2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

A Hybrid Verification Solution to RISC-V Vector Extension

Chenghuan Li
Mediatek.inc, Beijing, China
Chenghuan.Li@mediatek.com

Yanhua Feng
Mediatek.inc, Beijing, China
Yanhua.Feng@mediatek.com

Liam Li
Mediatek.inc, Beijing, China
Liam.Li@mediatek.com

accellera SYSTEMS INITIATIVE

MEDIATEK

Agenda

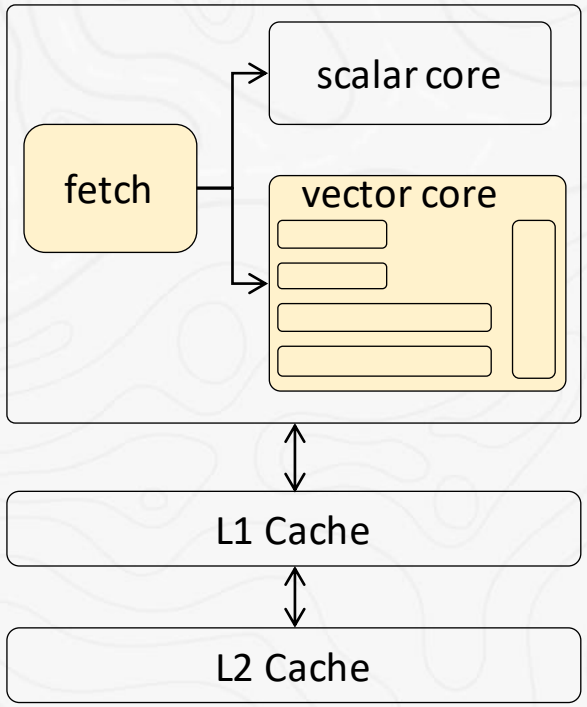
- Design/Testbench overview

- Instruction model auto-generation flow
- Exception verification
- Formal's application
  - Early-stage design exploration
  - Hazard verification

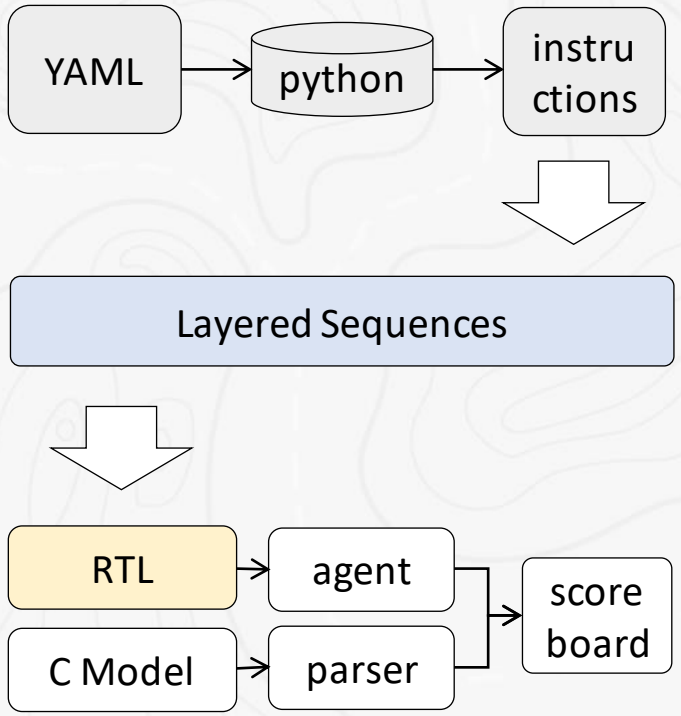
- Summary

Design/Testbench Overview

Design: 12-stage pipeline, superscalar core and configurable L1/L2 cache

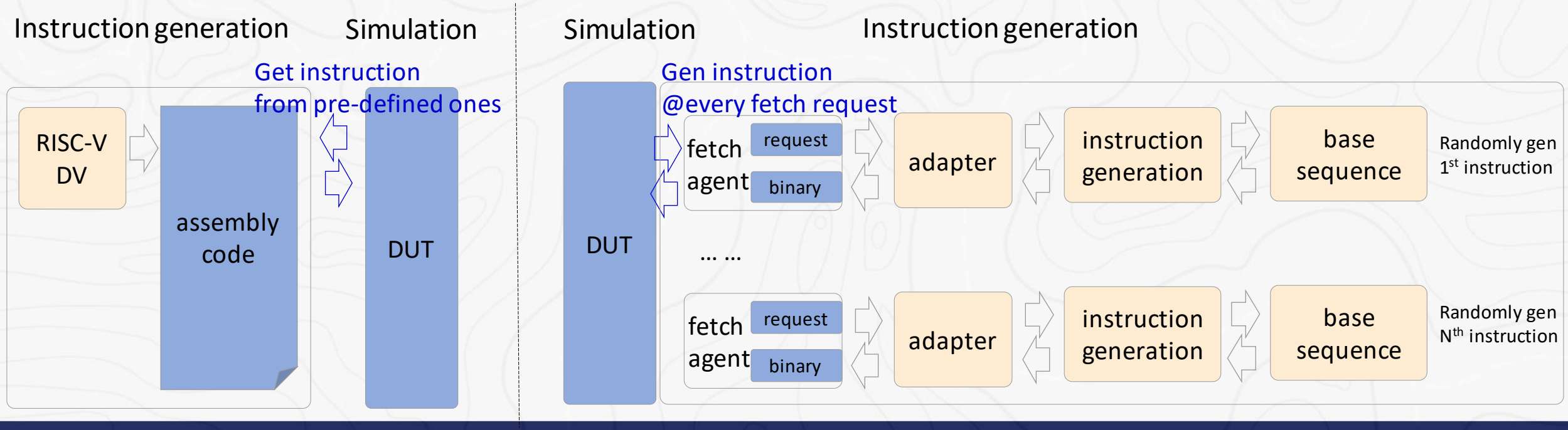


UVM-based simulation verification env.



Instruction Model Auto-Generation Flow

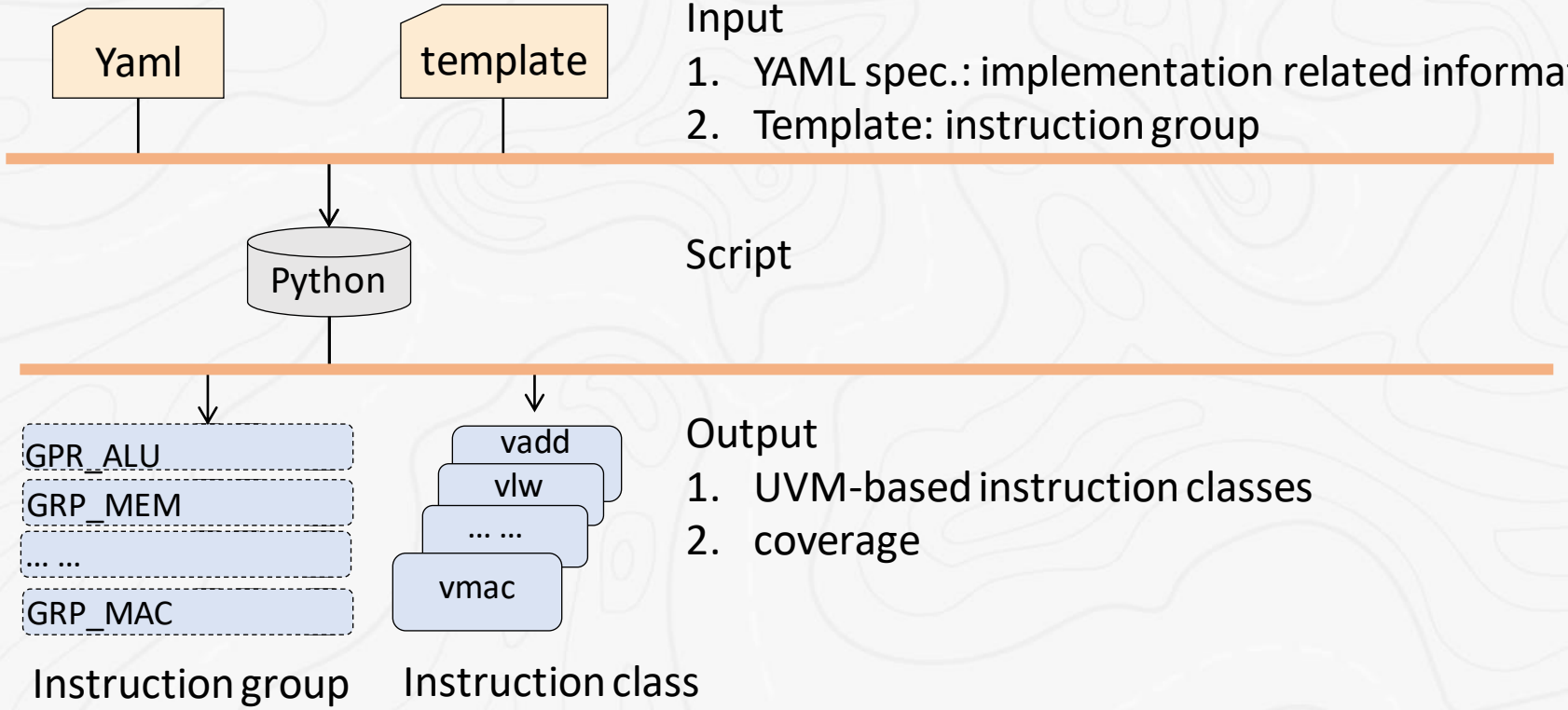
- Issue statement:
  1. Frequent ISA changes from both RISC-V evolution and MTK's customization.
  2. DV's Need for RTL's implementation information in both stimulus and checkers.
- Online vs. Offline instruction generator



Instruction Model Auto-Generation Flow (cont.)

- Auto-generate instruction classes per design spec.

```
vadd.vv
  MTKext: rvv
  hw_resource:xx
  name:vadd.vv
  issue: xx
  exception: xx
  opcode:
    image: '000000'
    type : string
    data:
      name : vs2
      opcode: xx
      pipe: xx
    … …
```



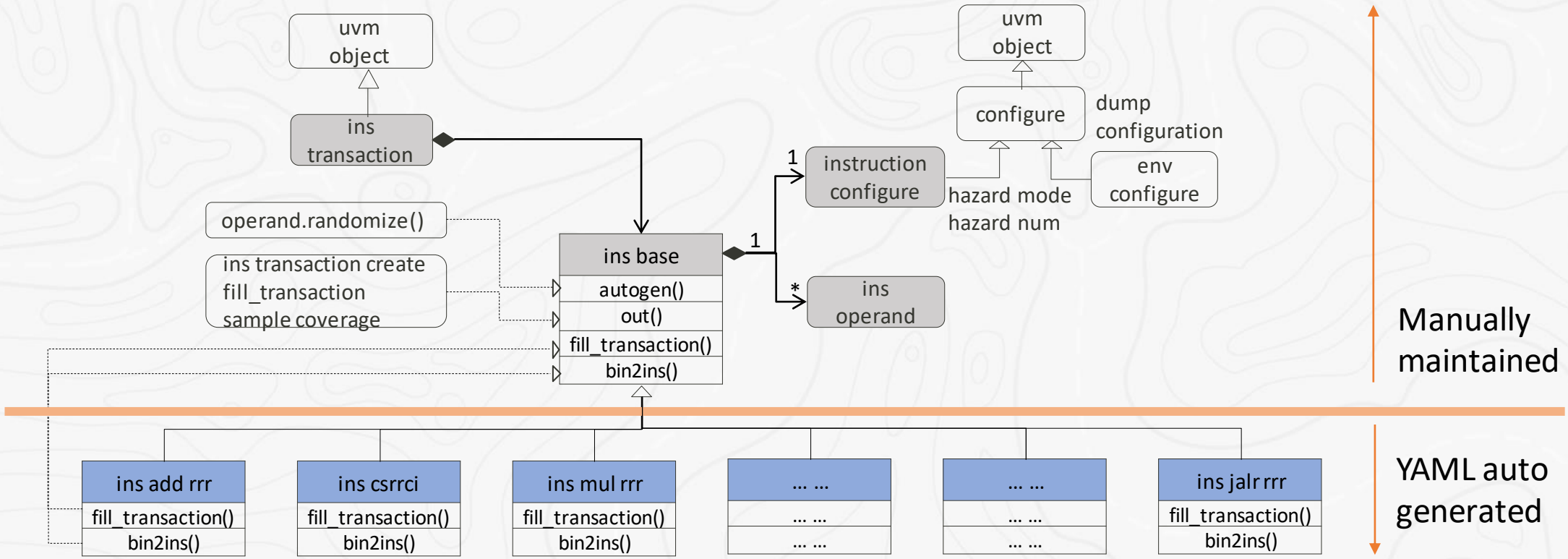
Input
1. YAML spec.: implementation related information
2. Template: instruction group

Script

Output
1. UVM-based instruction classes
2. coverage

Instruction Class

- Base class defines the instruction's prototype
- YAML auto-generated classes fulfill the specific implementation



Exception Verification

- RVV raises higher requirement for instruction's contextual relevance.

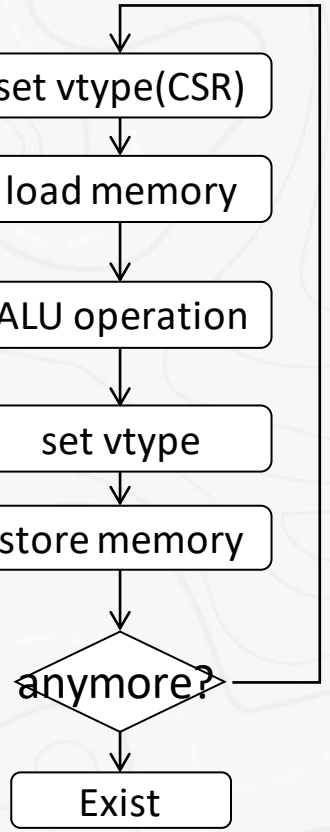
Exception category

Exceptions / Instructions		VTYPE			Operand index			Un-align	Rsvd	R/W
		LMUL	SEW	VILL	vd	vs2	vs1			
vsetvl(i)				√					√	
CSR				√						√
ALU	wdn	√	√	√	√	√	√			
	narr	√	√	√	√	√	√			
	norm			√	√	√	√			
Memory			√	√	√	√	√	√		
Special					√	√	√			
Customized		√	√	√	√	√	√	√	√	√

Snippet of RVV's typical usage

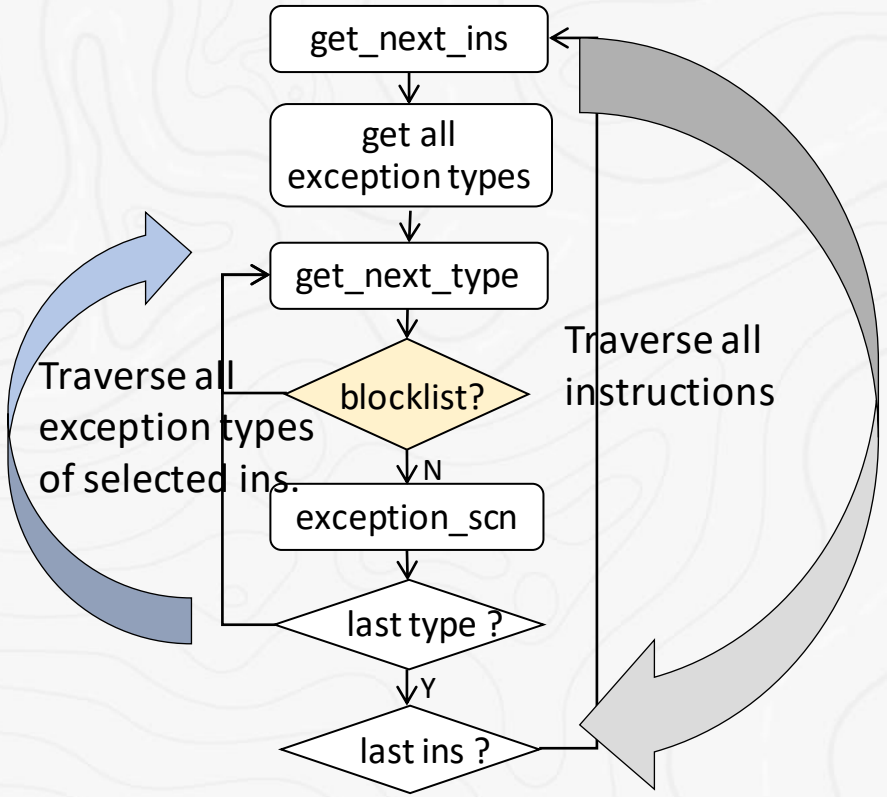
```
Loop:
    vsetvli a3, a0, E16 , M4
    vlh.v v4 (a1)
    slli t1, a3, 1
    add a1, a1, t1
    vwmul.vx v8, v4, x10

    vsetvli x0, a0, e32  , m8
    vsrl.vi v8, v8, 3
    vsw.v v8 , (a2)
    slli t1, a3, 2
    add a2, a2, t1
    sub a0, a0, a3
    bnez a0, loop
```

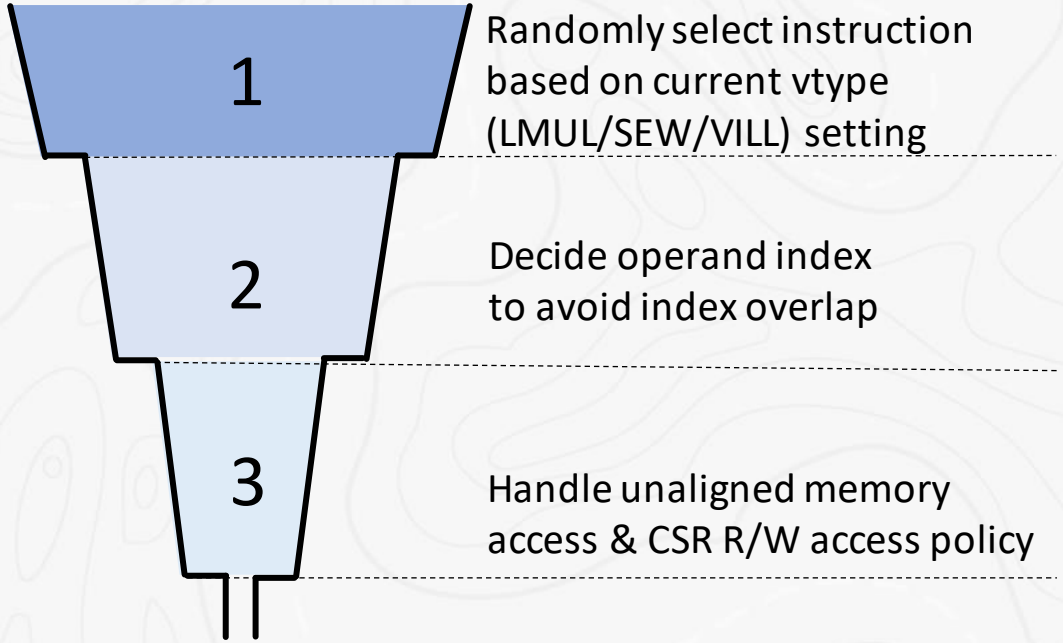


Exception Verification (cont.)

Exception verification: travers all possible exception cases

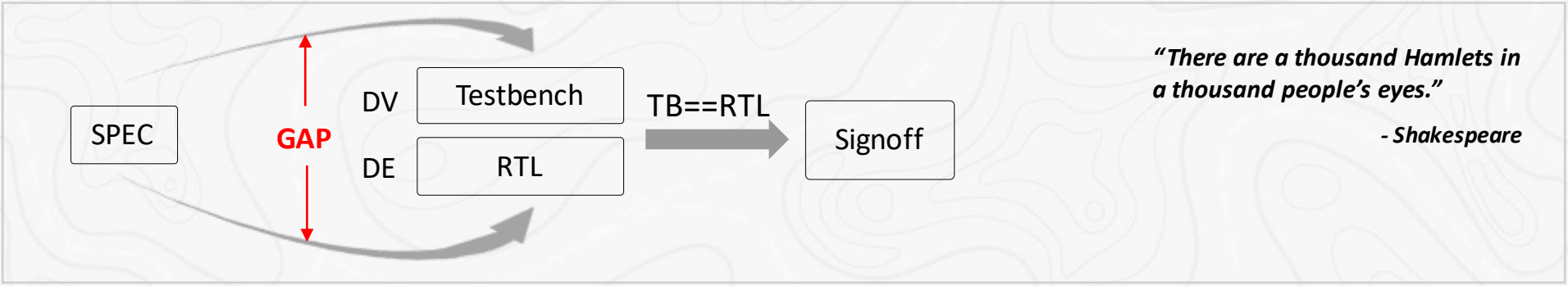


Normal cases: generate legal instruction as much as possible

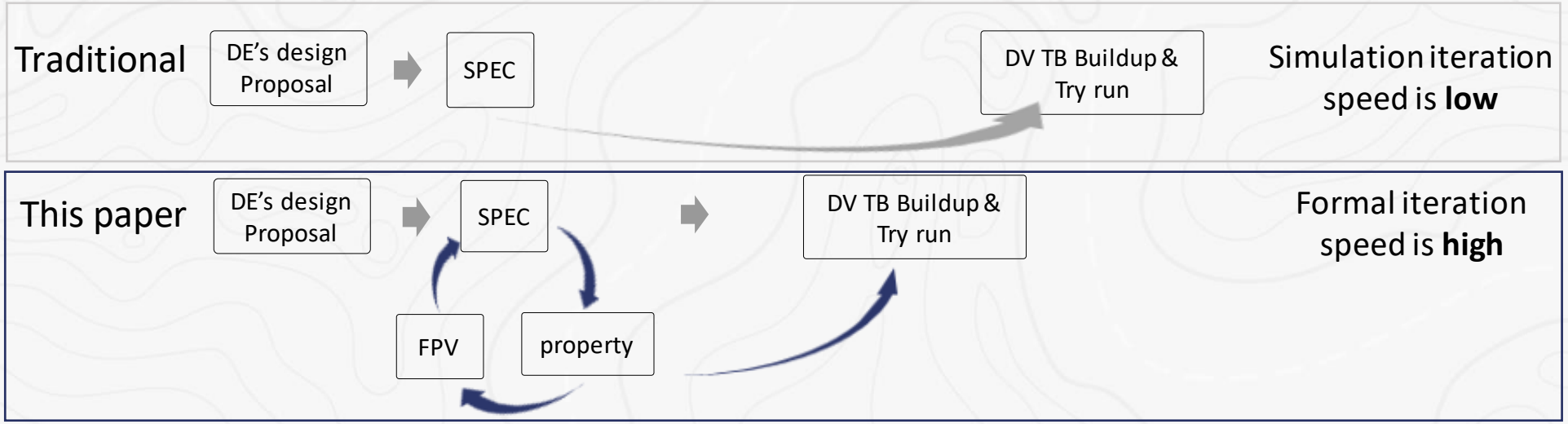


Early-Stage Design Exploration

- Issue statement

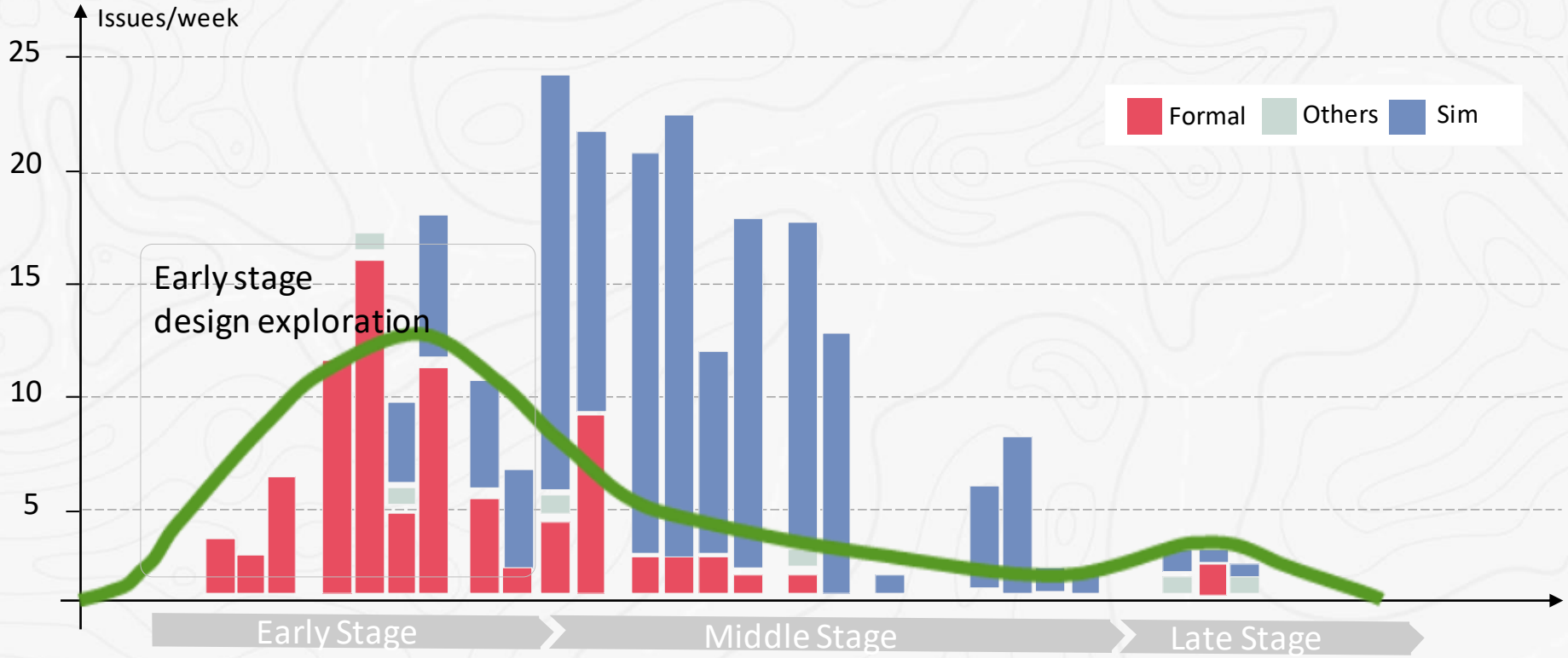


- Introduce formal property verification (FPV) in early stage



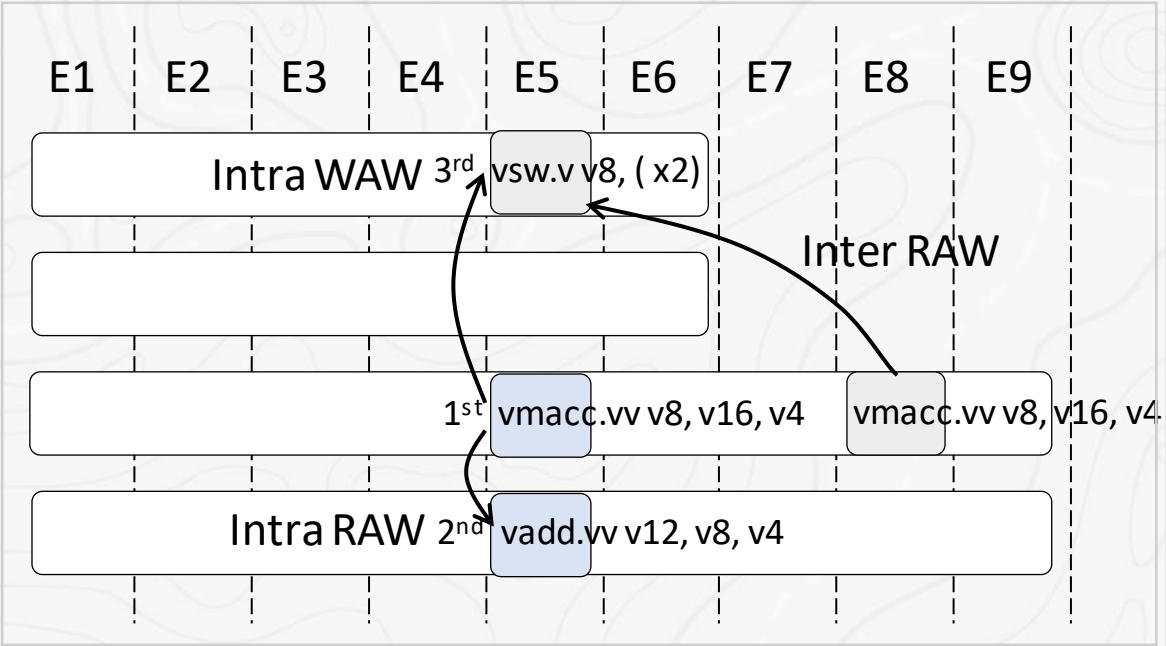
Results

- **~80%** early design issues detection with early-stage design exploration



Hazard Verification

- Issue statement
  1. Both inter and intra RAW/WAW/WAR data hazard needed handling in design
  2. Simulation failed to detect performance bugs w/o specific local checkers



Expect\Actual	Stall	Forward	Not handled
Stall	√	??	Func. bug
Forward	Perf. bug	√	Func. bug

formal

simulation

Hazard Verification (cont.)

Use **liveness** property to assert typical instruction's retirement

```
property ast_vmv_nfr_eventually_retire;
    logic [4:0] colored_idx;
    @ (posedge clk)) disable iff (! rstn)
    (insn_vld_e1 & insn_is_vmv_nfr_e1, colored_idx=insn_idx_e1) |->
    s_eventually (insn_wr_vrf & vrf_idx==colorder_idx);
endproperty
rvv_ast_vmv_nfr_eventually_retire: assert property (ast_vmv_nfr_eventually_retire);
```

Use **cover** property to explore longest stall cycles

```
property cov_stall_raised_20t (stall);
    @ (posedge clk) disable iff (! rstn)
    $rose (stall) |-> stall [*20];
endproperty
rvv_cov_stall_raised_20t: cover property (cov_stall_raised_20_t (x_stall));
```

Change liveness property to **safety** ones, re-prove it

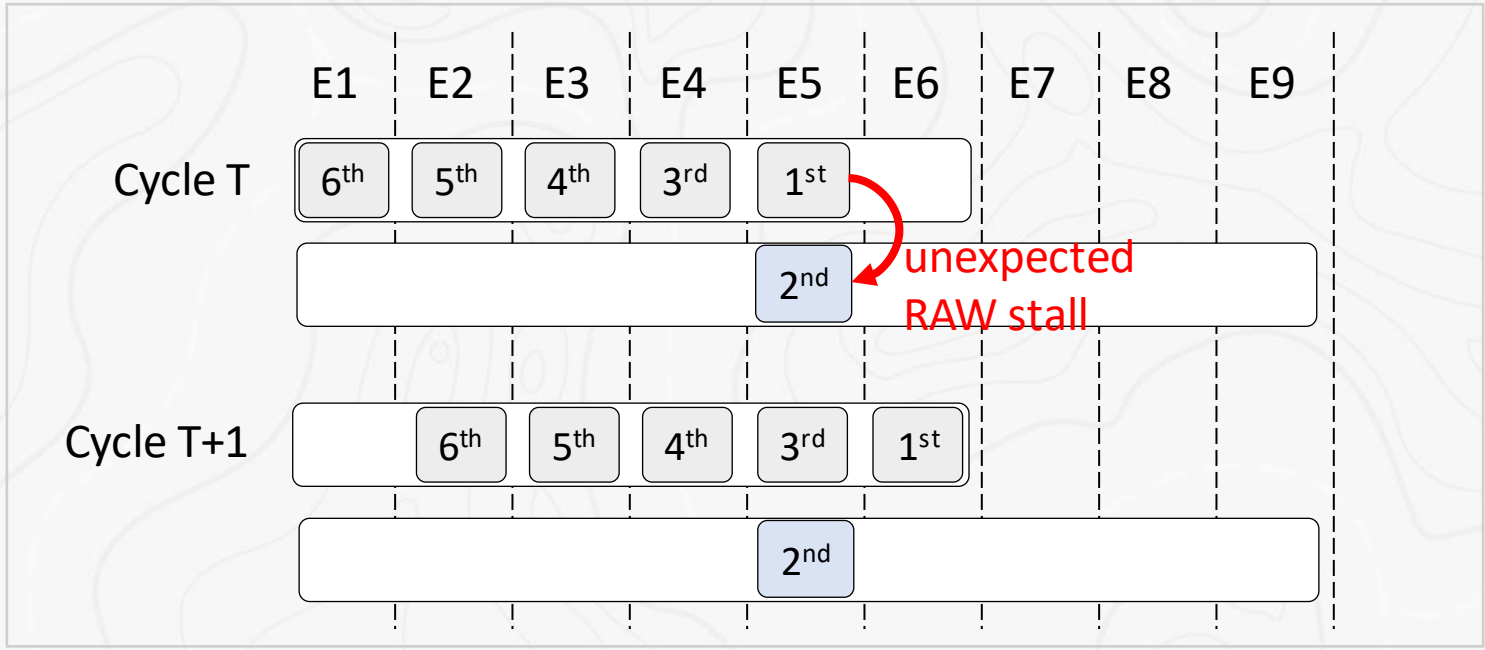
```
property ast_vmv_nfr_eventually_retire;
    logic [4:0] colored_idx;
    @ (posedge clk)) disable iff (! rstn)
    (insn_vld_e1 & insn_is_vmv_nfr_e1, colored_idx=insn_idx_e1) |->
    ## [8:30] (insn_wr_vrf & vrf_idx==colorder_idx);
endproperty
rvv_ast_vmv_nfr_eventually_retire: assert property (ast_vmv_nfr_eventually_retire);
```

Function Bug Example

- Assert property: instruction ‘vnclip.qv’ cannot be retired forever.
- Root cause: design didn’t consider LMUL when handling RAW stall
  - Consider double narrowing, ‘vnclip.qv’ needed v0, v1, v2 and v3, total 4 registers.
  - Consider LMUL=MF4, ‘vnclip.qv’ only need v0.

Instruction sequence

1st vlw.v v1, (x2)
2nd vnclip.qv, v8, v0 (MF4)
3rd vlw.v v2, (x2)
4th vlw.v v1, (x2)
5th vlw.v v2, (x2)
6th vlw.v v1, (x2)
… …

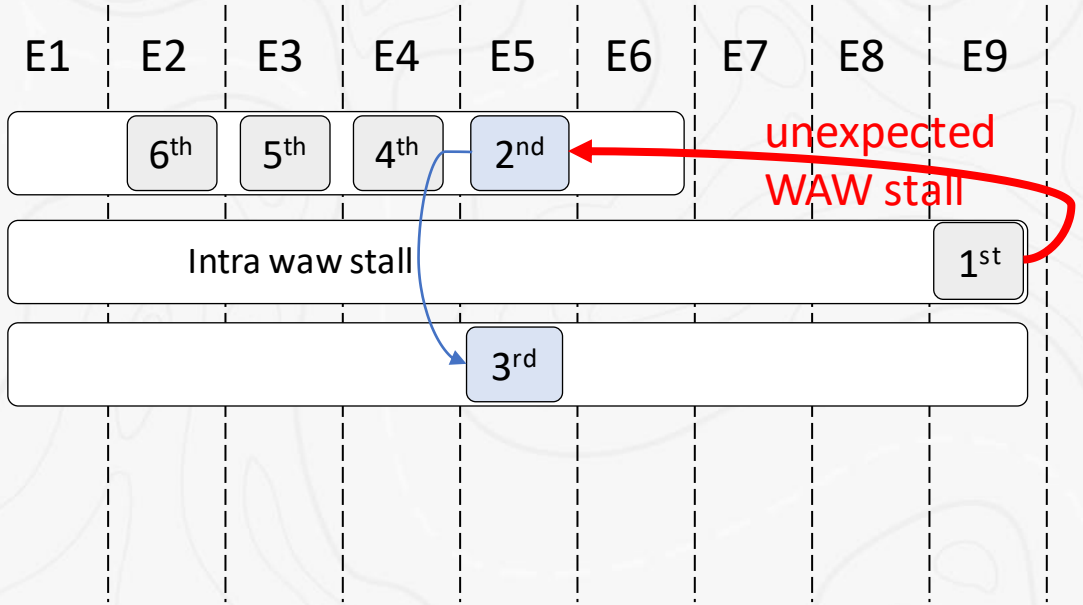


Performance Bug Example

- Cover property detected unexpected long stall
- Register retired order
  - Expected: v8 (vlw) -> v4 (vadd) -> v8 (vmac), totally need 10 cycles
  - Actual: v4(vadd) -> v8 (vlw) -> v8(vmac), totally need 14 cycles.

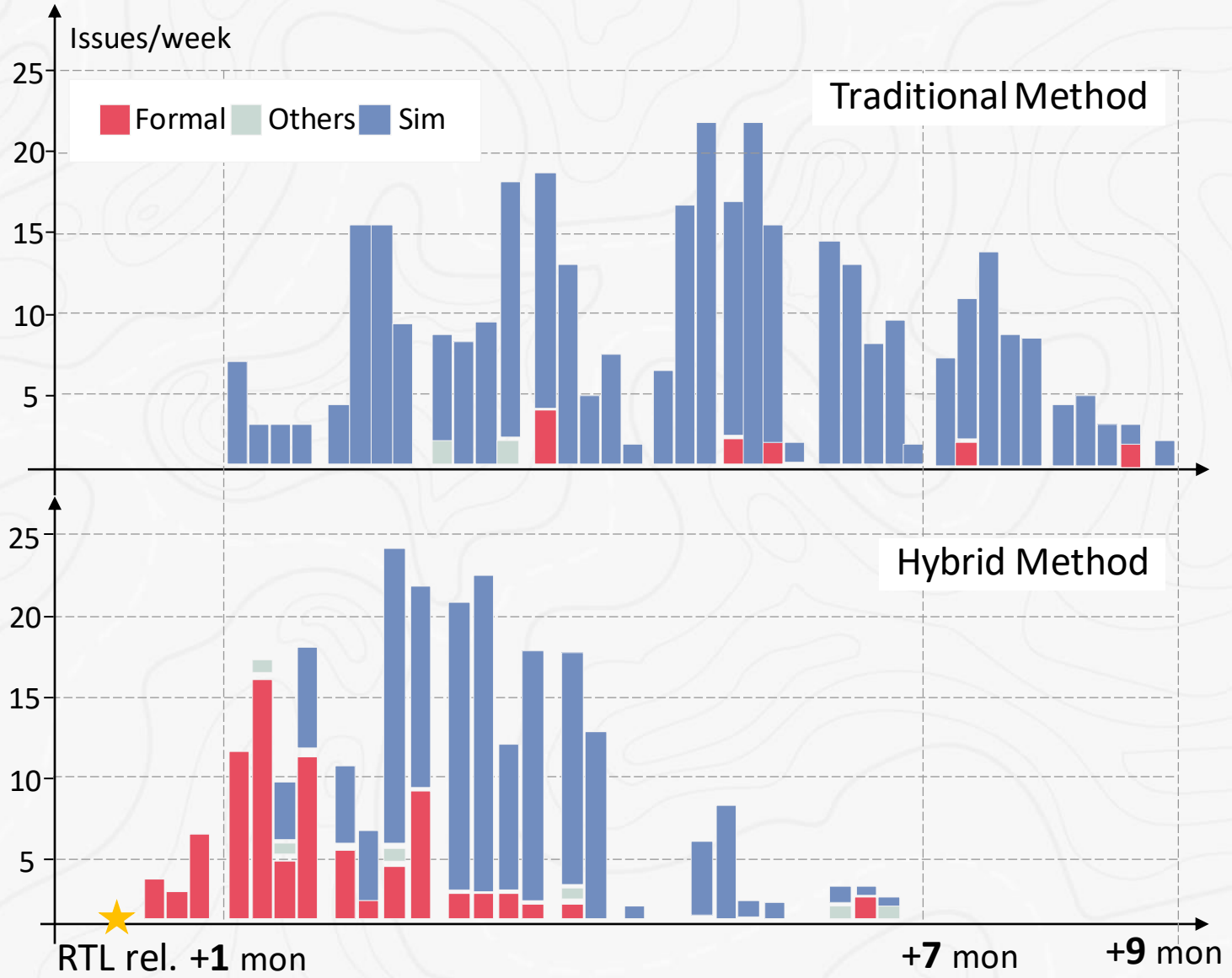
Instruction sequence

1st vadd v4, v2, v0
2nd vlw.v v8, (x2)
3rd vmac v8, v16, v24
4th vlb.v v10, (x3)
5th vlh.v v22, (x6)
6th vlw.v v27, (x7)



Results

- Early-stage bug hunting
- Signoff schedule shift left
- Better design quality



Summary

- YAML-based instruction model auto-generation flow responds well to RISC-V evolution and customization changes.
- Exceptions thoroughly verified with UVM-based solution.
- Hazard handling schedule shift left with simulation & formal hybrid solution.

Questions

- Any questions ?