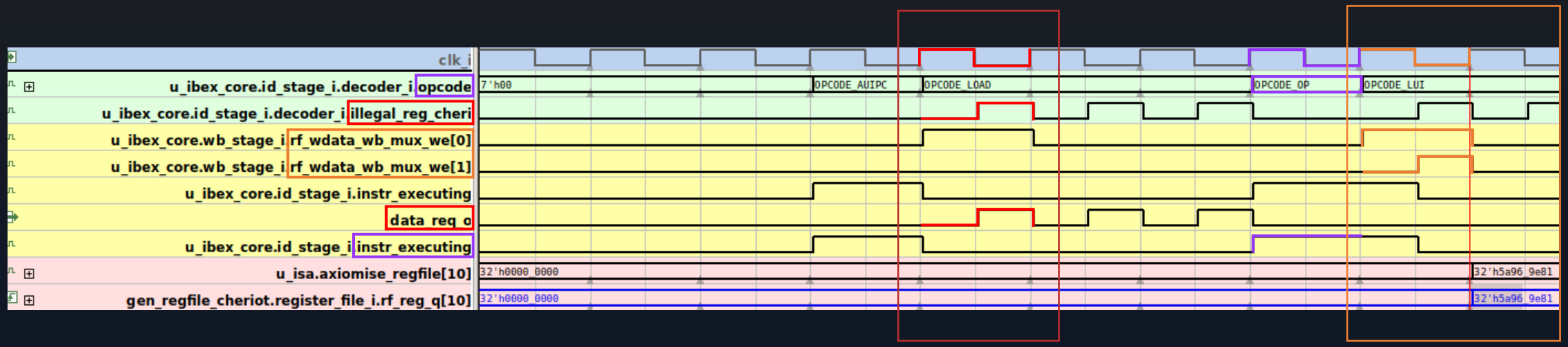
Corner case bugs

axiomise®
predictable formal verification

Illegal instruction handling

Verified in September 2024

The illegal instruction affected the execution of the valid instruction that followed it.



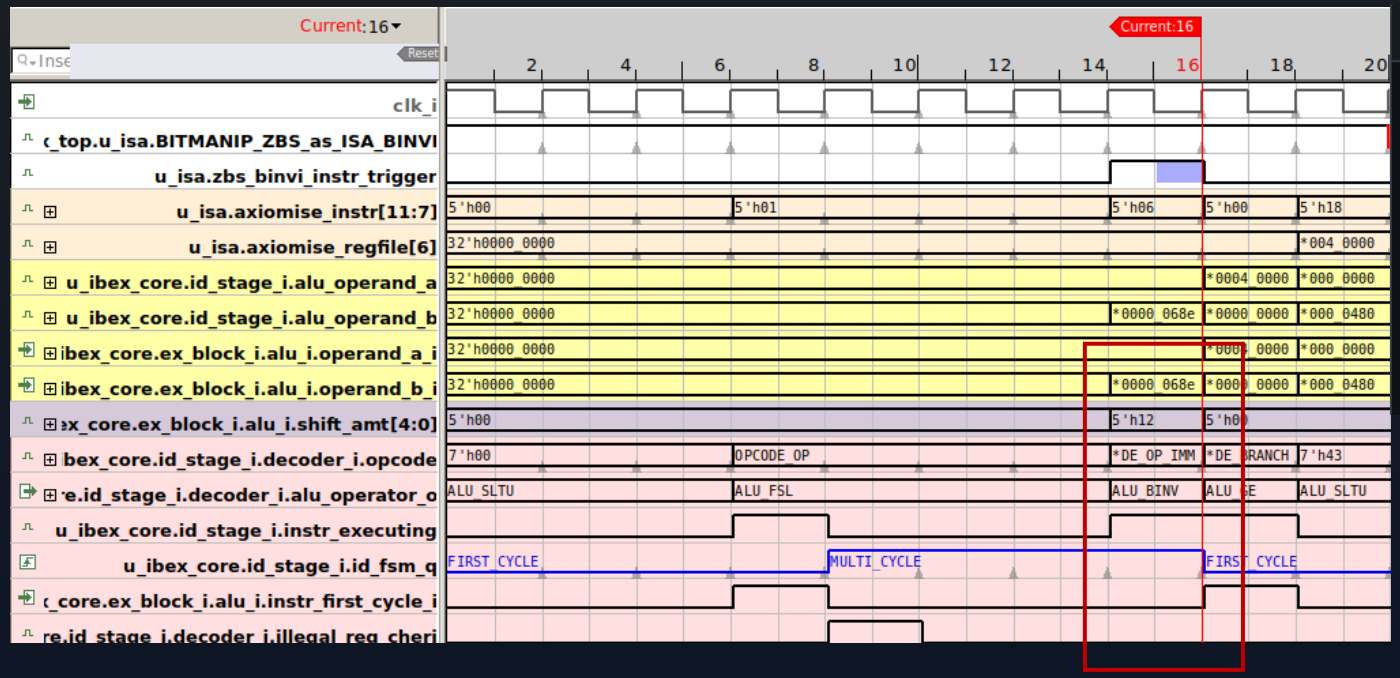
Issues

- Sending the illegal instruction request to the memory.
- Wasted execution power.
- Invalid data in the register file and subsequently in memory.

The illegal load instruction affected the execution of the valid AND (or any R-TYPE) instruction that followed it.

Illegal instruction handling – bit manipulation

After the first bug fix, bit manipulations instructions were broken



kliuMsft on Oct 15, 2024 Contributor

Looking further into the issue, the culprit seems to be that the id_fsm_d logic can't handle exception being issued in the 2nd half of a multi-cycle instruction. Specifically, the illegal_reg_cheri results in an EX stage exception but instr_kill is only raised in the 2nd half of a bit manipulation instruction (when rs3 is accessed). In this case multicycle_done is never issued and thus id_fsm_q will not updated properly.

@GregAC do you plan to keep supporting the bit instructions with rs3? if so I can try fix the behavior in cheriot-ibex. You may want to take a look at the upstream ibex implementation as well.

https://github.com/microsoft/cheriot-ibex/issues/51

Decoder Issues

Very interesting set of issues

Observed Behavior

https://github.com/microsoft/cheriot-ibex/issues/47

The RISC-V ISA for bit manipulation states that for the 32-bit implementations of the ISA, for the RORI instruction the bit 25 needs to be 1'b0. However, in the implementation we have downloaded from the Github on 17 July 2024, an instruction could be considered to be a valid RORI despite the bit 25 being 1'b1. It means there are two ways to decode the same instruction which means it is prone to security vulnerabilities and it does not comply with the ISA.

Decoder Issues

Very interesting set of issues

itsomaia on Oct 15, 2024 https://github.com/microsoft/cheriot-ibex/issues/47 Author

Thank you for your response. We also discovered that bit 26, which isn't defined as reserved and must be set to 0, may vary and still be identified as a valid BINVI, BCLRI, or BSETI instruction. These cases have been filed separately as issues #48, #49, and #50.

kliuMsft added a commit that references this issue on Oct 19, 2024

`fixed bit extension instr decoding (issue #47)` 94aeb07

kliuMsft on Oct 19, 2024 Contributor

@itsomaia, commit 94aeb07 should fix the issue along with issues #48, #49, and #50. Please verify. Thanks.

Observed Behavior

The RISC-V ISA for bit manipulation states that for the 32-bit implementations of the ISA, for the RORI instruction the bit 25 needs to be 1'b0. However, in the implementation we have downloaded from the Github on 17 July 2024, an instruction could be considered to be a valid RORI despite the bit 25 being 1'b1. It means there are two ways to decode the same instruction which means it is prone to security vulnerabilities and it does not comply with the ISA.

footprint™

Area analyser for silicon

Footprint – Area analyser for silicon

Cheriot-ibex

Design gate count	Design flop count	Redundant components	Estimated redundant gates per category
303,737	14,723	Counter: 3	Counter:768
		Register: 313	Register:16,440
		Array: 23	Array:7872

Summary

Formal methods is a necessity not a nice-to-have

Bugs are a natural consequence of implementing design

Bugs caught late in the design cycle result in very expensive fixes and catastrophic failures

Formal == efficient bug hunting & exhaustive proofs right at the time of design bring up

Architectural validation must employ formal verification to build "proofs" of bug absence

Architects, designers, verification engineers can all use *formalISA* without any FV experience

- Find bugs, build proofs, obtain inter-operable coverage model for use in simulation and other formal tools
- Use any formal verification tool of your choice

Find corner-case bugs as well as build exhaustive proofs

$\vdash$ $\models$ $\equiv$

axiomise®

predictable formal verification

www.axiomise.com

CONSULTING & SERVICES

TRAINING

CUSTOM SOLUTIONS

info@axiomise.com