Acceleration of Complex RISC-V Processor Verification Using Test Generation Integrated with Hardware Emulation

Wei-Hua Han

Legal Disclosure

CONFIDENTIAL INFORMATION

The information contained in this presentation is the confidential and proprietary information of Synopsys. You are not permitted to disseminate or use any of the information provided to you in this presentation outside of Synopsys without prior written authorization.

IMPORTANT NOTICE

This presentation may include information related to Synopsys' future product or business plans. Such plans are as of the date of this presentation and subject to change. Synopsys is not obligated to update this presentation or develop the products with the features and/or functionality discussed in this presentation. Additionally, Synopsys' products and services may only be offered and purchased pursuant to an authorized quote and purchase order or a mutually agreed upon written contract.

FORWARD LOOKING STATEMENTS

This presentation may include certain statements including, but not limited to, Synopsys' financial targets, expectations and objectives; business and market outlook, business opportunities, strategies and technological trends; and more. These statements are made only as of the date hereof and subject to change. Actual results or events could differ materially from those anticipated in such statements due to a number of factors. Synopsys undertakes no duty to, and does not intend to, update any statement in this presentation, whether as a result of new information, future events or otherwise, unless required by law.

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

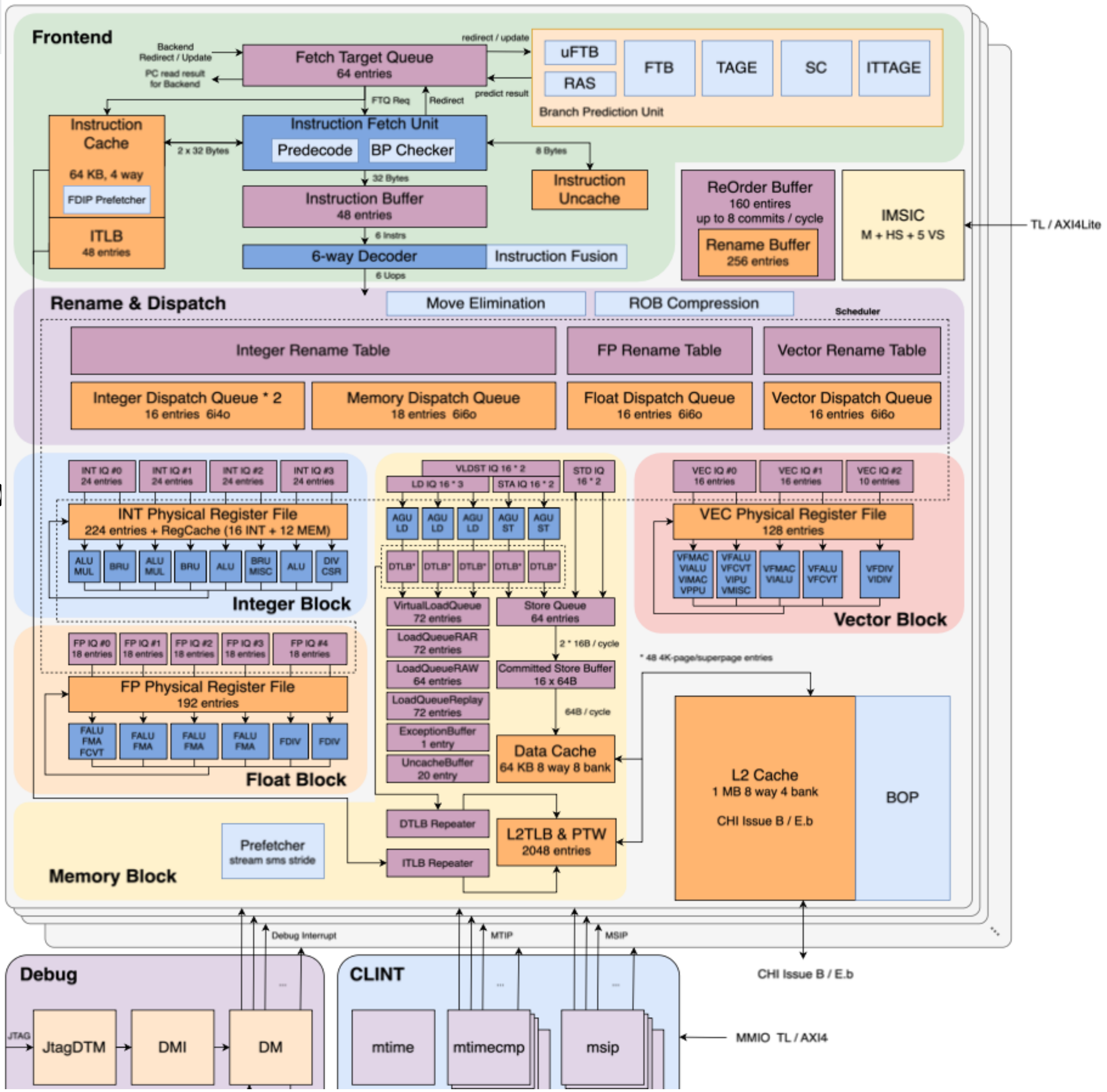
Verification Requirements

- 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)**

DUT: XiangShan RISC-V Core

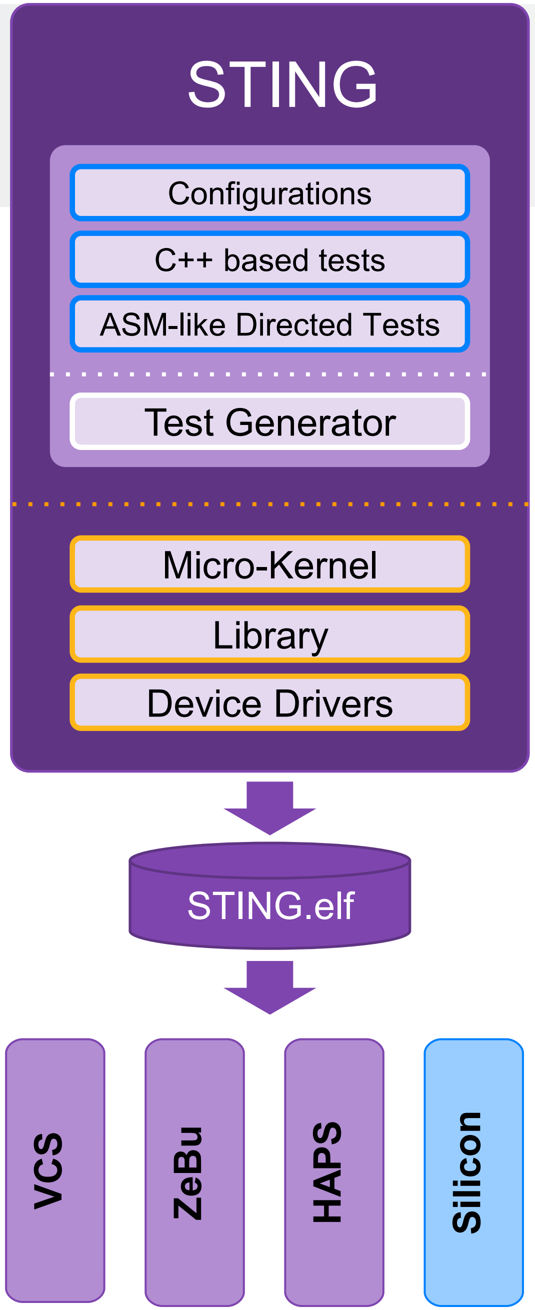
- Supports RV64 and its extended instruction set
- Supports RVV 1.0, VLEN 128bit x 2.
- Supports unaligned access to Cacheable space
- Memory Management Unit (MMU)
- Up to 48-bit physical addresses, and 39-bit and 48-b
- Timer interrupts and the RVA23-Sstc feature
- ICache 64KB, supports Parity
- DCache, up to 64KB, supports ECC
- Unified L2, up to 1MB, supports ECC
- Supports Level 1 and Level 2 TLB
- CRS/IMSIC compliant with AIA 1.0



STING – Bare Metal Test Generator for RISC-V

- 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, unratified 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)

Silicon-Proven RISC-V Processor Verification



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 emulation or prototyping hardware (e.g. Synopsys ZeBu or HAPS)

STING generates self-checking tests

Reference model results (e.g. ImperasFPM) embedded in the elf file

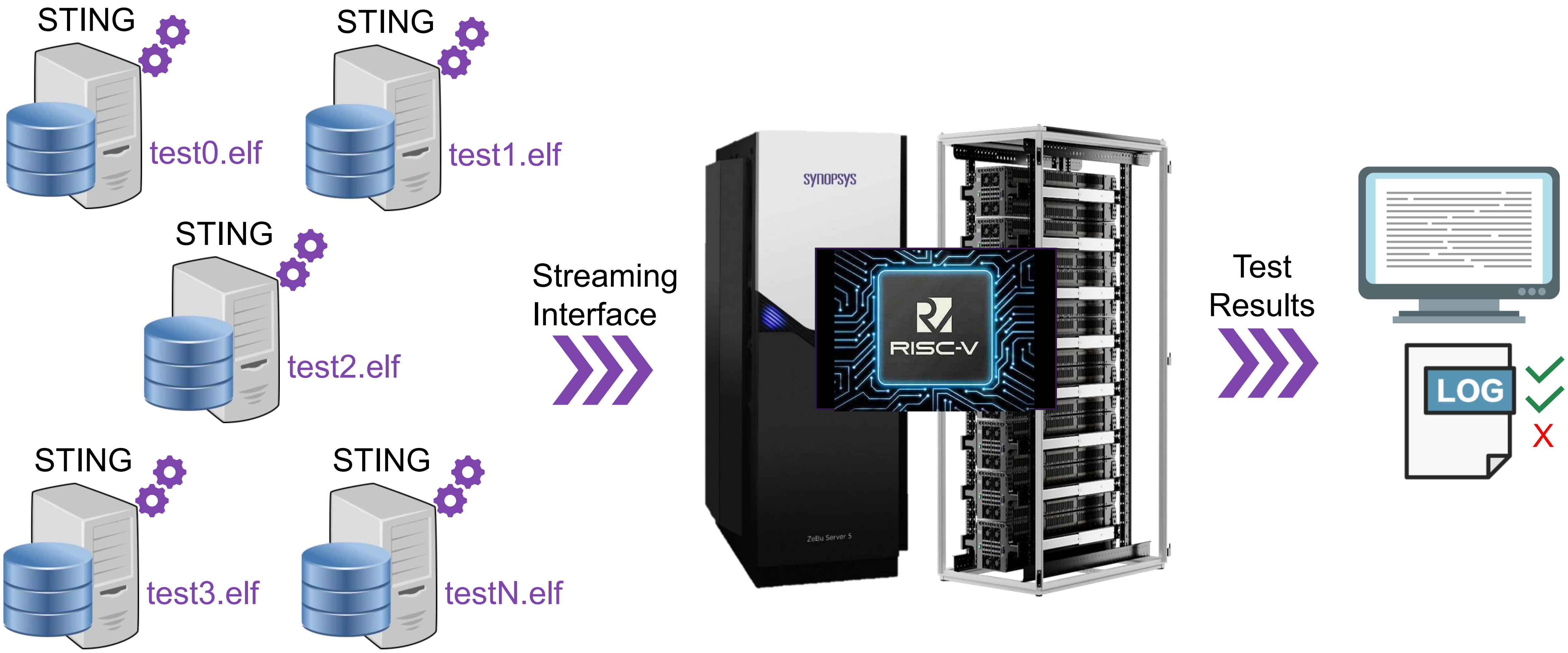
Generate many tests in parallel

To keep up with the speed of hardware execution

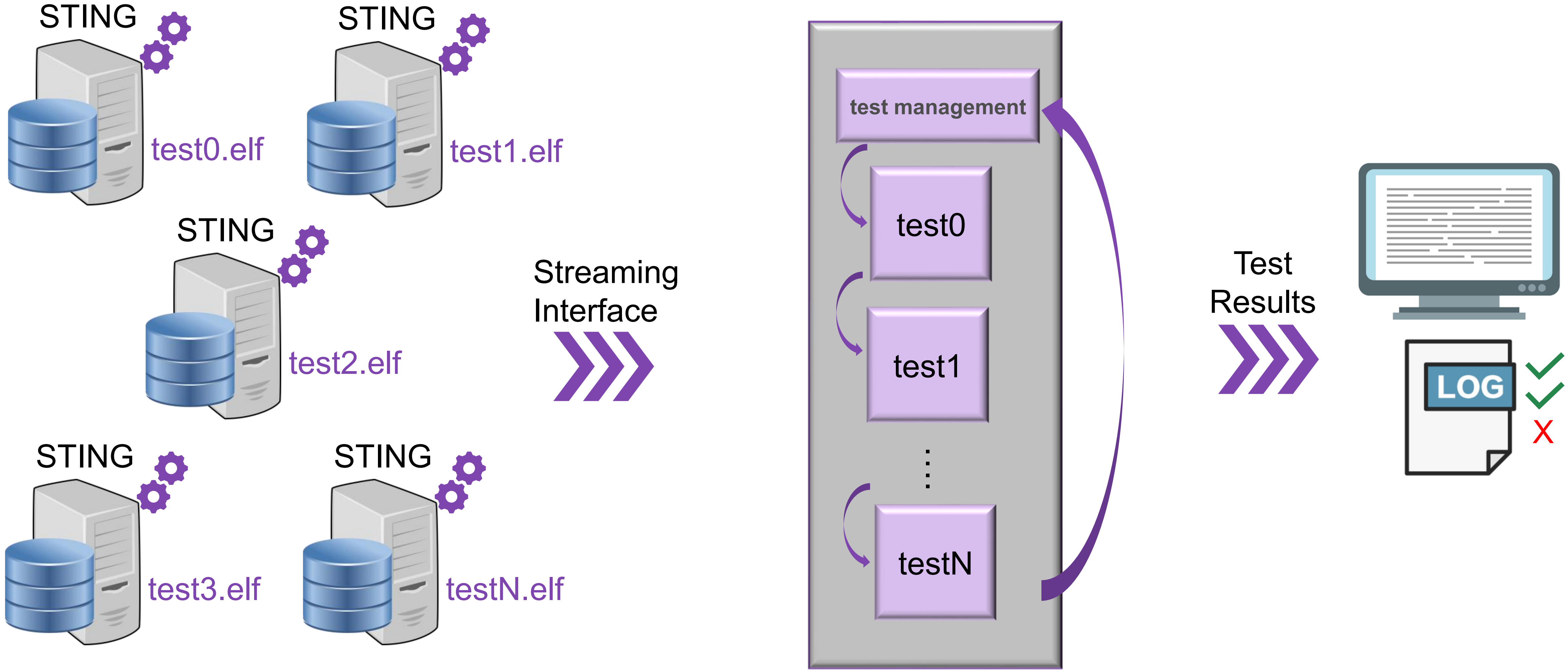
Continuously populate new tests in memory

Using a streaming interface maximizes test throughput

Hardware-Assisted Test Solution for RISC-V



H-A Test Solution for RISC-V: How it works



Streaming methodology enables significant performance improvement

- **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

- Simulation to Emulation improvement : **~6000X per test**

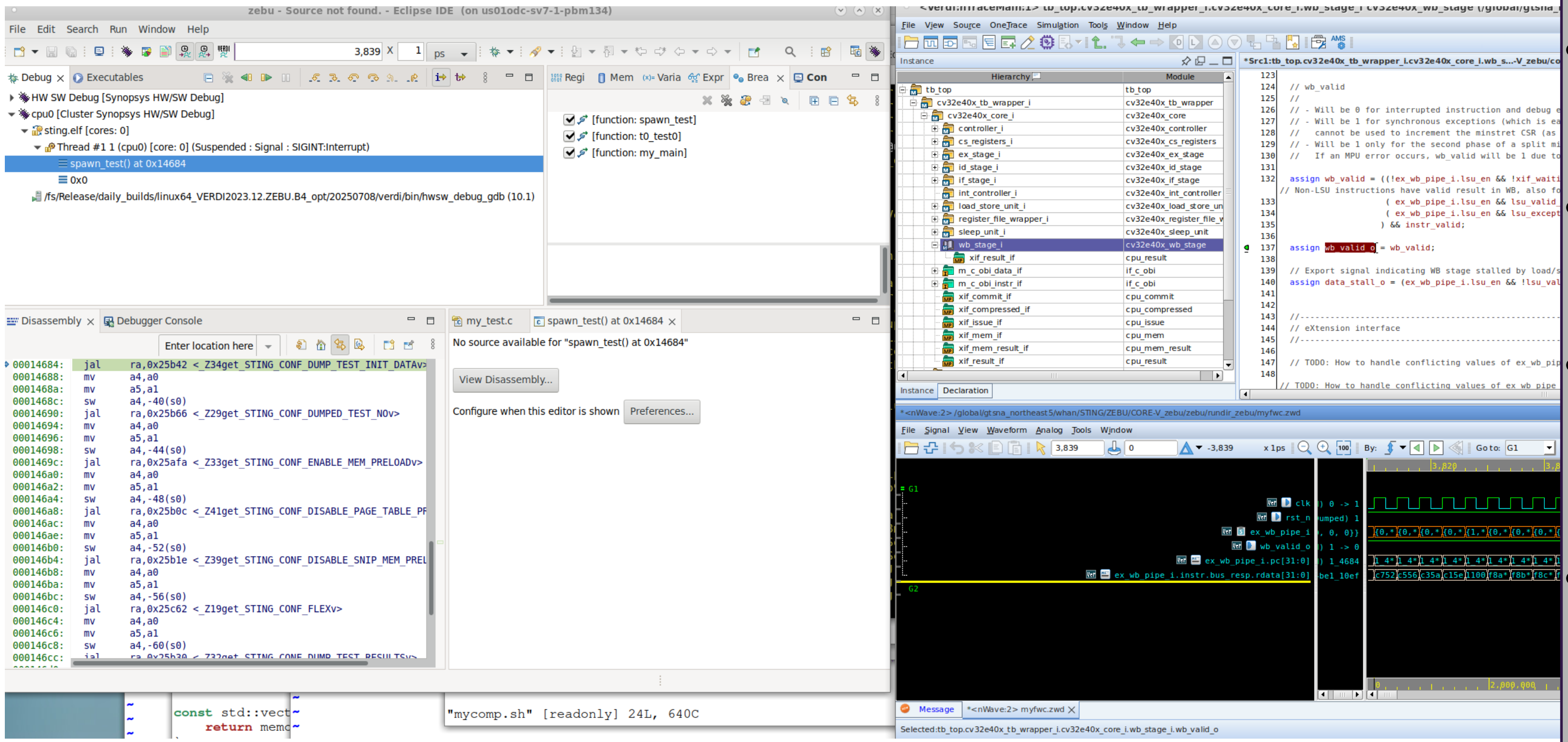
H-A TS for RISC-V: Debug methodology

- 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_srai.1988591093057426060.el
[ST_INFO] test: 3 ../saved/run_result/round_34/sting.m000.only_ldst_c_addi.4526835087824188500.
[ST_INFO] test: 4 ../saved/run_result/round_34/sting.m001.only_div_lb.1264517951919719249.elf
[ST_INFO] test: 5 ../saved/run_result/round_34/sting.m002.stores_after_loads.568299493738325299
[ST_INFO] test: 6 ../saved/run_result/round_35/sting.m000.only_arith_lbu.4430798757004886781.el
[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_srai.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 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
- Integration of functional coverage
- Streaming performance improvement
- Debug automation

Summary

HAV plays a critical role in high quality RISC-V processor verification

- High performance is needed to address the verification cycle challenge

HAV Test Solution for RISC-V combines a high-performance test generation technology (STING) with a hardware-assisted verification platform

- Resulting in comprehensive stimulus and high verification throughput

Debug techniques are needed to efficiently address failures found in high-speed regression runs

- Hardware/Software debug is an effective approach enabling concurrent debug of test program with DUT waveforms

Thank you