Addressing Formal Verification Challenges with GenAI Technology and RISC-V Solutions

Raja Mahadevan

Ravindra Aneja

SYNOPSYS®

accellera SYSTEMS INITIATIVE™

2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

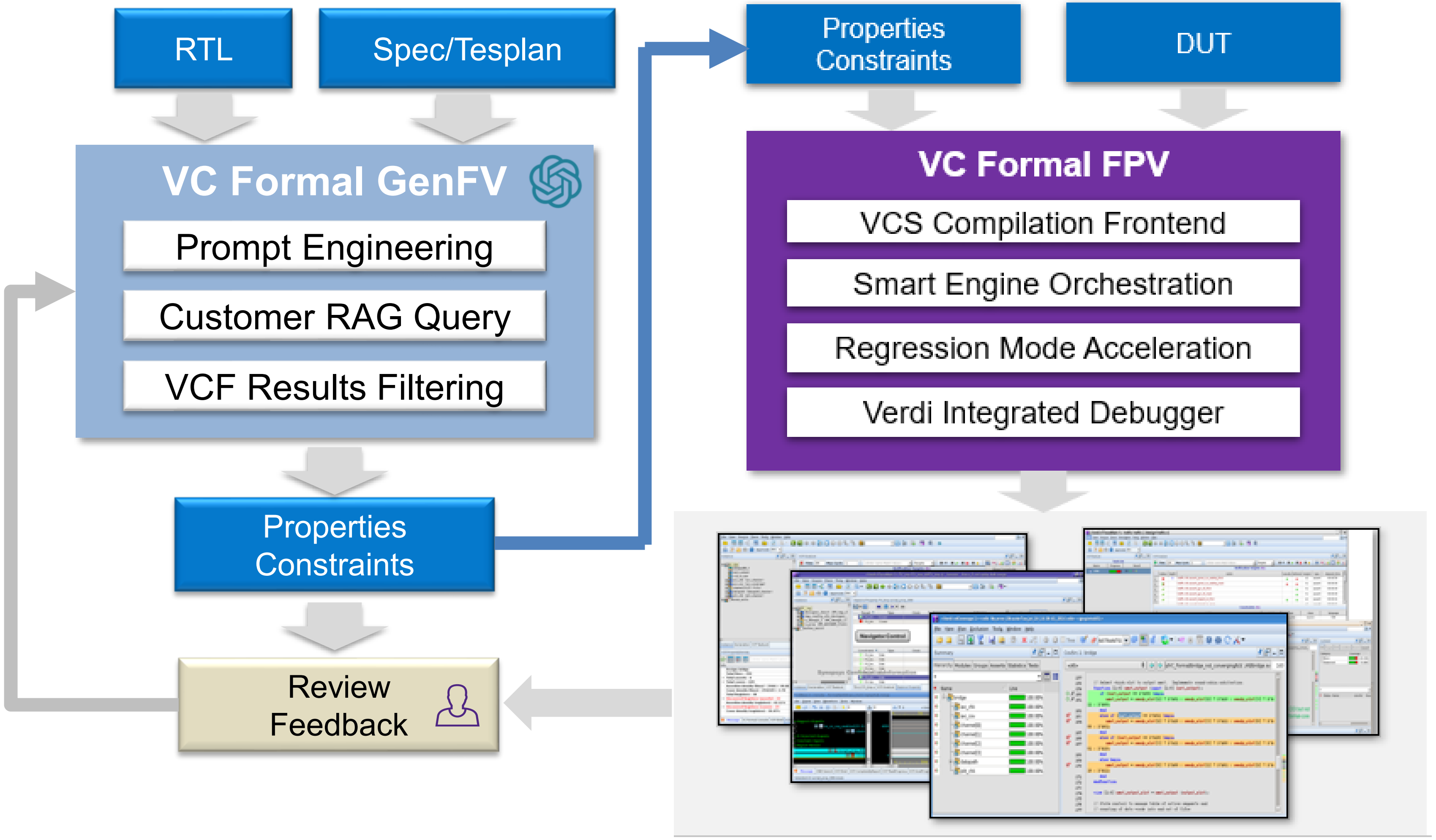
Presenter: Raja Mahadevan
Principal Engineer, Applications Engineering
Synopsys India Pvt Ltd

Formal Advisor (aka GenFV)

- Applying Generative AI in Formal Verification

WHY	• Today Most Formal Usage is Limited to Experts in Large Companies • Growing Demand for Formal Adoption in the Industry • VC Formal GenFV Holds the Promise to Democratize Formal Adoption
WHAT	• VC Formal GenFV Leverages LLM to Generate SVA Properties • Help Non-Experts to Learn and Adopt Formal Verification • Improve Productivity for Expert Formal Users
HOW	• Leverage the Capabilities of LLMs • Natural Language Interface Integrated with VC Formal and Verdi • Prompt Engineering and Results Filtering Improve QoR
WHEN	• Currently Engaged with Various Customers with Very Positive Feedback

Formal Advisor with FPV
Seamless Integration At Work



RTL

Spec/Tesplan

VC Formal GenFV

Prompt Engineering

Customer RAG Query

VCF Results Filtering

Properties Constraints

Review Feedback

Properties Constraints

DUT

VC Formal FPV

VCS Compilation Frontend

Smart Engine Orchestration

Regression Mode Acceleration

Verdi Integrated Debugger

Leveraging GenAI for Formal Testbench Generation

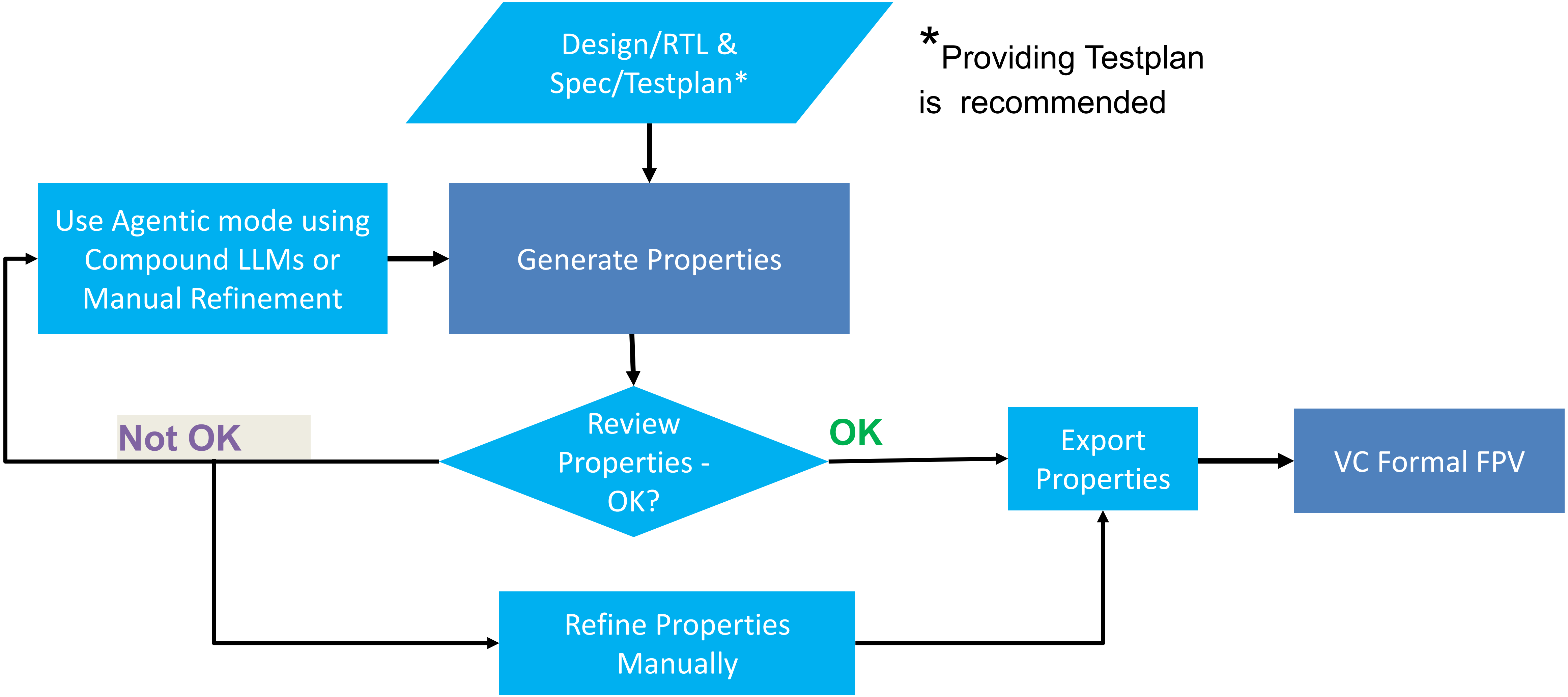
- Improve formal testbench development productivity

Usage Model

- User provides RTL files & selects text from test plan
- LLM Generates multiple properties
- Agentic mode with Compound LLMs for review / edit / refine / accept suggestions

  - Optionally save to RAG
- Generate Formal bench, Bind files and TCL scripts
- Run formal verification (FPV) seamlessly

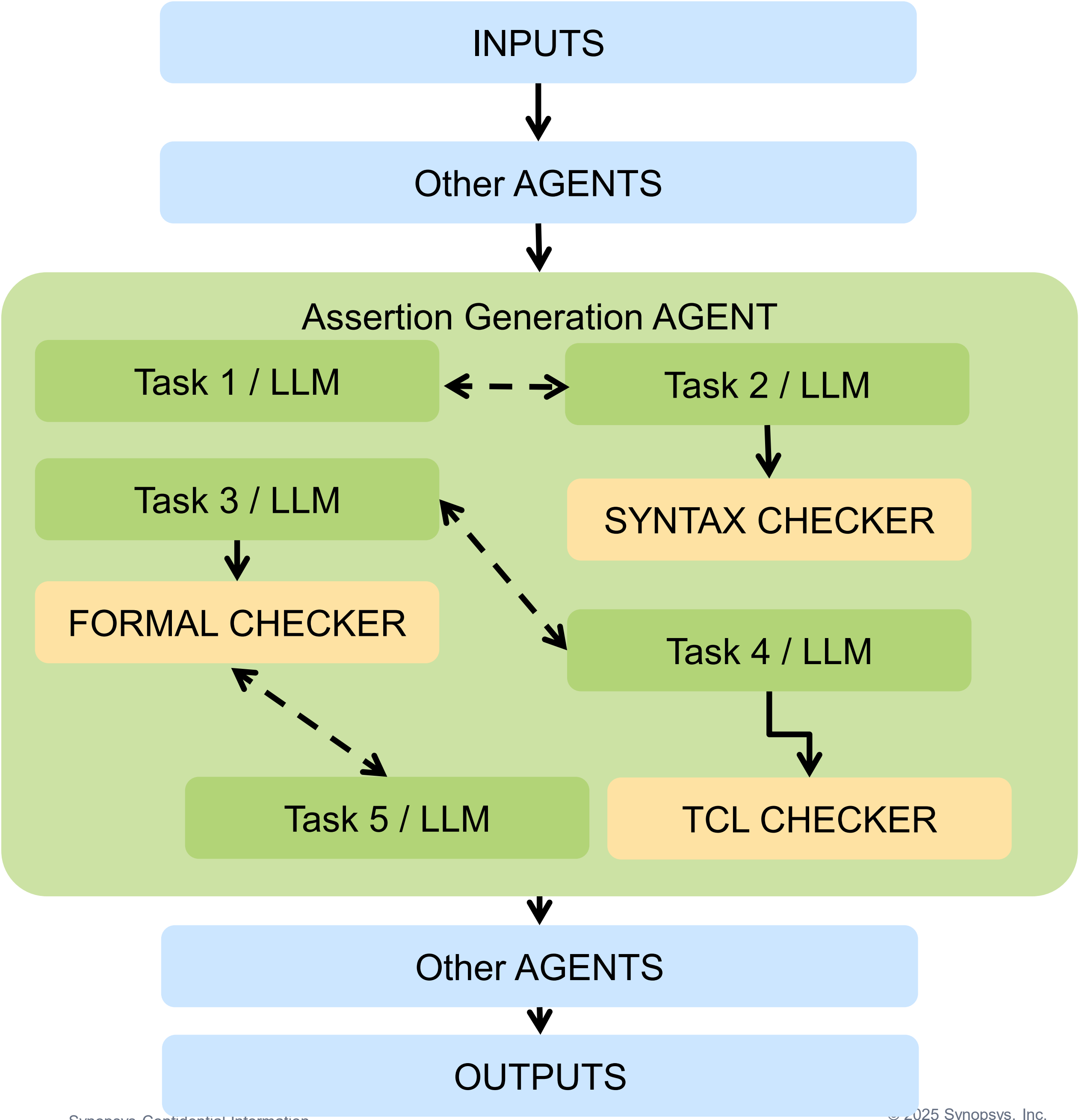
Formal Advisor - A Bird’s Eye view



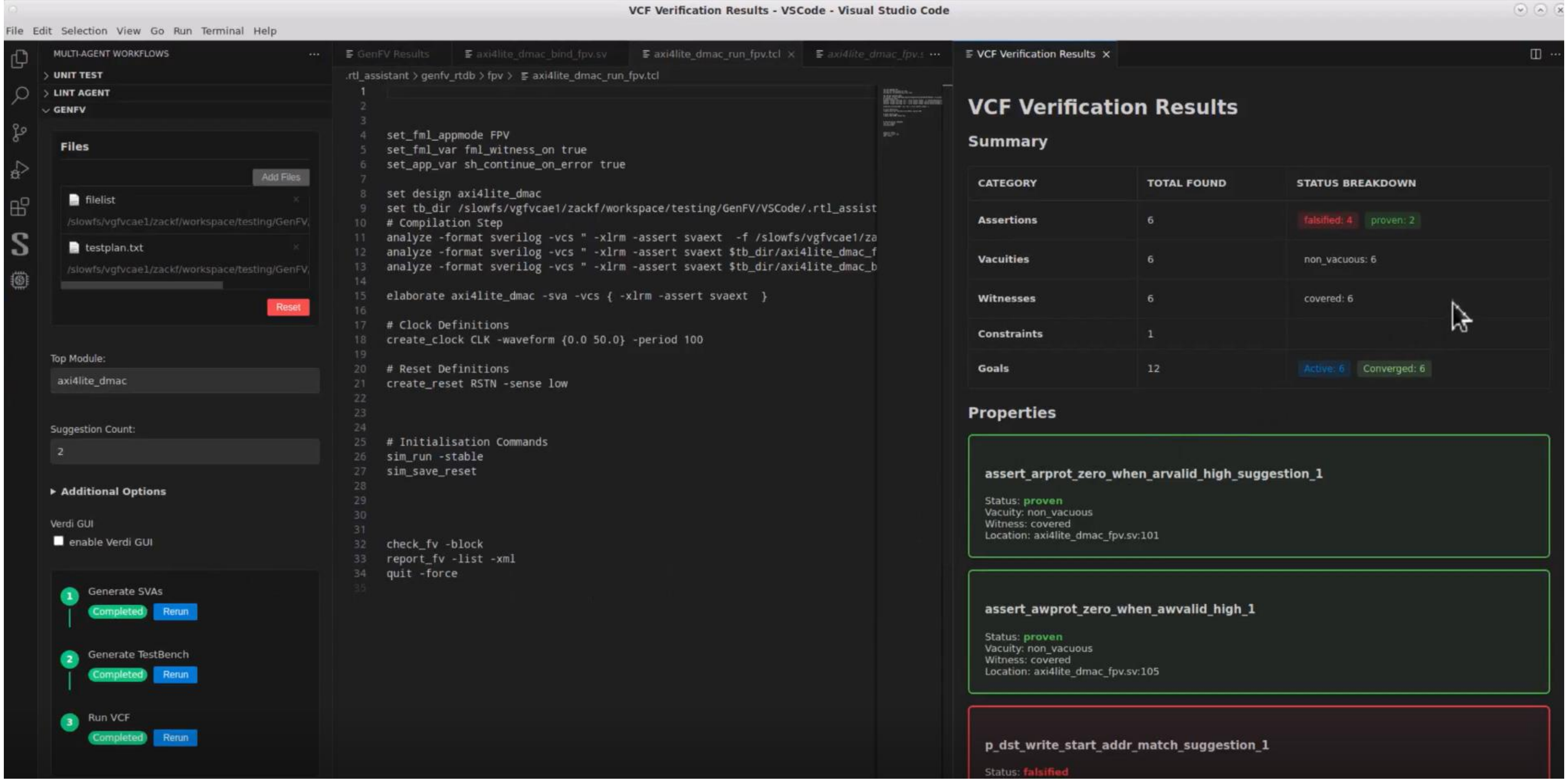
*Providing Testplan is recommended

Agentic Mode

- Agentic Flow Gives High Quality Results
- Multiple LLMs interacting together
- Collaborating with Core VC Formal components
- Fixing issues
  - RTL, Formal, TCL, etc.
- Reporting best results
- Plug & play with other agents



VC Formal’s Formal Advisor in VS Code



Files

filelist
/slowfs/vgfvcae1/zackf/workspace/testing/GenFV,

testplan.txt
/slowfs/vgfvcae1/zackf/workspace/testing/GenFV,

Top Module:
axi4lite_dmac

Suggestion Count:
2

Verdi GUI
□ enable Verdi GUI

1 Generate SVAs — Completed

2 Generate TestBench — Completed

3 Run VCF — Completed

```
set_fml_appmode FPV
set_fml_var fml_witness_on true
set_app_var sh_continue_on_error true

set design axi4lite_dmac
set tb_dir /slowfs/vgfvcae1/zackf/workspace/testing/GenFV/VSCode/.rtl_assist
# Compilation Step
analyze -format sverilog -vcs " -xlrm -assert svaext  -f /slowfs/vgfvcae1/za
analyze -format sverilog -vcs " -xlrm -assert svaext $tb_dir/axi4lite_dmac_f
analyze -format sverilog -vcs " -xlrm -assert svaext $tb_dir/axi4lite_dmac_b

elaborate axi4lite_dmac -sva -vcs { -xlrm -assert svaext  }

# Clock Definitions
create_clock CLK -waveform {0.0 50.0} -period 100

# Reset Definitions
create_reset RSTN -sense low



# Initialisation Commands
sim_run -stable
sim_save_reset




check_fv -block
report_fv -list -xml
quit -force
```

VCF Verification Results

Summary

CATEGORY	TOTAL FOUND	STATUS BREAKDOWN
Assertions	6	falsified: 4 proven: 2
Vacuities	6	non_vacuous: 6
Witnesses	6	covered: 6
Constraints	1	
Goals	12	Active: 6 Converged: 6

Properties

assert_arprot_zero_when_arvalid_high_suggestion_1

Status: proven
Vacuity: non_vacuous
Witness: covered
Location: axi4lite_dmac_fpv.sv:101

assert_awprot_zero_when_awvalid_high_1

Status: proven
Vacuity: non_vacuous
Witness: covered
Location: axi4lite_dmac_fpv.sv:105

p_dst_write_start_addr_match_suggestion_1

Status: falsified

Formal Advisor Success Stories

Feedback on Productivity Gain

RESULTS

“Saved 9.25 hours (35%) developing formal testbench using GenFV”

Leading CPU Provider → **35%** Productivity Gain

“A complex SVA assertion that took half a day to create was generated by GenFV in perfect form in minutes”

Leading Data Infra Company → **50+X** Productivity Gain

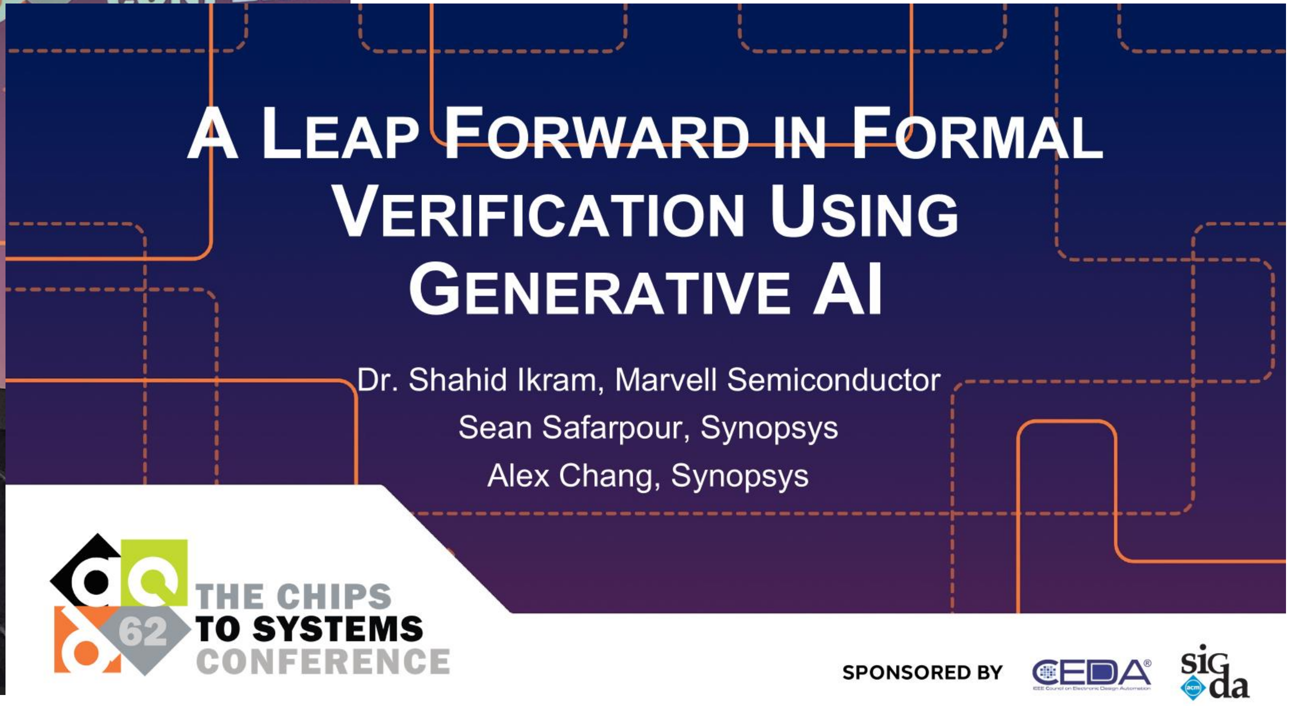
“Incrementally adding accepted properties to RAG works very well to improving quality of generated assertions”

Synopsys IP Group → **40%** Productivity Gain

** Results are in comparison to formal testbench development without GenFV*

Success Story: Best Presentation Award at DAC US 2025

Best Presentation - Front-End Design
A Leap Forward in Formal Verification Using Generative AI
Shahid Ikram (Marvel Semiconductor), Sean Safarpour (Synopsys), and Alex Chang (Synopsys)
THE CHIPS TO SYSTEMS CONFERENCE



A Leap Forward in Formal Verification Using Generative AI

Dr. Shahid Ikram, Marvell Semiconductor
Sean Safarpour, Synopsys
Alex Chang, Synopsys

THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA sigda

Microsoft Success: Productivity Gain with GenFV

Productivity gain throughout the whole formal verification flow

Source: SNUG, Silicon Valley, 2025

Proof Bring-Up

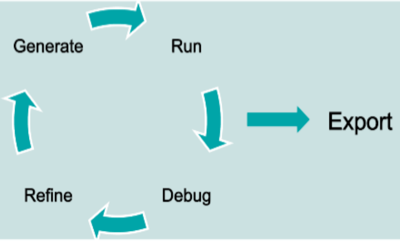
75% faster

New Users

15 mins of training to get to first property

Integrated property generation loop

25% time saving



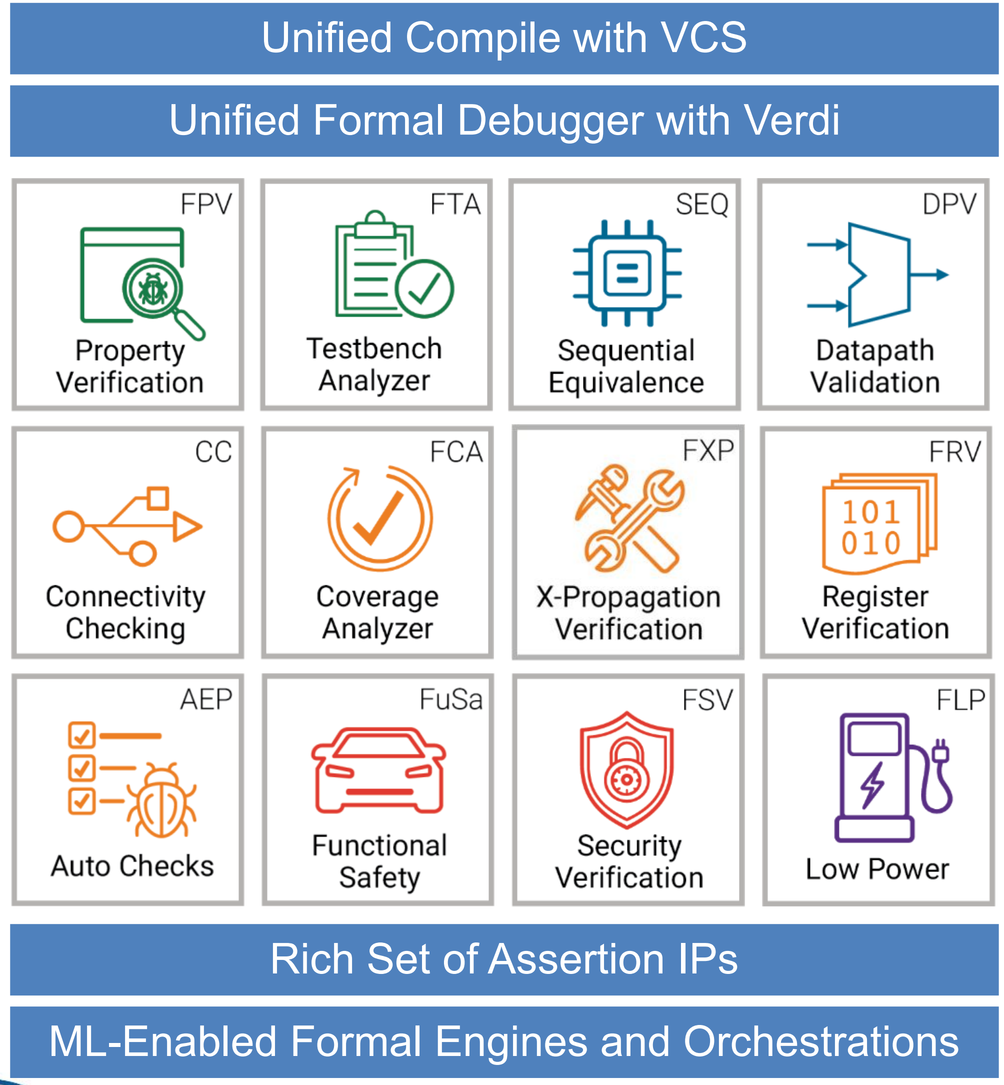
Formal Verification for RISC-V Designs

RISC-V Processor Verification Methodology

Design verification from unit to SoC

Design Level	Example	Tool/Methodology
Unit	Pipeline, FPU	Formal + predefined assertion IP
	Security	Formal + predefined security assertion IP
Architecture	ISA	Dynamic
		Formal: predefined assertion IP
Custom instructions, CSRs	Custom DSP, matrix	Dynamic
		Formal sequential equivalence checking, register verification, Datapath validation
Processing subsystem	Coherent cache, multi- or many-processor accelerator	Dynamic, especially using hardware assisted verification
		Formal property verification for cache coherence verification

Synopsys VC Formal – Leading Formal Innovations



Unified Compile with VCS

Unified Formal Debugger with Verdi

Rich Set of Assertion IPs

ML-Enabled Formal Engines and Orchestrations

Industry's Fastest Growing Formal Solution!

Deliver highest performance

Innovative formal engines and ML-based orchestrations find more bugs and achieve more proofs on larger designs

Enable formal signoff

Exhaustive formal analysis catches corner-case bugs and enables formal signoff for control and datapath blocks

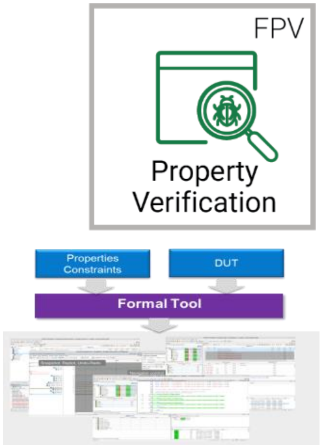
Ease Formal adoption

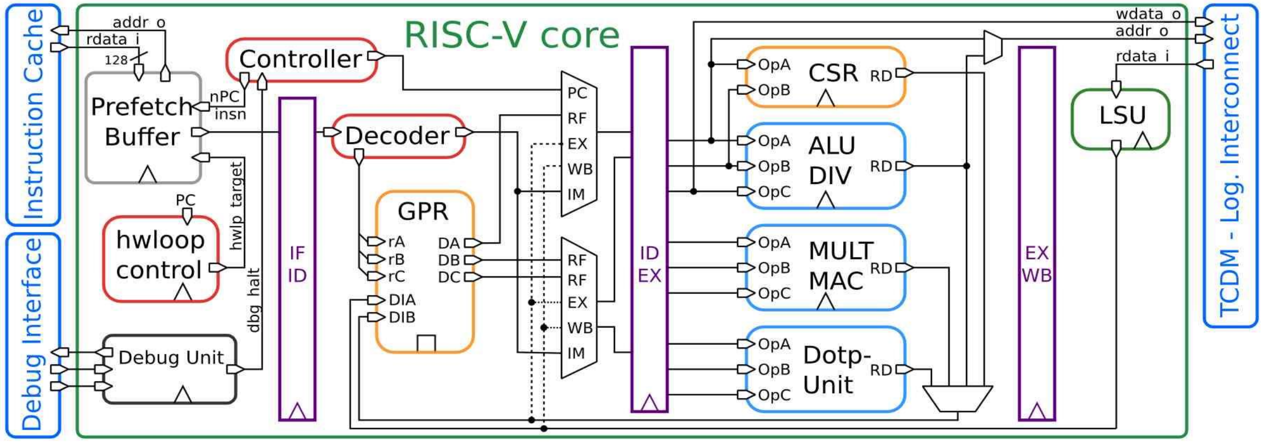
Easy-to-use formal apps, native integration with VCS and Verdi, and Formal Consulting Services reduce formal adoption effort

RISC-V Core Unit Verification Task Examples Using Formal Property Verification

- Prefetch Buffer:
  - Redirect/Clear from various components: BPU/EX etc. should cause proper action and in a priority order
  - Instruction Cache
  - Direction/Target Prediction
  - Branch Target Buffer
  - Wake: Detecting a ready instruction
  - Dispatch: Need to select (oldest woken-up instruction first)
  - Resolve Dependencies
- Decoder
  - Check for undefined instructions
  - Fusing check if 2 or 3 instructions can be used together
- Execution (ALU):
  - Simple ALU functions
  - Bypass Functionality Checking
  - Misprediction should lead to redirect; Correct prediction should result in completion

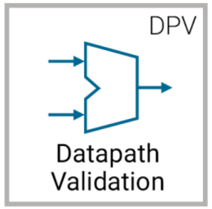
- Load/Store Unit (LSU):
  - Load addr should be sent before Load data
  - Store addr should be sent before store data
  - Load/Store Functionality
- Pipeline
  - Control logic



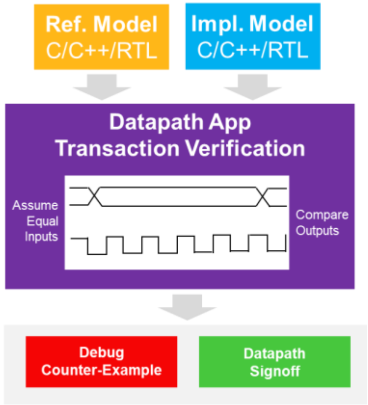


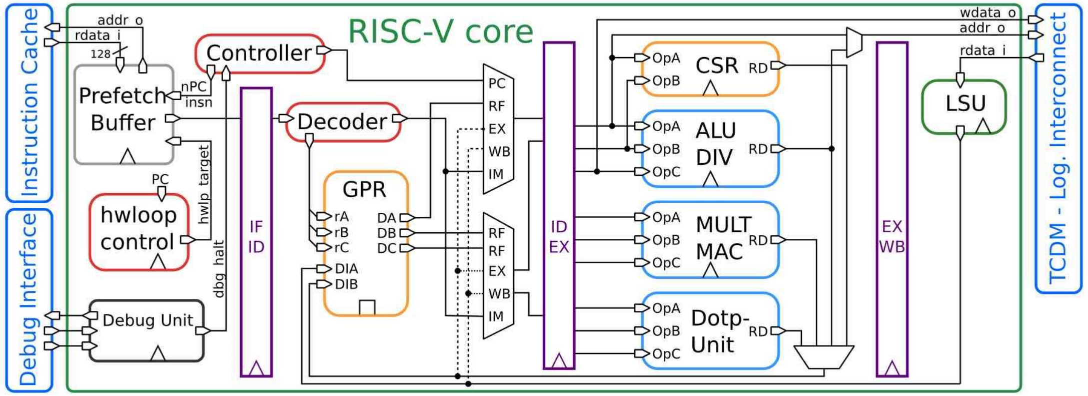
Source: https://www.semanticscholar.org/paper/Near-Threshold-RISC-V-Core-With-DSP-Extensions-for-GautschiSchiavone/47f8ce7e0f0f64d0707a13c83c32c30959aa64d5/figure/6

RISC-V Core Unit Verification Task Examples Using Formal Datapath Validation



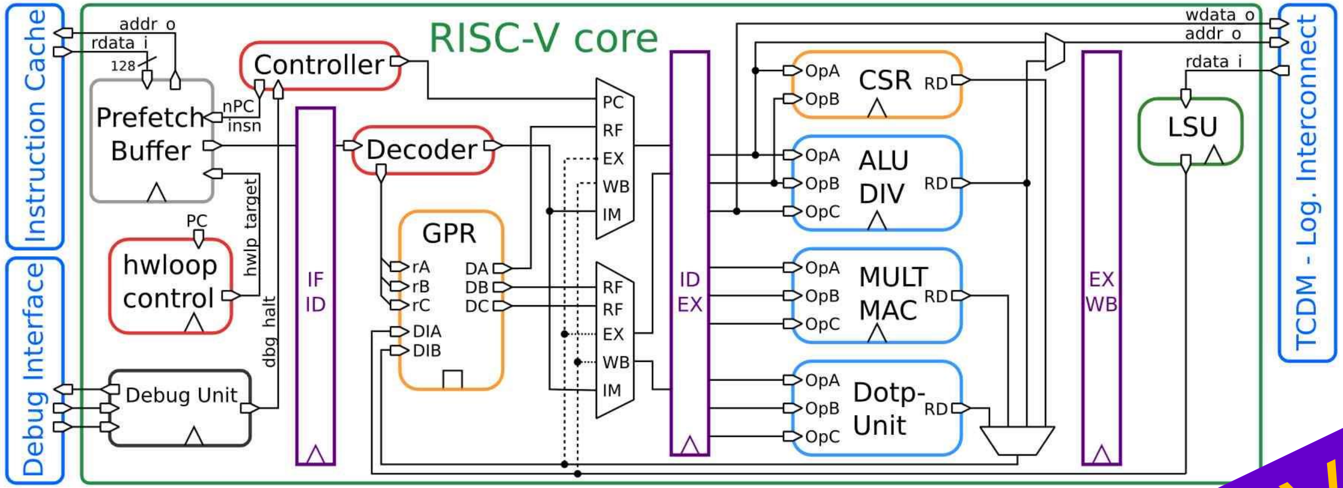
- Verify RISC-V CPU cores using pre-packaged models of RISC-V arithmetic instructions
- Generate reference model depending on the instruction and use it to verify execution result of data path related instructions
  - SoftFloat Library – RISC-V support
  - Math Library or Openlib
- Applicable to M (Multiplier/Divider), F(Single Precision Floating Point) and D (Double precision Floating Point) standard extensions
- Execution (ALU/MULT/Dotp):
  - Complex ALU functions





Source: https://www.semanticscholar.org/paper/Near-Threshold-RISC-V-Core-With-DSP-Extensions-for-GautschiSchiavone/47f8ce7e0f0f64d0707a13c83c32c30959aa64d5/figure/6

RISC-V Core – Formal Verification Overview



Source: https://www.semanticscholar.org/paper/Near-Threshold-RISC-V-Core-With-DSP-Extensions-for-GautschiSchiavone/47f8ce7e0f0f64d0707a13c83c32c30959aa64d5/figure/6

- FRV (Formal Register Verification)
  - Control and Status Registers (Zicsr)
    - CSR Write
    - CSR Read

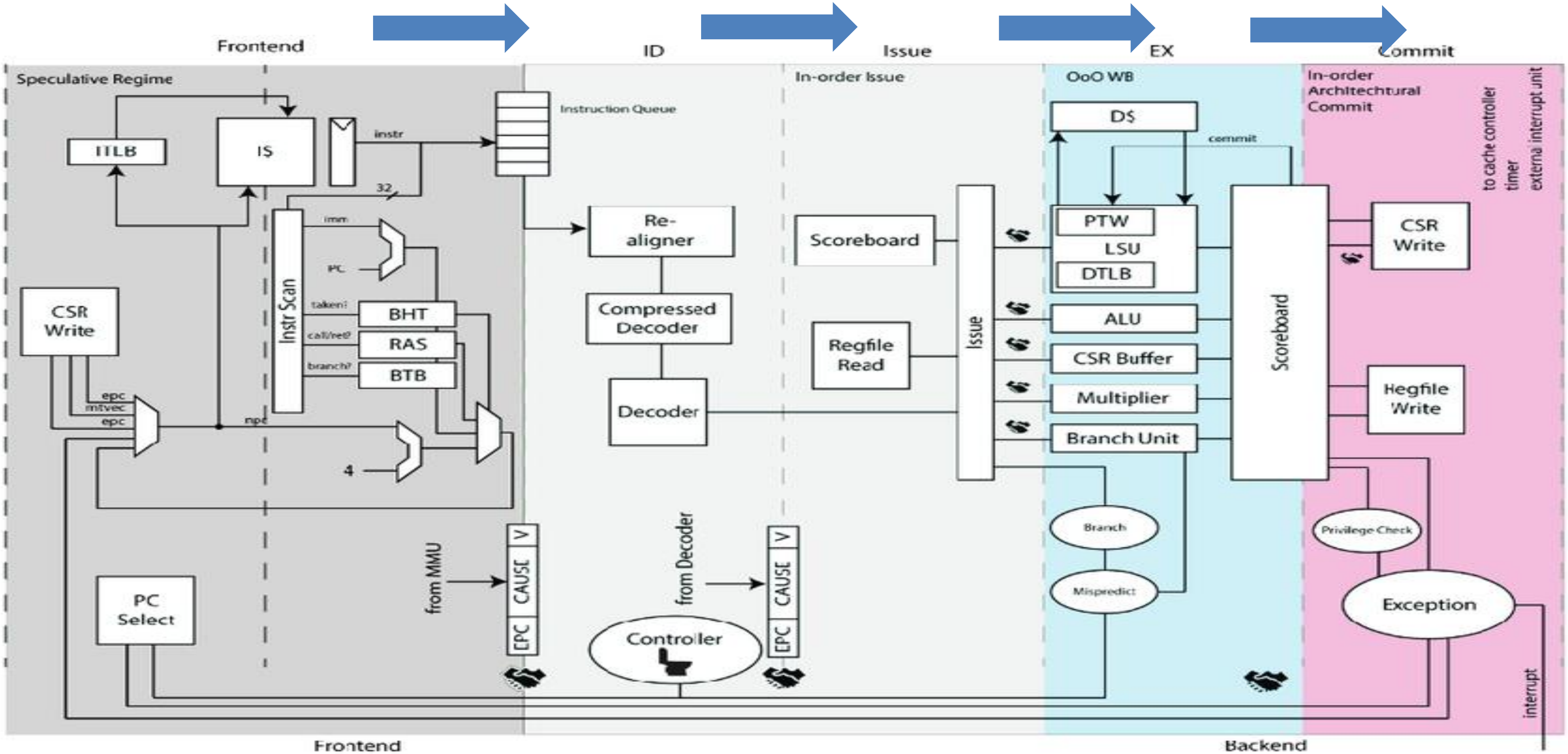
- SEQ (Equivalence Checking):
  - Clock gating verification in every functional unit
  - Designs comparison in presence of new features/timing changes

- FSV (Formal Security Verification)
  - Secure/Non-secure data propagation

But What About ISA Verification?

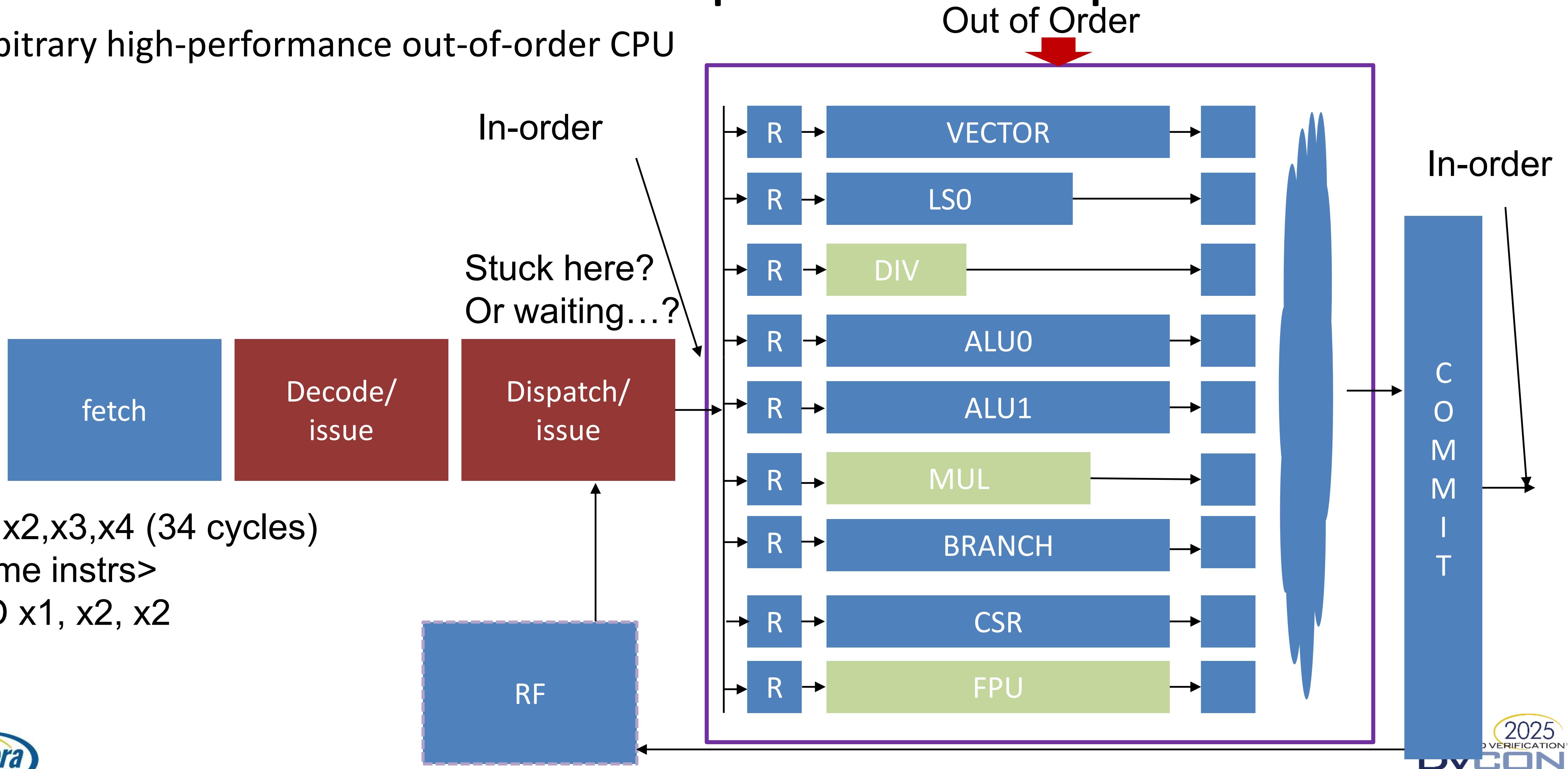
Single Issue Complex Core Pipeline Example

OpenHW CVA6



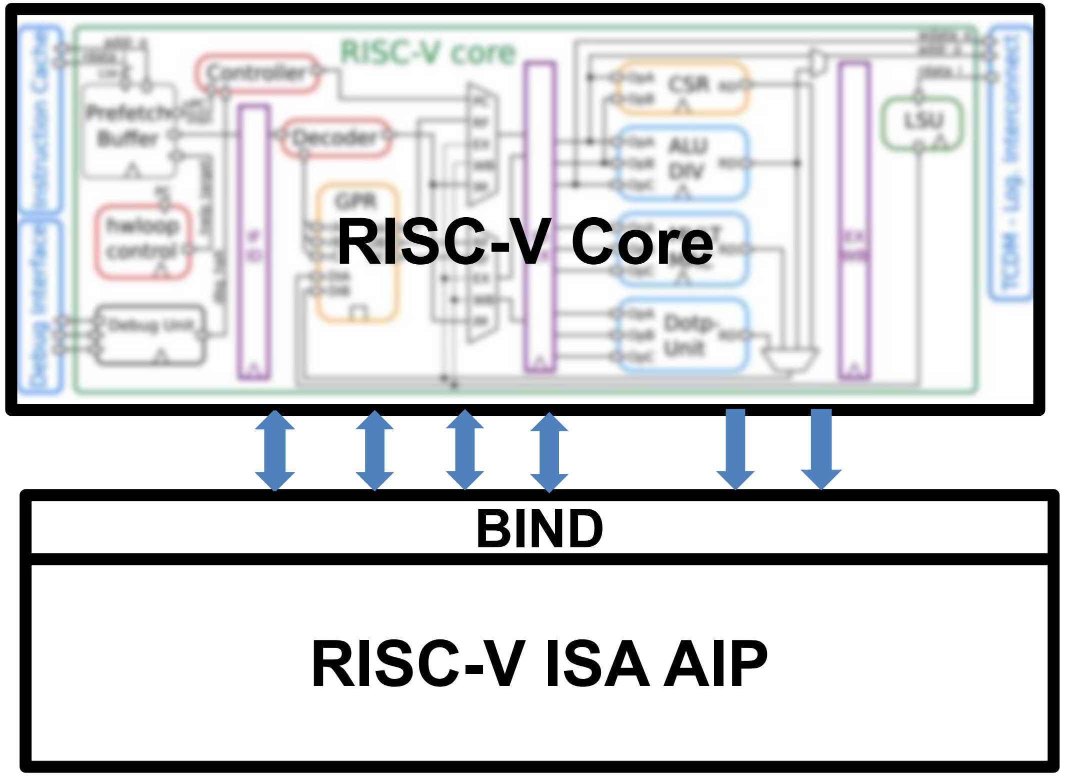
Out of Order Pipelines Example

Arbitrary high-performance out-of-order CPU



DIV x2,x3,x4 (34 cycles)
<some instrs>
ADD x1, x2, x2

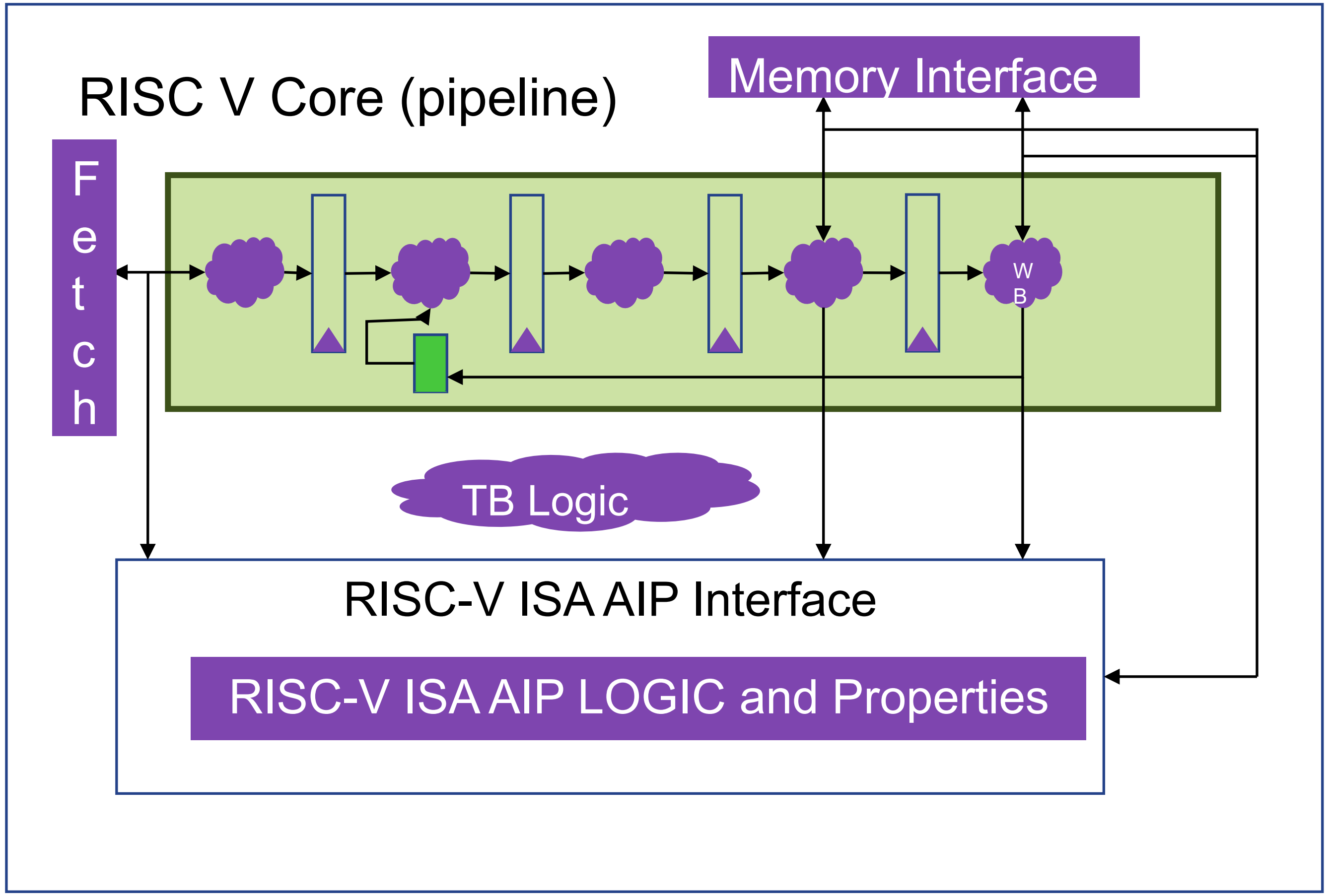
VC Formal RISC-V AIP for Exhaustive ISA Verification



Benefits of RISC-V ISA AIP Formal Verification

- Formal exhaustively tests all possible RISC-V instruction scenarios
- Availability of RISC-V ISA AIP reduces debug turn-around-time
- RISC-V ISA AIP validates instruction execution control and base-ISA data path
  - For complex math operations (MUL/DIV), will need DPV verification to ensure datapath correctness
- RISC-V ISA AIP can be used for multiple configurations and cores
- Verification quality and confidence are high

Formal RISC-V ISA AIP Applied to Design



- RISC-V ISA AIP needs minimal access to a small number of points around the pipeline to observe certain events
- Interfaces to bind to RISC-V ISA AIP
  - Instruction Fetch Interface
  - Data Memory Interface
  - Register Write Interface
  - Instruction Retire Interface
- Testbench logic
  - DUT signal expressions to bind to RISC-ISA AIP interfaces
  - DUT-specific constraints
- RISC-V ISA AIP offers all the properties and policies for checking the instruction architecture

RISC-V ISA AIP Property Example

Checker for Load address

Check conditions

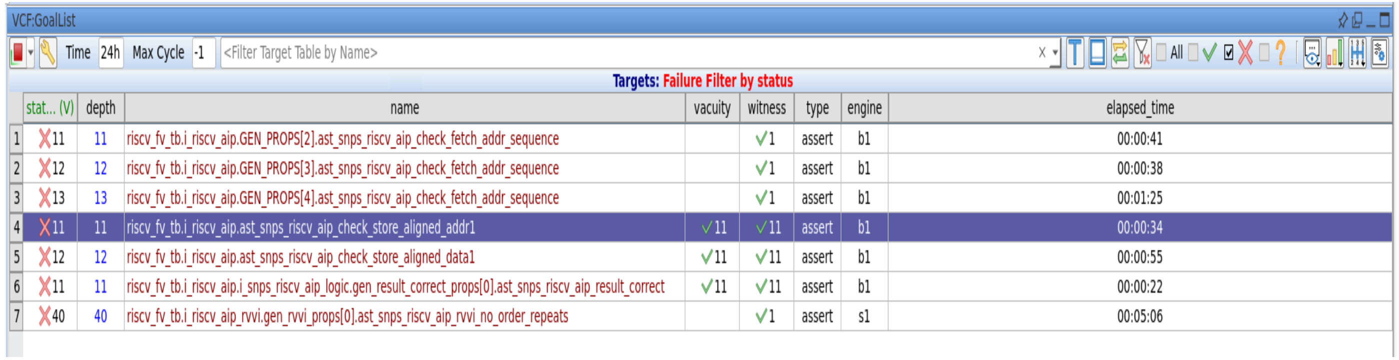
```
property p_snps_riscv_aip_check_load_addr1;
   (instructions_q[instr_mem_load_addr_count_q].instr_load  &&
    instructions_q[instr_mem_load_addr_count_q].instr_state== INSTR_LOAD_ADDR1 &&
    read_addr_valid
    |-> ((read_addr == instructions_q[instr_mem_load_addr_count_q].mem_addr1) || // aligned case
         (read_addr == instructions_q[instr_mem_load_addr_count_q].effective_addr))); // unaligned case
endproperty
```

Memory read address signal in DUT

Expected memory read address calculated in AIP

RISC-V ISA AIP Functionality – In a Nutshell

- RISC-V ISA AIP keeps track of all control flow instructions
- RISC-V ISA AIP keeps track of all hazards between instructions
- RISC-V ISA AIP supports multi-issue/out of order execution
- RISC-V ISA AIP keeps track of the most up to date value of all registers

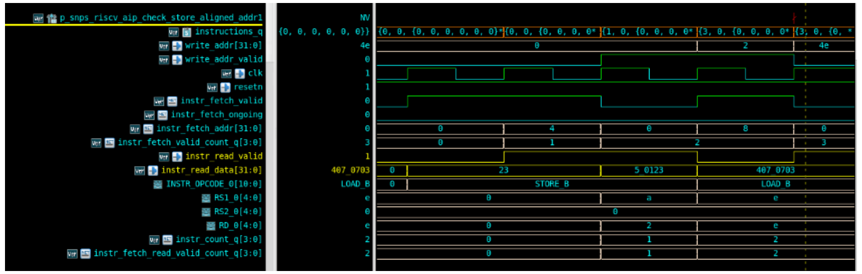


VCF:GoalList

Time 24h Max Cycle -1 <Filter Target Table by Name>

Targets: Failure Filter by status

	stat... (V)	depth	name	vacuity	witness	type	engine	elapsed_time
1	✗11	11	riscv_fv_tb.i_riscv_aip.GEN_PROPS[2].ast_snps_riscv_aip_check_fetch_addr_sequence		✓1	assert	b1	00:00:41
2	✗12	12	riscv_fv_tb.i_riscv_aip.GEN_PROPS[3].ast_snps_riscv_aip_check_fetch_addr_sequence		✓1	assert	b1	00:00:38
3	✗13	13	riscv_fv_tb.i_riscv_aip.GEN_PROPS[4].ast_snps_riscv_aip_check_fetch_addr_sequence		✓1	assert	b1	00:01:25
4	✗11	11	riscv_fv_tb.i_riscv_aip.ast_snps_riscv_aip_check_store_aligned_addr1	✓11	✓11	assert	b1	00:00:34
5	✗12	12	riscv_fv_tb.i_riscv_aip.ast_snps_riscv_aip_check_store_aligned_data1	✓11	✓11	assert	b1	00:00:55
6	✗11	11	riscv_fv_tb.i_riscv_aip.i_snps_riscv_aip_logic.gen_result_correct_props[0].ast_snps_riscv_aip_result_correct	✓11	✓11	assert	b1	00:00:22
7	✗40	40	riscv_fv_tb.i_riscv_aip_rvvi.gen_rvvi_props[0].ast_snps_riscv_aip_rvvi_no_order_repeats		✓1	assert	s1	00:05:06



RISC-V Cores Examples Verified by RISC-V ISA AIP

- Example RISC-V cores supported by RISC-V AIP
  - OpenHW CV32E40P and CV32E40X
  - OpenHW CVA6 (as both 32bit and 64bit variants)
  - IBEX
  - WD Swerv EH2
  - Synopsys RISC-V Cores

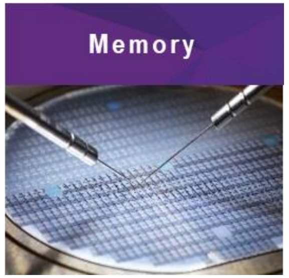
…

Examples of Bugs Found With RISC-V Formal AIP

Bug description	FV runtime	Likely to find in simulation
Simultaneous writes to same destination register from stalled LOAD_FP retiring out of order with subsequent OP_FP	~20 min	Low
RV32F LOAD_FP unexpectedly writing 64-bit floating point values to FP register file when core is configured as 64bit integer pipeline (RV64I) with RV32F – core overrides RV32F and instantiates 64bit FP pipeline (config bug)	~20 min	High
A power optimization problem where inadvertent multiple register writes were seen for stalled or unaligned load	~2 min	Med
Core fully executes instruction that was not requested and updates the integer register file – instr_read_valid without first instr_fetch_valid. Although protocol is violated, core does not protect the pipeline (expose security hole)	~1 min	Med

Synopsys Addresses Needs of RISC-V Design Community

- Synopsys is a strategic member of RISC-V International
- Partner with key RISC-V core providers, foundries and universities
- Interoperability of Synopsys IP with RISC-V solutions
- Customized and adaptive flows for implementation and verification
- Collateral, user guides, training, cloud-based solution and design services are Available



Q & A

Thank You