



Acceleration of Complex RISC-V Processor Verification Using Hardware- Assisted Test Solution

Aimee Sutton (aimees@synopsys.com)

Product Management Sr. Director

November 24, 2025



Verification to Meet the Needs of Today's RISC-V

RISC-V Today

- Complex extensions (e.g. vector, hypervisor)
- Virtual memory
- MMU, TLB
- Multi-level cache hierarchies
- Interrupts: AIA / IMSIC
- Multi-hart, multi-issue, out-of-order execution



Verification Requirements

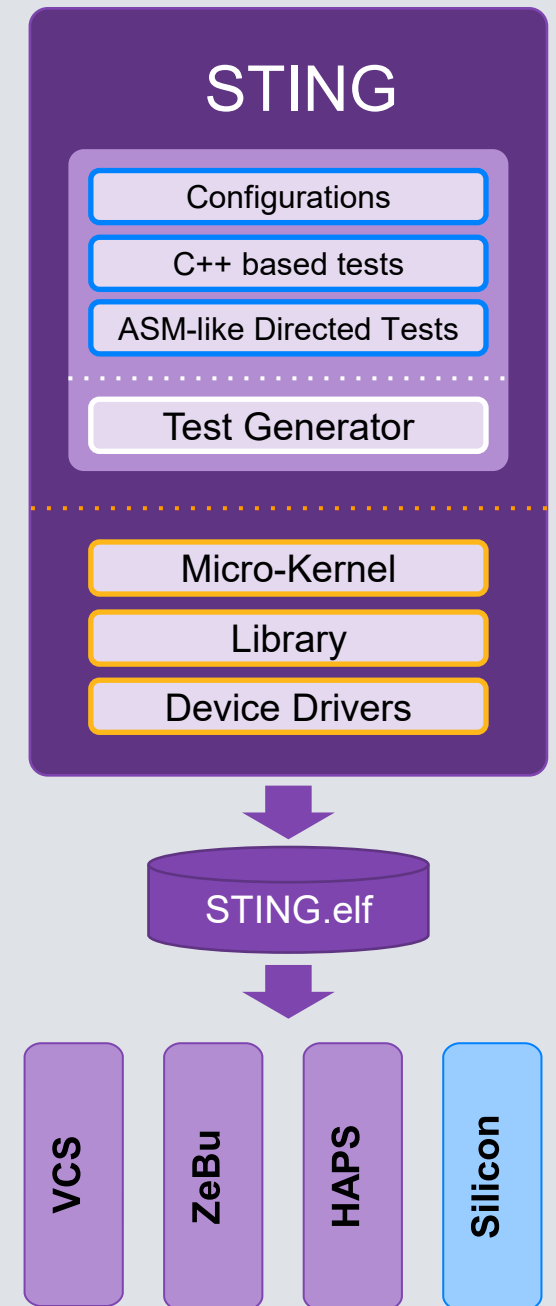
- Test stimulus based on a thorough understanding of the RISC-V spec
- Complex and long-running tests to get the design into interesting states
- A fast execution platform to achieve verification closure in a reasonable time

To verify a typical high-end RISC-V core, it takes a staggering number of cycles – on the order of **10^{15} (one quadrillion)**

STING – Bare Metal Test Generator for RISC-V

Silicon-Proven RISC-V Processor Verification

- Bare metal tool using a software driven methodology for RISC-V design verification
- Integrates several test generation methodologies to give the best verification coverage and productivity:
 - Random stimulus, workloads, directed tests, real world scenarios
- Generates both self-checking and pure stimulus tests
 - Tests are portable across simulation, emulation, FPGA and silicon
- Complete support for 32-bit and 64-bit RISC-V specifications
 - All ratified extensions, and stable, unrated extensions
 - Comprehensive coverage of privilege specification: MMU, PMP, PMA, Hypervisor, Supervisor, CSRs
 - Compatible with any system configuration / memory map
 - Supports multi-hart, multi-processor designs
 - Support for RVA22 and RVA23 profiles
- Includes a library of over 100,000 test fragments (snippets)



Stimulus for Advanced RISC-V Implementations

- Complex implementations need longer and more stressful workloads to build interesting microarchitectural states
- Most areas like cache coherency, ordering and OS use cases need workloads - long running test sequences to get desired coverage.
- For MP platforms, same test sequence is advised to be repeatedly executed to cover all possible combinations of instruction scheduling across the processors

Algorithmic

- Atomicity Tests
- Synchronization Tests (Dekker)
- Forward Progress
- Code Modification

Ordering

- Litmus Tests
- Fence Ordering Testing
- Random Memory Ordering Tests

Random

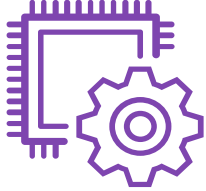
- Conflicting Traffic
- Non-conflicting Traffic
- ISA Coverage

Cache Coherency

- True Sharing
- False Sharing
- Cache fills, evictions

Fast execution platform is needed for complex, long-running stimulus

Hardware-Assisted Test Solution for RISC-V



Hardware Assisted Verification

The DUT is synthesized and run at a high rate of speed using Synopsys ZeBu or HAPS (emulation and prototyping hardware)



STING generates self-checking tests

Reference model results (ImperasFPM) embedded in the elf file



Generate many tests in parallel

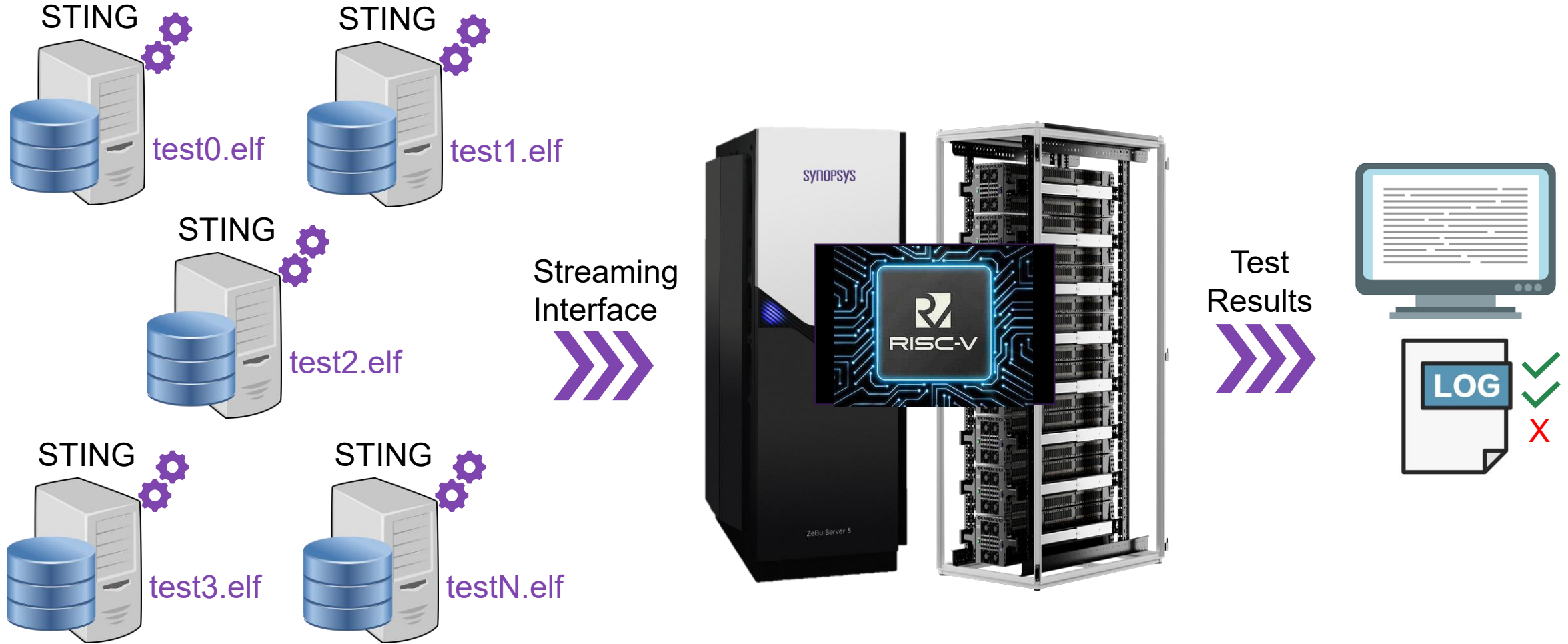
To keep up with the speed of hardware execution



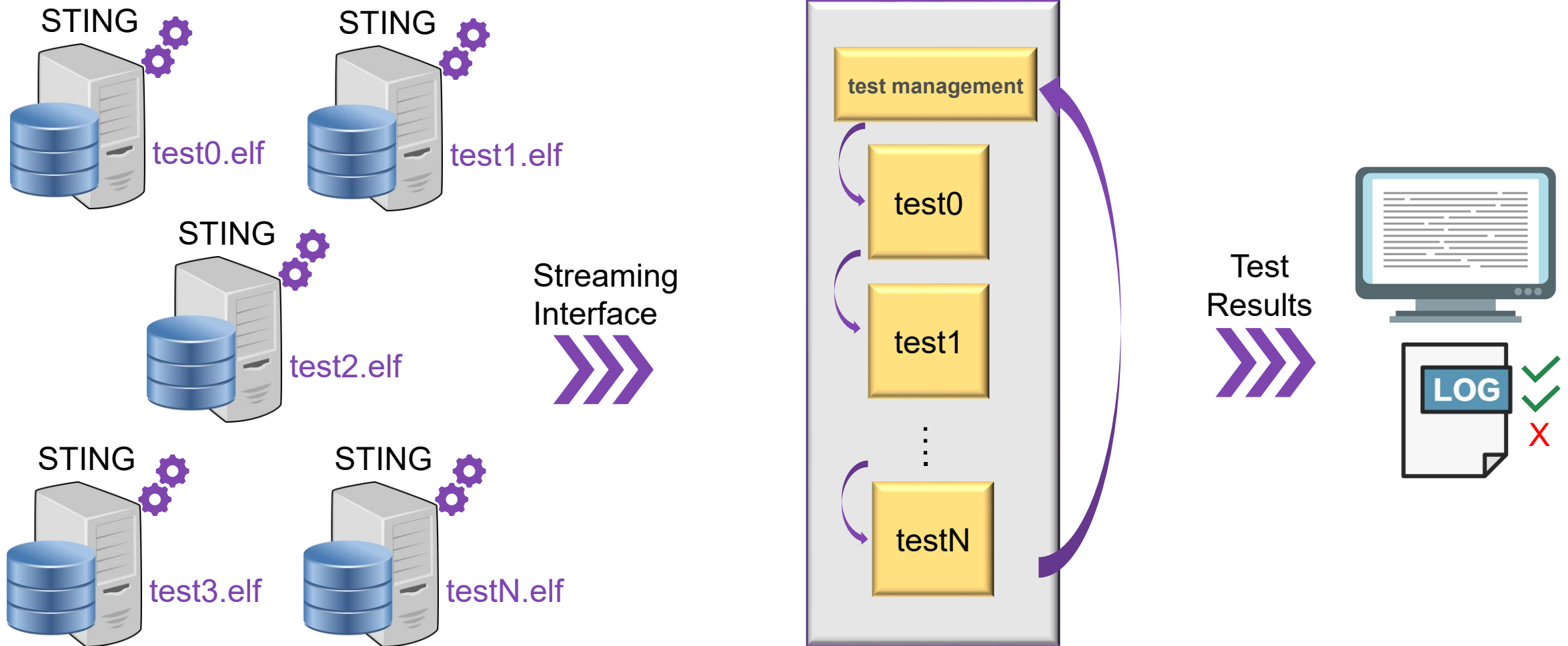
Continuously populate new tests in memory

Using a streaming interface maximizes test throughput

HW-Assisted Test Solution for RISC-V : How it works



HW-Assisted Test Solution for RISC-V : How it works



Streaming enables significant improvement in verification throughput

- **Performance benefits through:**
 - Avoiding re-initialization of Zebu HW
 - Avoiding redundant configuration cycles per test
 - Concurrent generation and execution
- Results across 70 STING tests

	Real	User	sys
Streaming	277.98	600.72	367.23
Non streaming	1081.26	1189.22	451.07

- **4X in real time, 2X in User time**
- Simulation to Emulation improvement : **~6000X per test**

HW Assisted TS for RISC-V: Debug methodology

- Steps to debug a failing test

1. Re-run a specific streaming-enabled test
 - This may not reproduce the failure because the microarchitecture is in a "clean" state
2. Re-run the same test multiple times
 - This may not reproduce the failure if it is due to an artifact from a different test
3. Re-run the same sequence of N tests
 - Increase the number of tests in the sequence until the failure is identified

```
[ST_INFO] set stream wait time to 60
[ST_INFO] Wait TEST_START
[ST_INFO] test: 0 ../saved/run_result/round_33/sting.m000.only_ld_add.3920044246335413322.elf
[ST_INFO] test: 1 ../saved/run_result/round_33/sting.m001.only_st_c_sub.7632128932872382806.elf
[ST_INFO] test: 2 ../saved/run_result/round_33/sting.m002.only_st_c_srui.1988591093057426060.elf
[ST_INFO] test: 3 ../saved/run_result/round_34/sting.m000.only_ldst_c_addi.4526835087824188500.elf
[ST_INFO] test: 4 ../saved/run_result/round_34/sting.m001.only_div_lb.12645179519919719249.elf
[ST_INFO] test: 5 ../saved/run_result/round_34/sting.m002.stores_after_loads.568299493738325299.elf
[ST_INFO] test: 6 ../saved/run_result/round_35/sting.m000.only_arith_lbu.4430798757004886781.elf
[ST_INFO] load flush elf
[ST_WARNING] ../saved/run_result already existed
[ST_INFO] ELF target directory is ready: ../saved/run_result
[ST_INFO] Stream_debug_sequence: run 2 round and 1 tests
[ST_INFO] trigger TEST_START

start to run
[ST_INFO] ST_Stream::wait_st_test
[ST_INFO] Thread is still running. Waiting...
Flush!
Test Name : sting.m000.only_ld_add.3920044246335413322.elf
T: 0

STING Test Status: PASS

-
Test Name : sting.m001.only_st_c_sub.7632128932872382806.elf
T: 0

STING Test Status: PASS

-
Test Name : sting.m002.only_st_c_srui.1988591093057426060.elf
T: 0

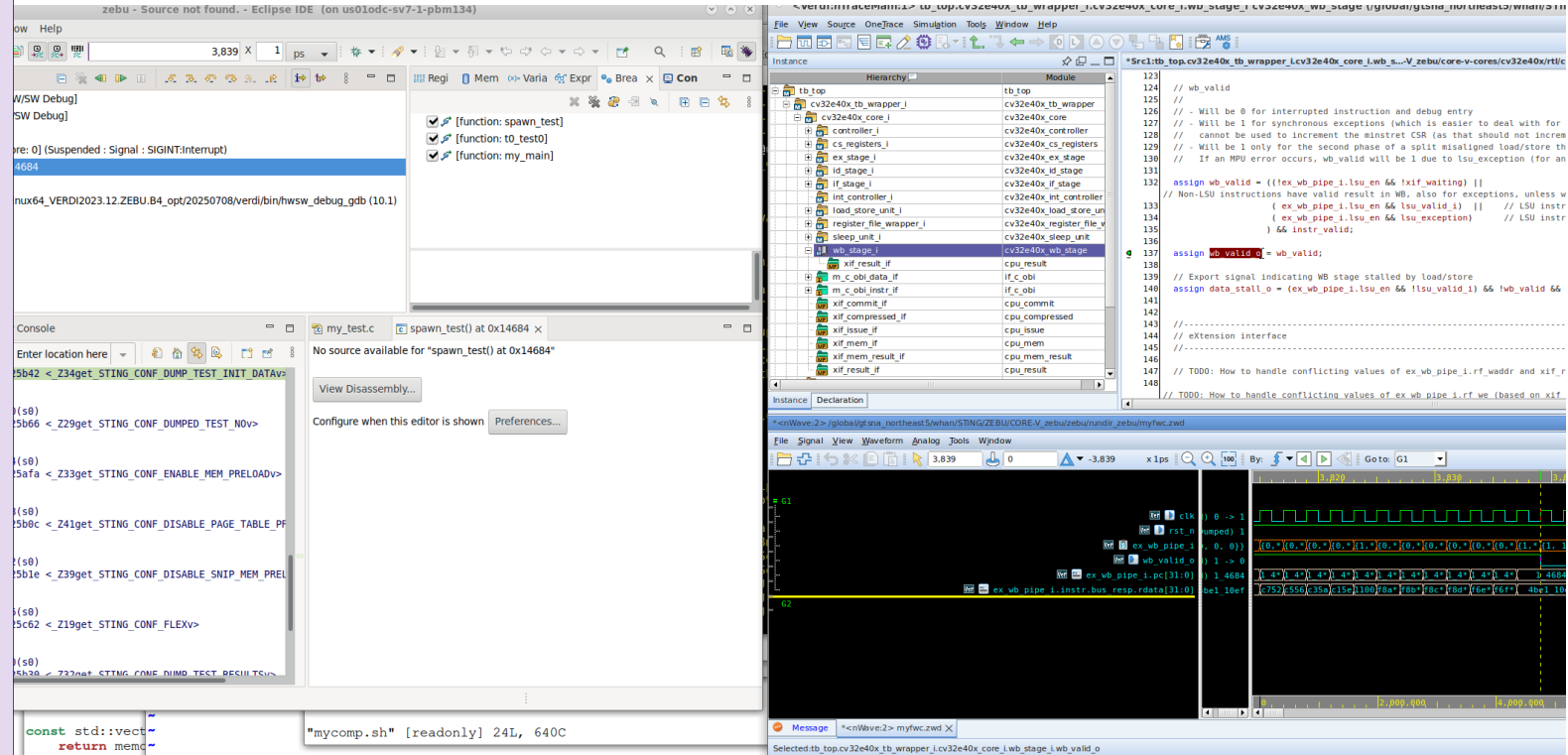
STING Test Status: PASS

-
Flush!
[ST_INFO] Ready for new round: 2
```

```
streaming_enabled: yes
streaming_debug_sequence: yes
streaming_debug_sequence_nof_test: 7
streaming_debug_sequence_elf_file_name: ../saved/run_result/round_35/sting.m000.only_arith_lbu.4430798757004886781.elf
elf_target_dir: ../saved/run_result
nof_test: 3
```

Debug test failures using Verdi Hardware/Software Debug

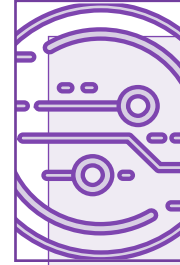
- CPU trace data is extracted from the test execution (PC, GPRs, etc.)
- Waveform data is dumped from the hardware platform
- Test can be replayed in the debugger using breakpoints, single stepping
- Waveform is synchronized with the program execution



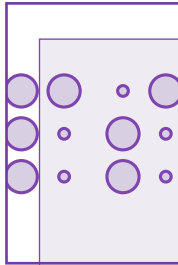
Future development



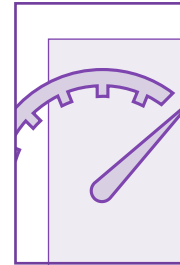
Integration of
ImperasFPM and
checking



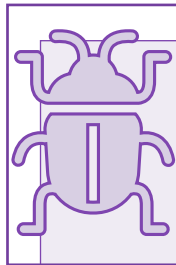
STING workload for
memory ordering
verification



Integration of
functional coverage



Streaming
performance
improvement



Debug automation

Summary

Hardware-Assisted Verification plays a critical role in RISC-V processor DV

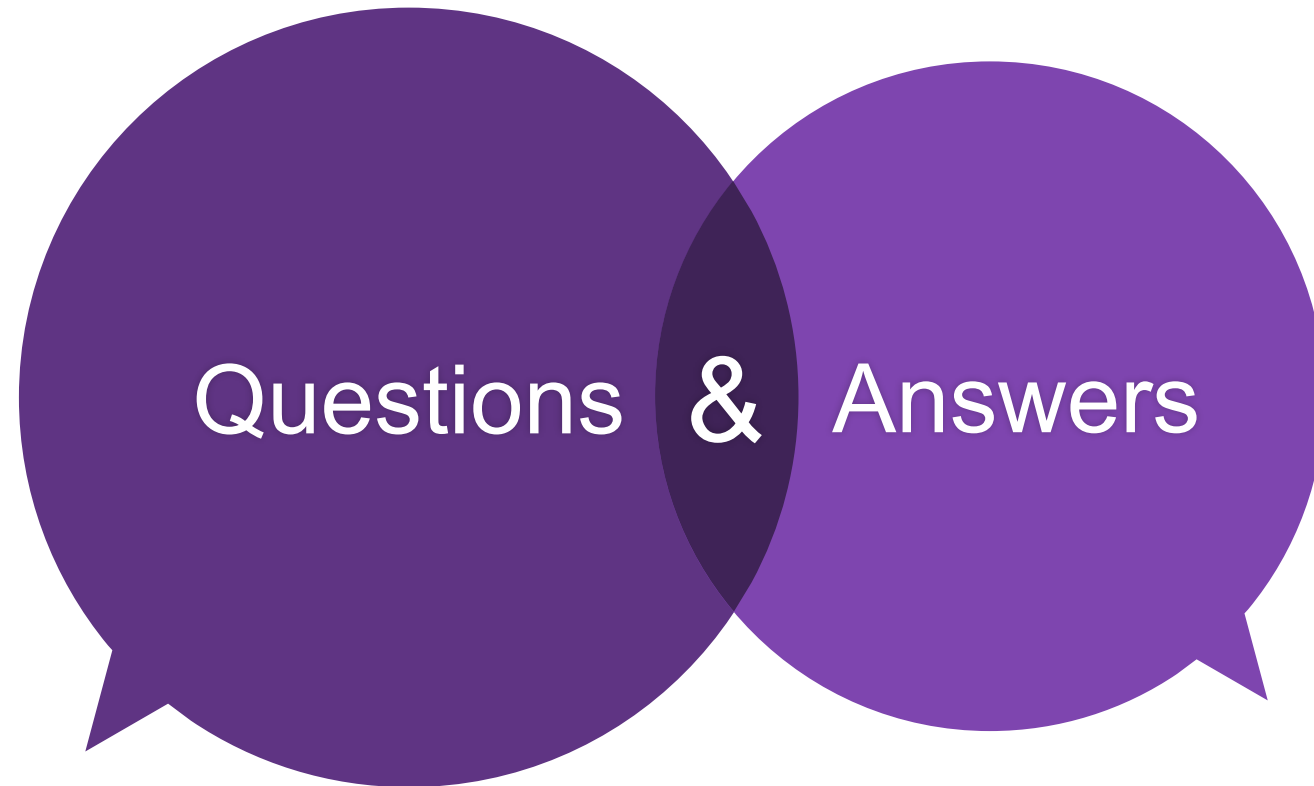
High performance is needed to address the verification cycle challenge

HW-Assisted Test Solution for RISC-V combines a high-performance test generation technology (STING) with Synopsys ZeBu and HAPS platforms

Delivering comprehensive test stimulus and high verification throughput

Hardware/Software debug with Verdi enables concurrent debug of test program with DUT waveforms





THANK YOU

synopsys®