BREKER™

RISC-V Core & SoC Verification: A New Normal in Verification Techniques

Verification Futures Reading 2024

Adnan Hamid

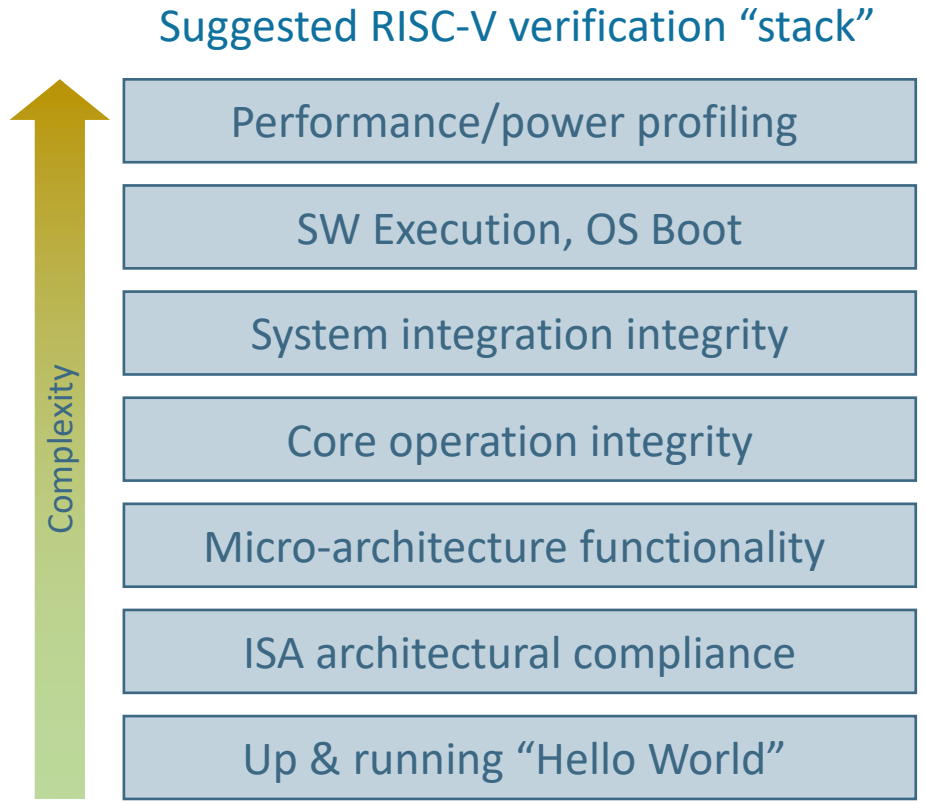
A Look At RISC-V

- Open Instruction Set Architecture (ISA) gaining significant traction in multiple applications
- Significant verification challenges
  - Arm spends $150M per year on 10^{15} verification cycles per core
  - Hard for RISC-V development group to achieve this same quality
  - Lots of applications expands verification requirements
  - Requires automation, reuse and new thinking
- RISC-V International developing certification committee to certify devices against architectural compliance tests

© Breker Verification Systems, Inc. All rights reserved.

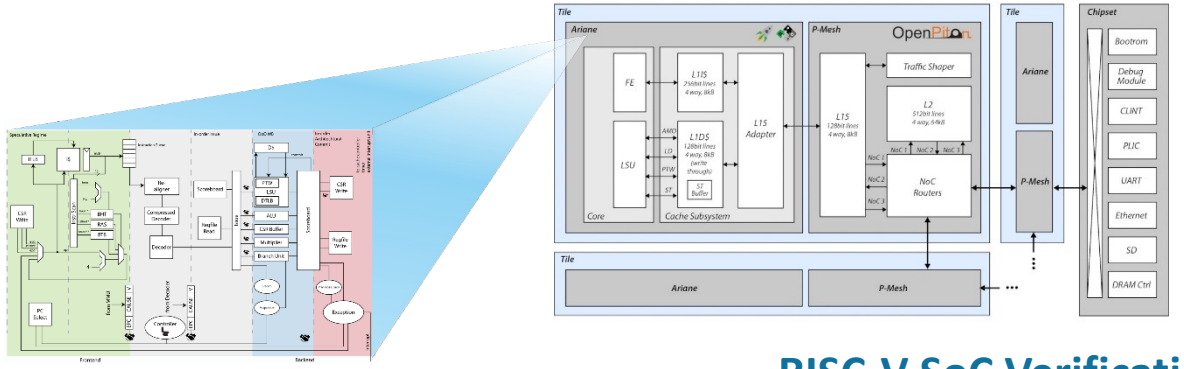
Meeting RISC-V Verification Challenges

- Reuse & automation to meet quality expectation
  - Automated test generation key
- RISC-V special requirements
  - Custom instruction verification
  - Compliance assurance
  - Broad range of architectures
- Different processors have different needs
  - Embedded cores
  - Processor clusters
  - Application processors



© Breker Verification Systems, Inc. All rights reserved.

Different Challenges for Core vs SoC Verification



RISC-V Core Verification Challenges

Random Instructions	Do instructions yield correct results
Register/Register Hazards	Pipeline perturbations dues to register conflicts
Load/Store Integrity	Memory conflict patterns
Conditionals and Branches	Pipeline perturbations from synchronous PC change
Exceptions	Jumping to and returning from ISR
Asynchronous Interrupts	Pipeline perturbations from asynchronous PC change
Privilege Level Switching	Context switching
Core Security	Register and Memory protection by privilege level
Core Paging/MMU	Memory virtualization and TLB operation
Sleep/Wakeup	State retention across WFI
Voltage/Freq Scaling	Operation at different clock ratios
Core Coherency	Caches, evictions and snoops

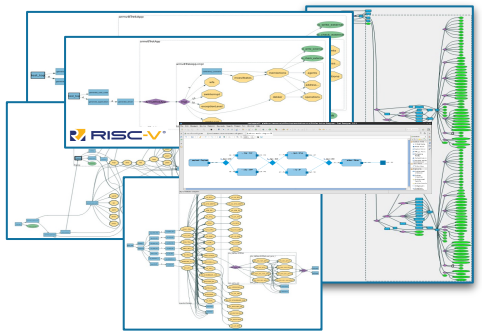
RISC-V SoC Verification Challenges

System Coherency	Cover all cache transitions, evictions, snoops
System Paging/IOMMU	System memory virtualization
System Security	Register and Memory protection across system
Power Management	System wide sleep/wakeup and voltage/freq scaling
Packet Generation	Generating networking packets for I/O testing
Interface Testing	Analyzing coherent interfaces including CXL & UCIe
Random Memory Tests	Test Cores/Fabrics/Memory controllers across DDR, OCRAM, FLASH etc
Random Register Tests	Read/write test to all uncore registers
System Interrupts	Randomized interrupts through CLINT
Multi-core Execution	Concurrent operations on fabric and memory
Memory Ordering	For weakly order memory protocols
Atomic Operation	Across all memory types

© Breker Verification Systems, Inc. All rights reserved.

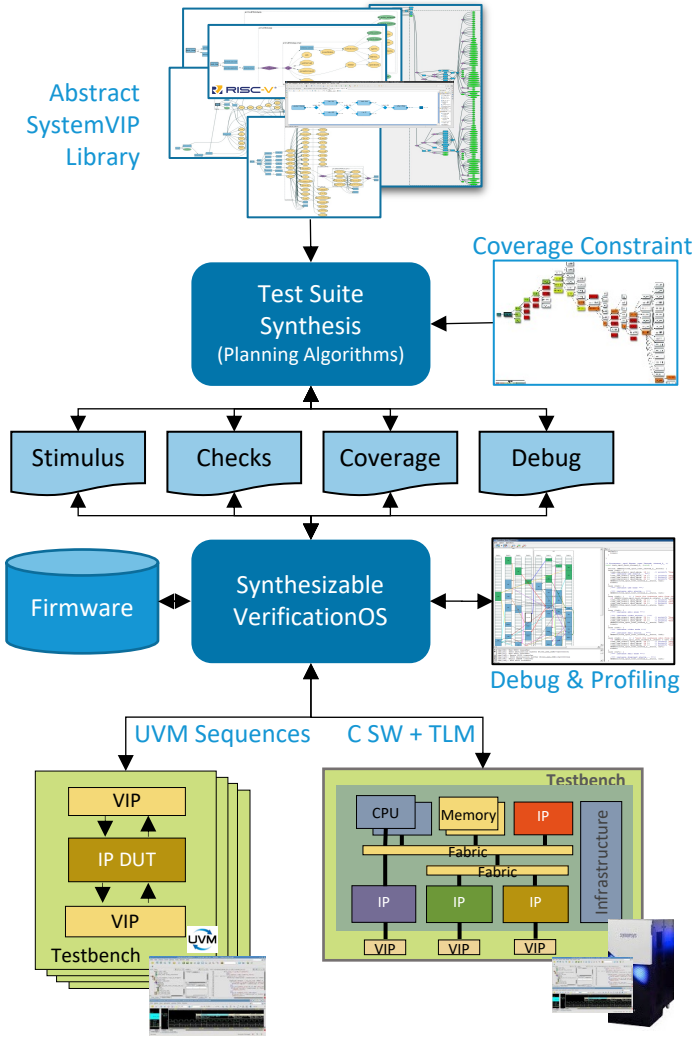
Breker Background:
Test Suite Synthesis for RISC-V Cores & SoCs

- Breker is a key, longstanding part of the verification ecosystem for processors and SoCs based on x86 and Arm architectures
- Breker has become part of the verification ecosystem for processors and SoCs based on RISC-V architectures
  - Working with multiple RISC-V developers and users/integrators
- RISC-V has room to grow if we solve the verification barrier
  - We are experienced in x86 and Arm verification, now are sharing this experience with RISC-V teams through automated tests



The Breker SystemVIP Library

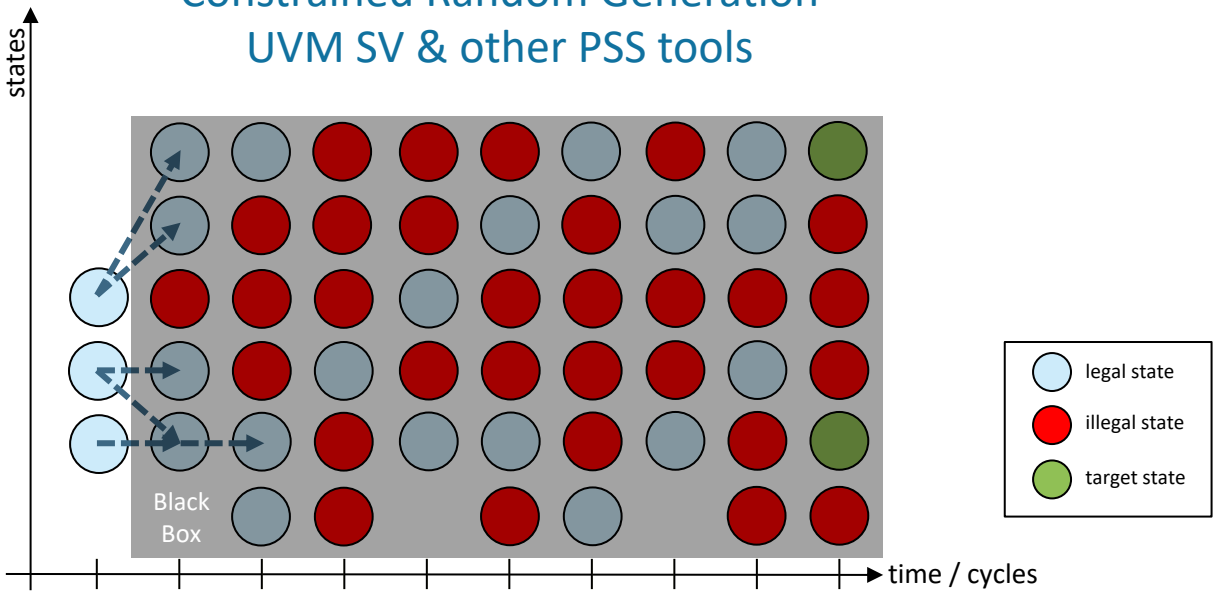
- *Core Integrity FastApps*
- *RISC-V System Integrity TrekApp*
- *ARM System Integrity TrekApp*
- *Cache Coherency TrekApp 2.0*
- *Firmware-First TrekApp*
- *Power Management TrekApp*
- *Security TrekApp*
- *Networking TrekApp*



© Breker Verification Systems, Inc. All rights reserved.

Constrained Random vs AI Planning Algorithm Synthesis

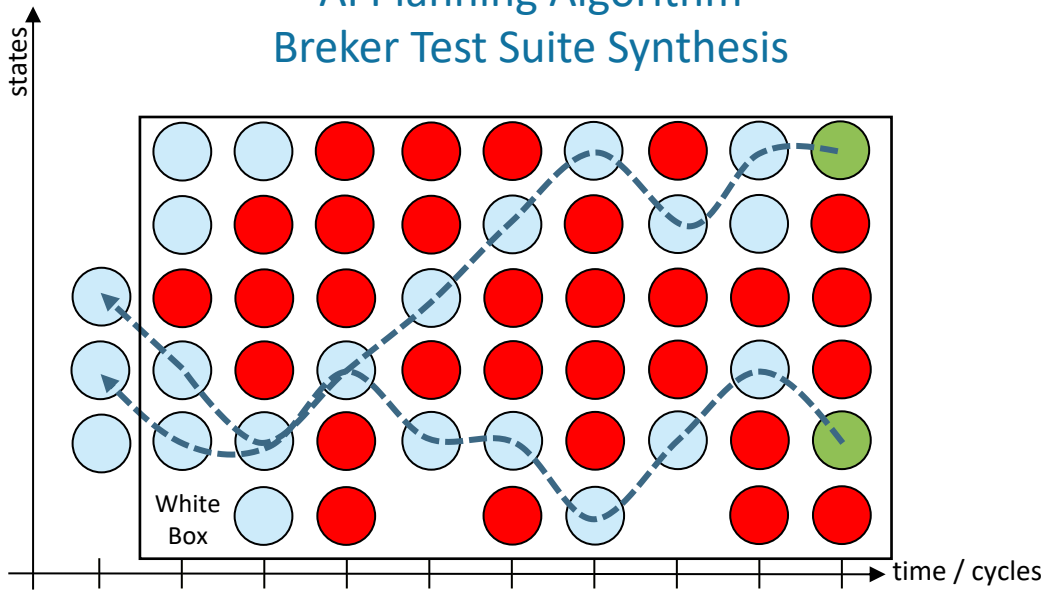
Constrained Random Generation
UVM SV & other PSS tools



Design black box, shotgun tests to search for key state

Low probability of finding complex bug

AI Planning Algorithm
Breker Test Suite Synthesis

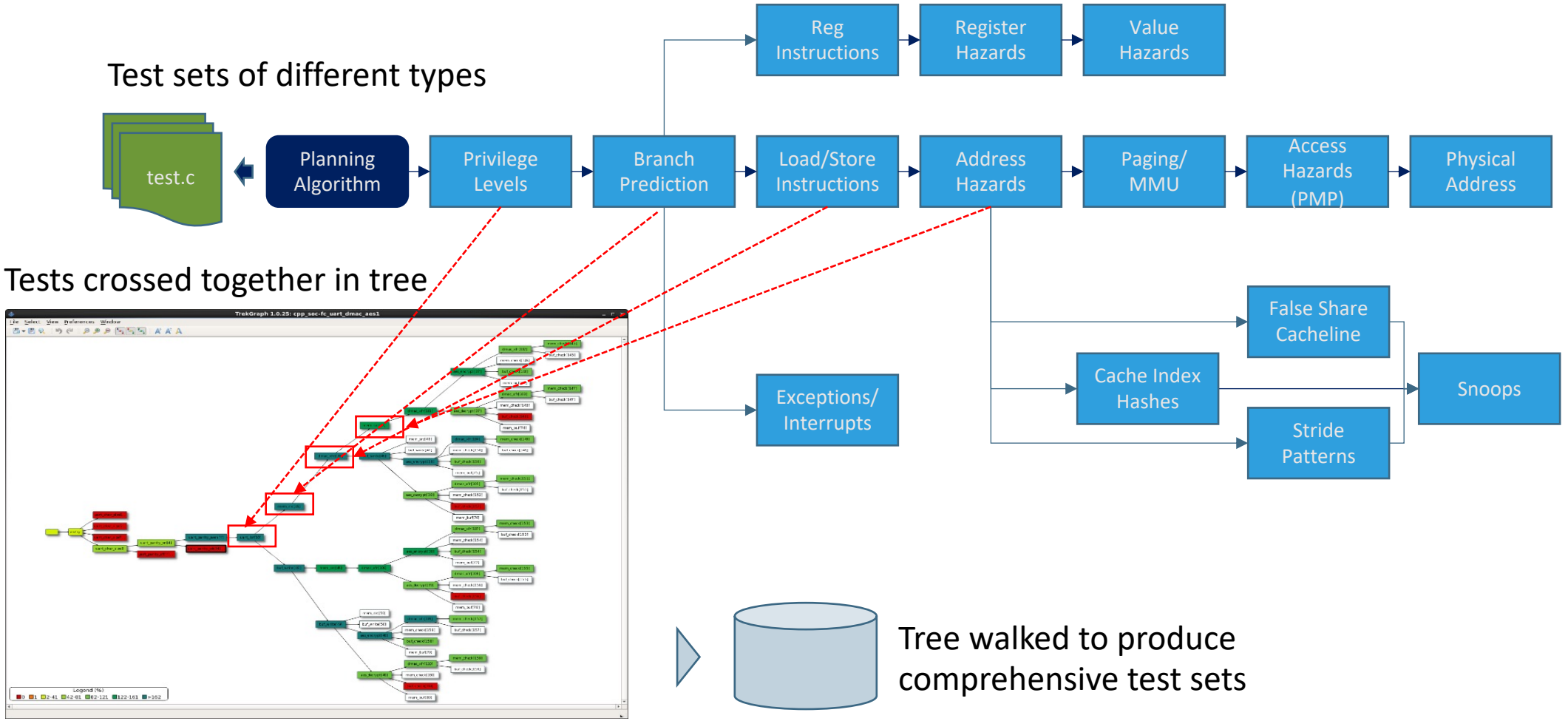


Starts with key state and intelligently works backward through space

Deep sequential, optimized test discovers complex corner-cases

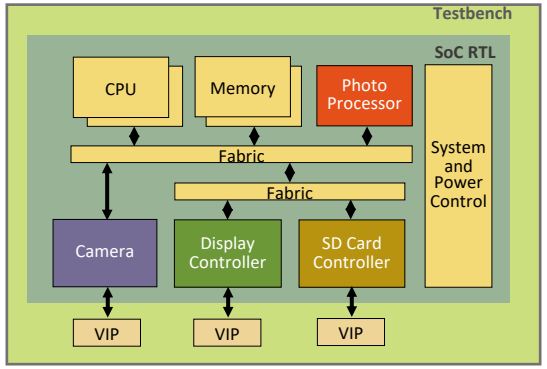
© Breker Verification Systems, Inc. All rights reserved.

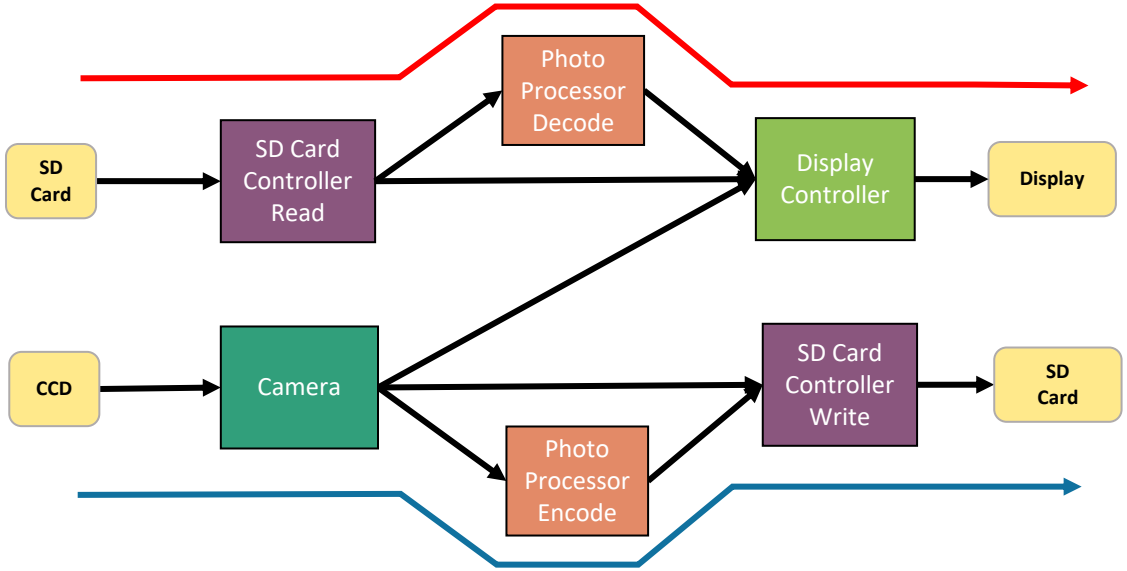
Crossing RISC-V Core Verification Components

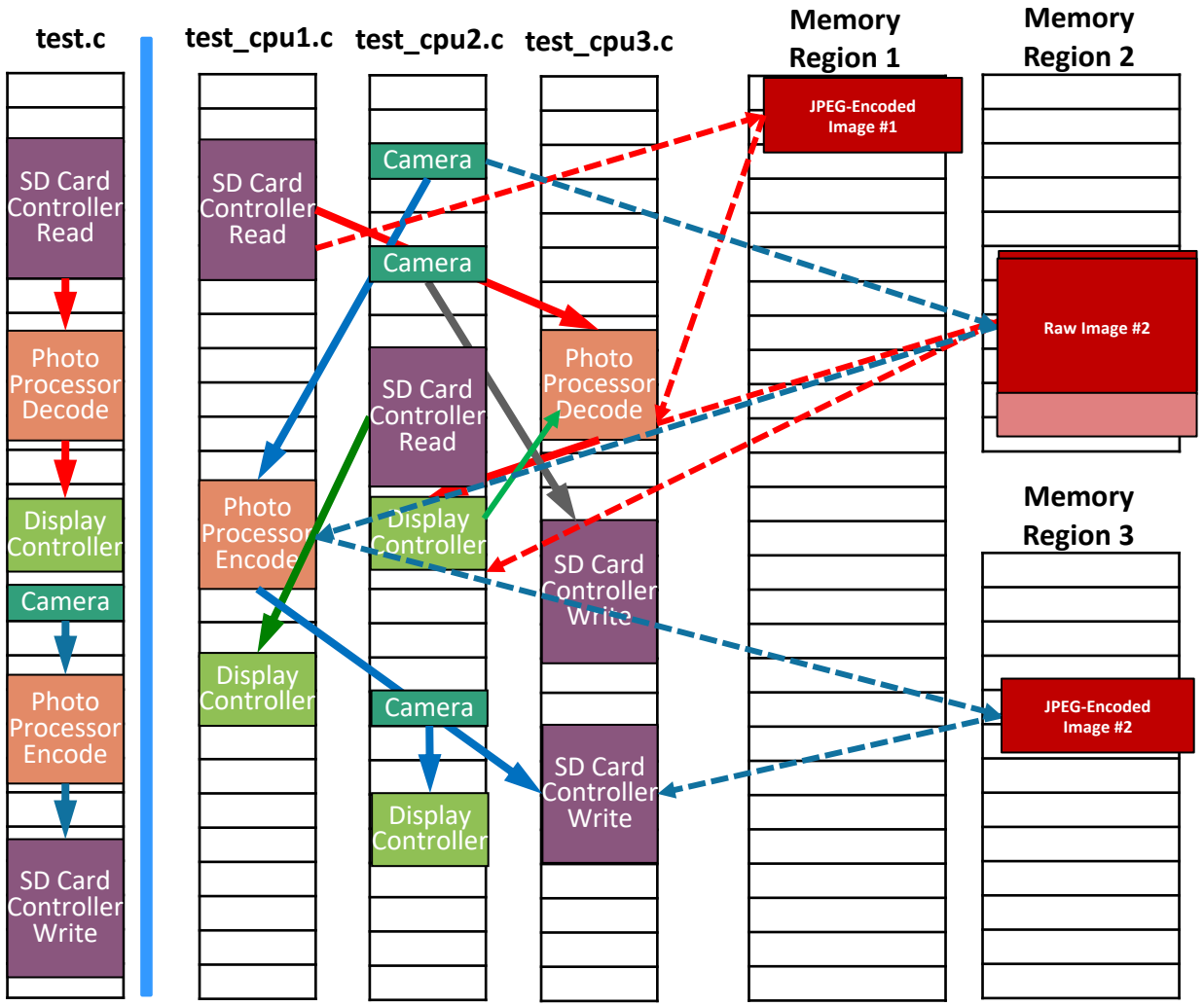


© Breker Verification Systems, Inc. All rights reserved.

Concurrent Test Execution







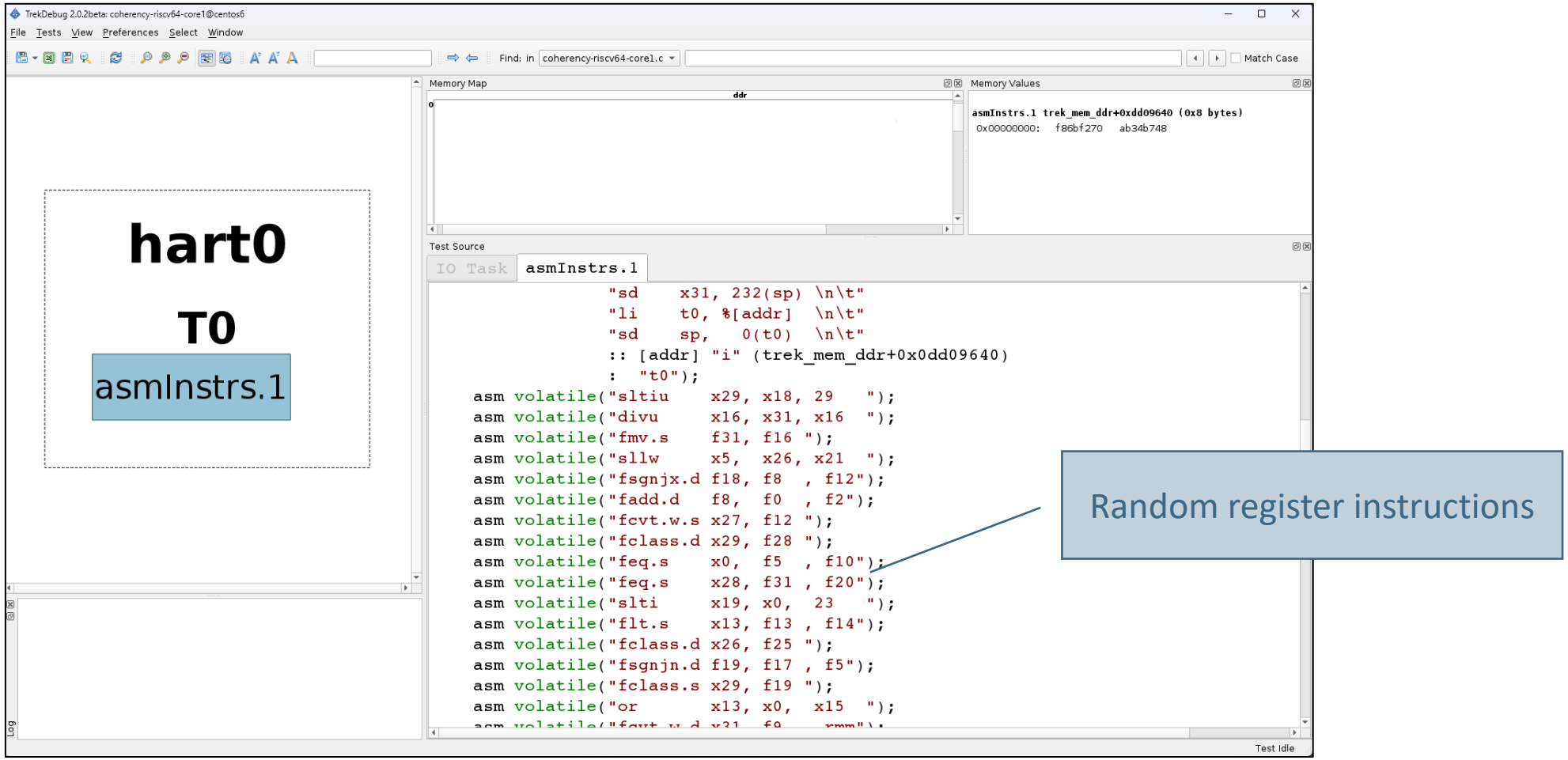
© Breker Verification Systems, Inc. All rights reserved.

Core-Integrity Challenges

Random Instructions	Do instructions yield correct results
Register/Register Hazards	Pipeline perturbations dues to register conflicts
Load/Store Integrity	Memory conflict patterns
Conditionals and Branches	Pipeline perturbations from synchronous PC change
Exceptions	Jumping to and returning from ISR
Asynchronous Interrupts	Pipeline perturbations from asynchronous PC change
Privilege Level Switching	Context switching
Core Security	Register and Memory protection by privilege level
Core Paging/MMU	Memory virtualization and TLB operation
Sleep/Wakeup	State retention across WFI
Voltage/Freq Scaling	Operation at different clock ratios
Core Coherency	Caches, evictions and snoops

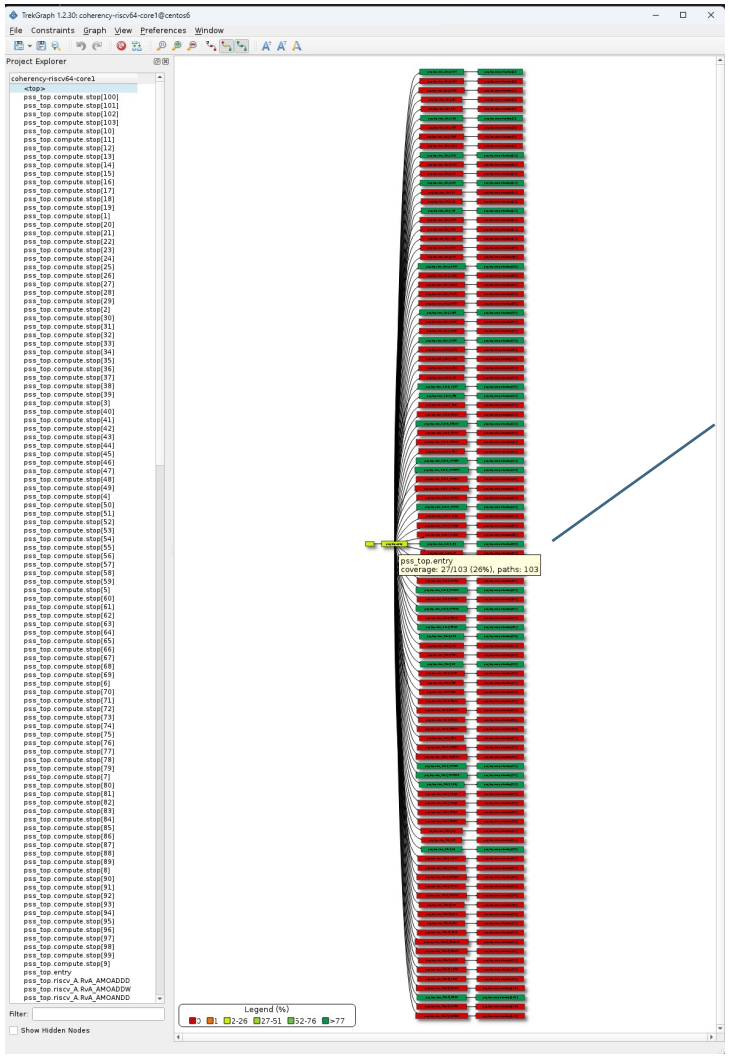
© Breker Verification Systems, Inc. All rights reserved.

RV64 Core Instruction Generation



© Breker Verification Systems, Inc. All rights reserved.

Instruction Coverage Analysis

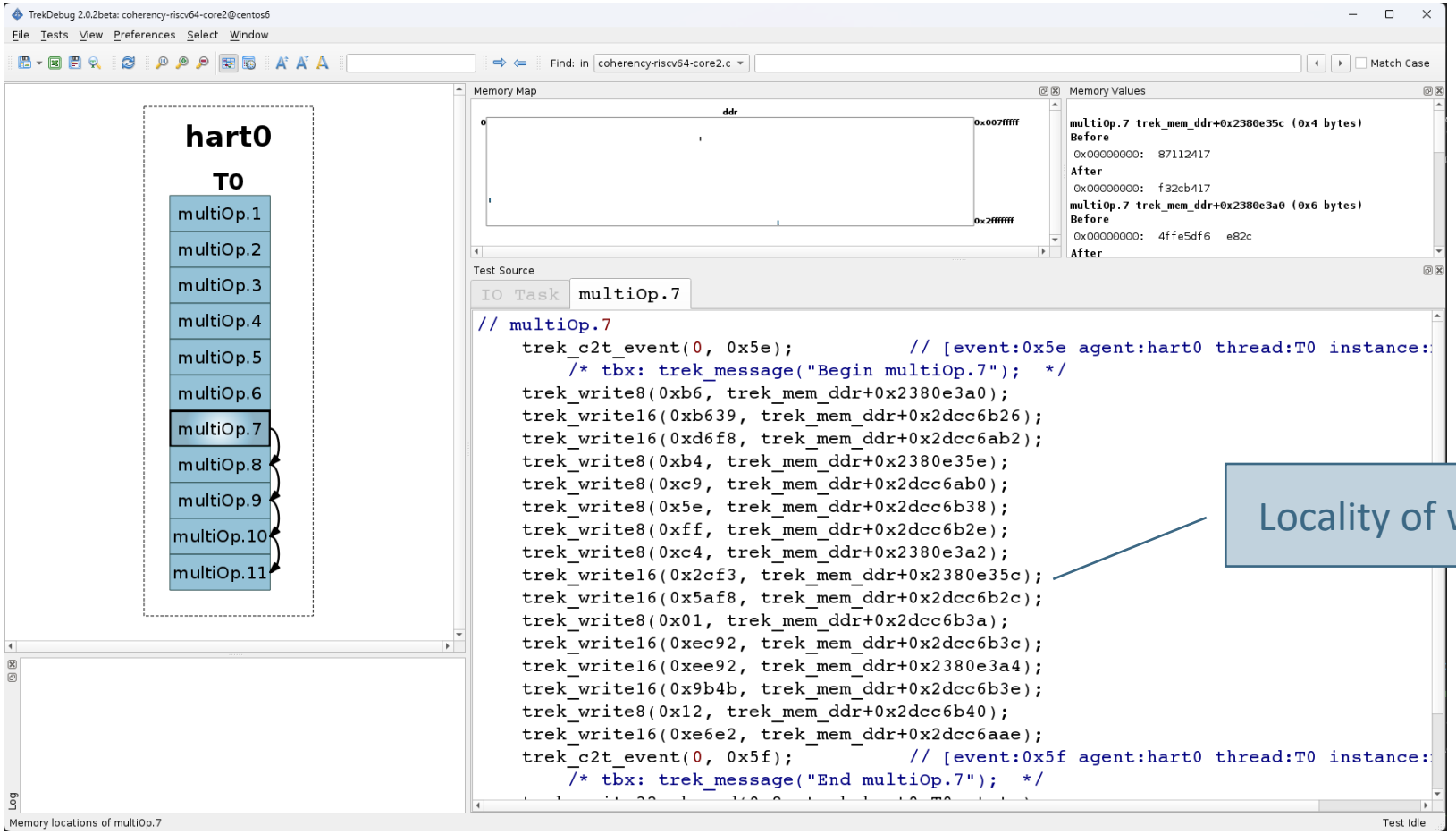


27/103 reachable opcode have been exercised

Atomics, loads and stores not reachable in register only test

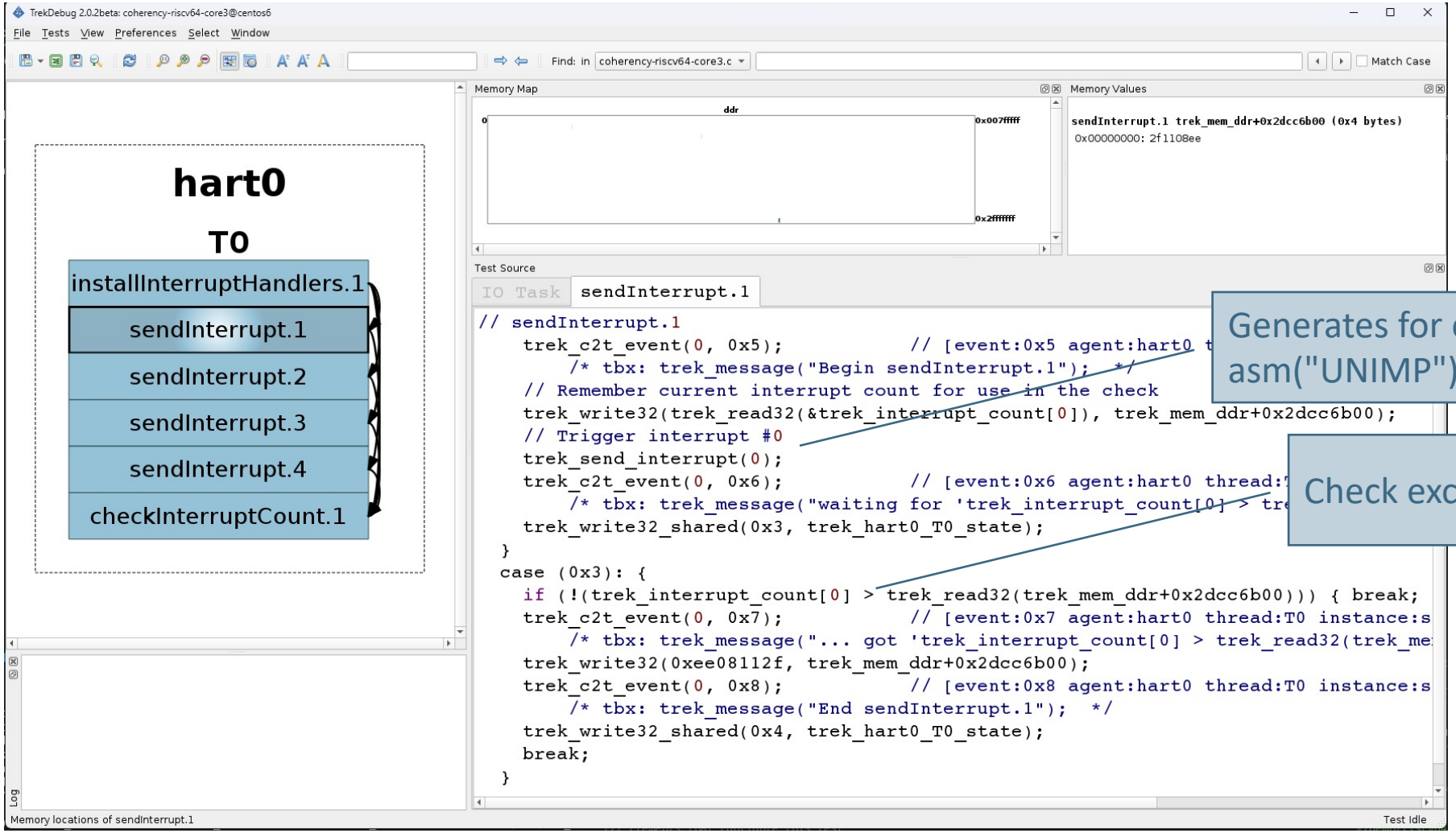
© Breker Verification Systems, Inc. All rights reserved.

RV64 Core Load/Store



© Breker Verification Systems, Inc. All rights reserved.

RV64 Core Exception Testing

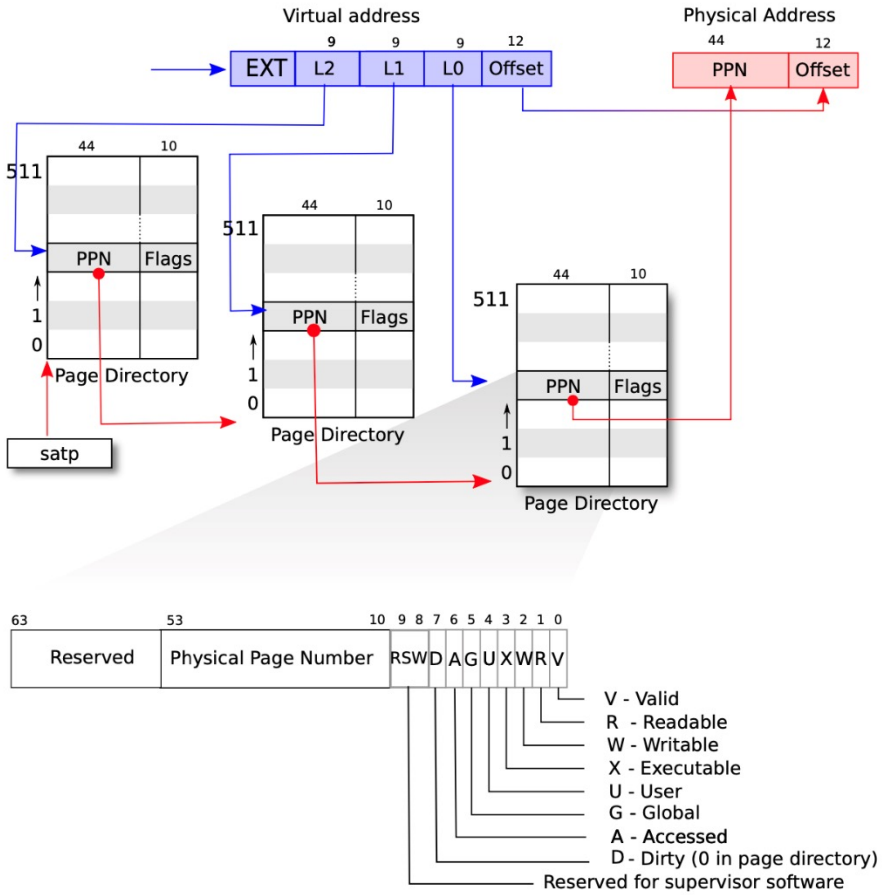


Generates for example, asm("UNIMP");

Check exception counts

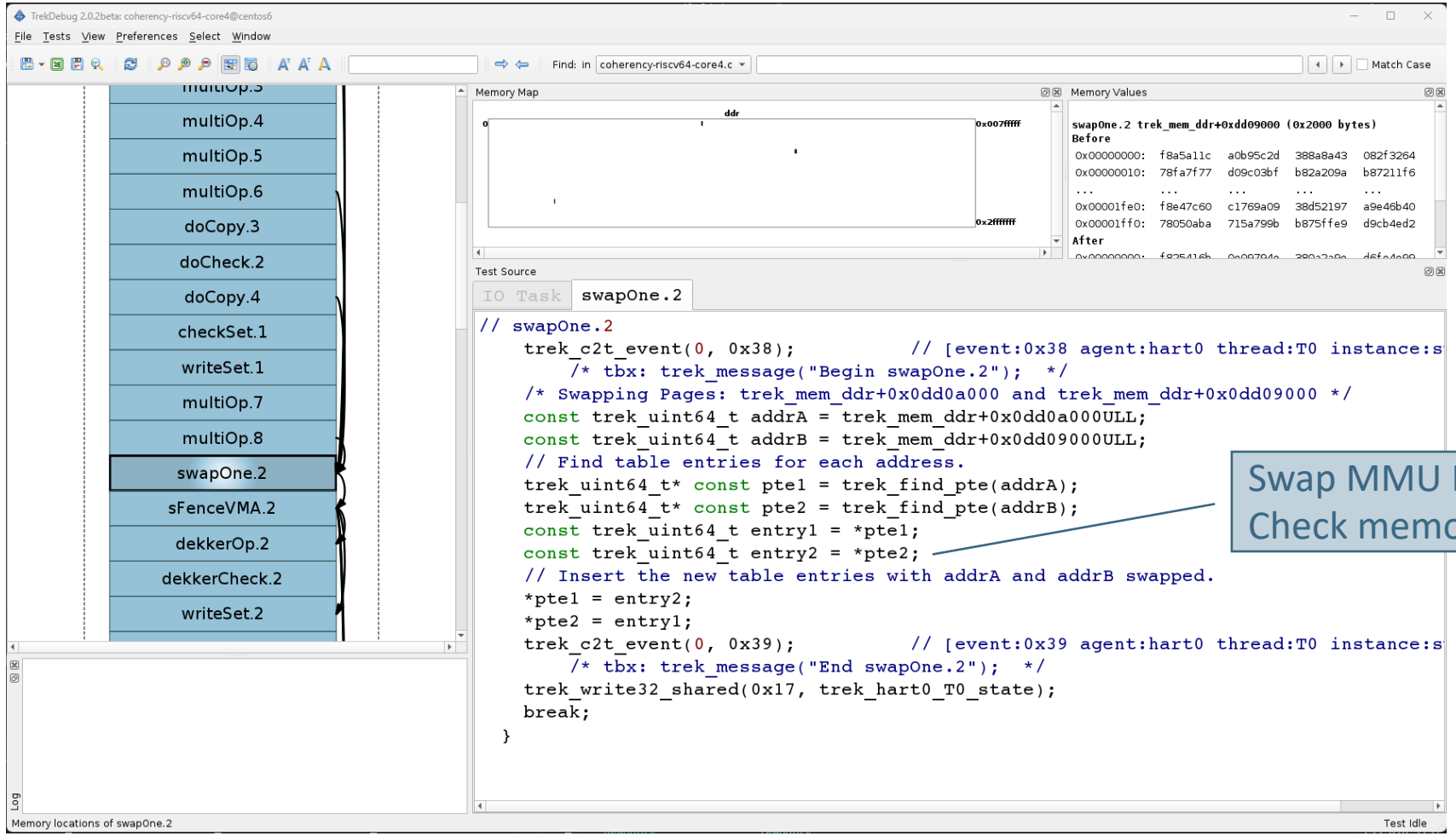
© Breker Verification Systems, Inc. All rights reserved.

Page Based Virtual Memory Tests



© Breker Verification Systems, Inc. All rights reserved.

RV64 Core Page Based MMU Tests

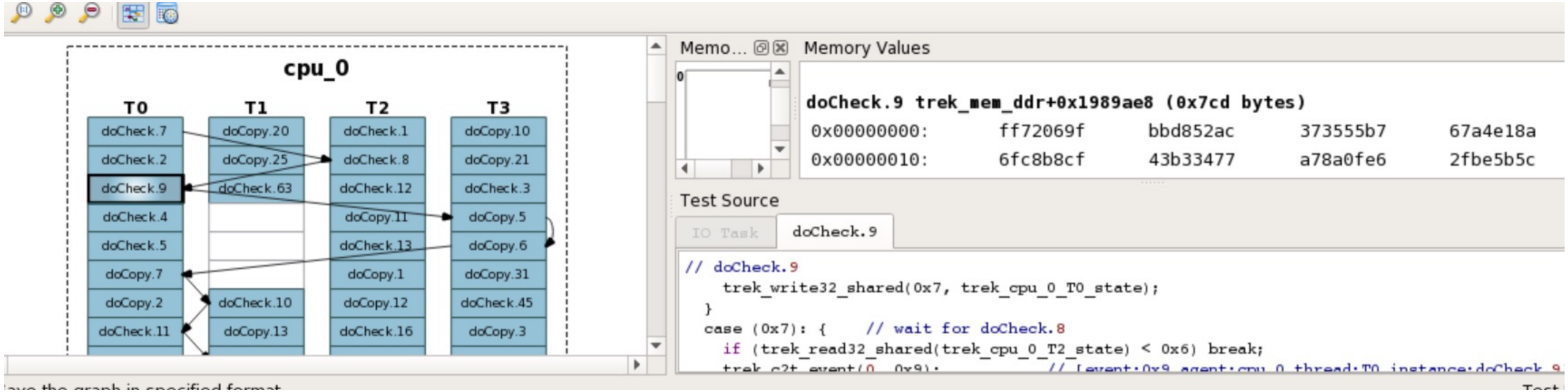


Swap MMU PTE's and Check memory access

© Breker Verification Systems, Inc. All rights reserved.

Core-Integrity: Single Core, 4 Threads

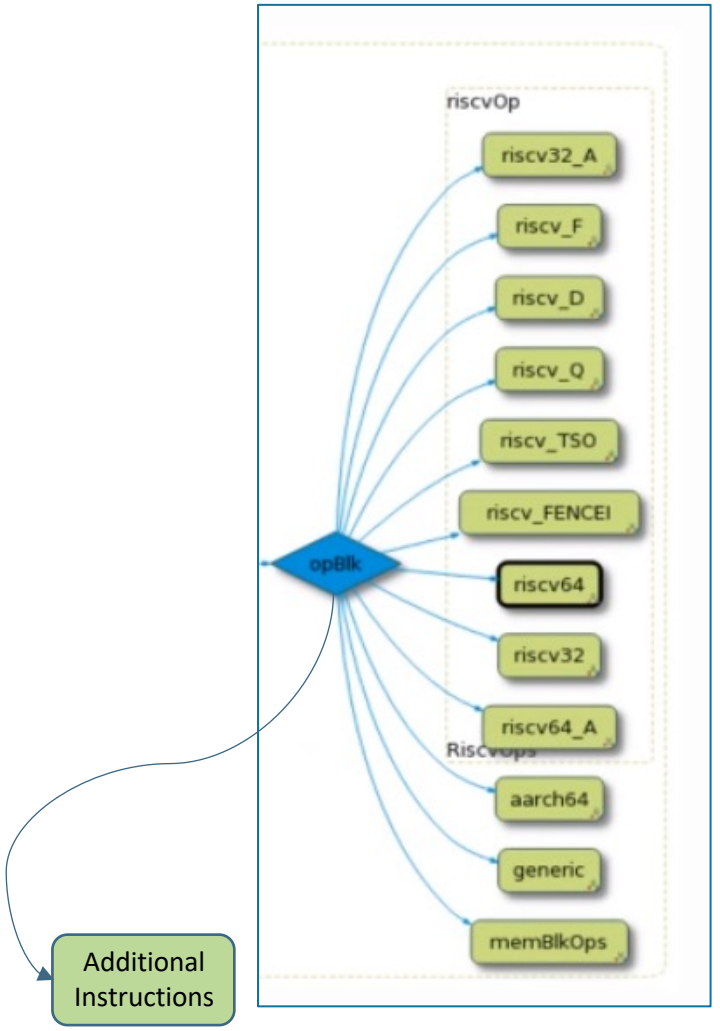
Tests utilizes processor's available resources/software threads



© Breker Verification Systems, Inc. All rights reserved.

Testing a Custom Instruction

- RISC-V ISA custom instructions pose a particularly difficult verification challenge
- Custom instructions need to be tested with the processor tests, not as an afterthought
- Breker solution allows custom instruction tests to be easily added into test graph
- Breker synthesis combines these tests with the app to ensure full custom processor testing



© Breker Verification Systems, Inc. All rights reserved.

SoC-Integrity Challenges

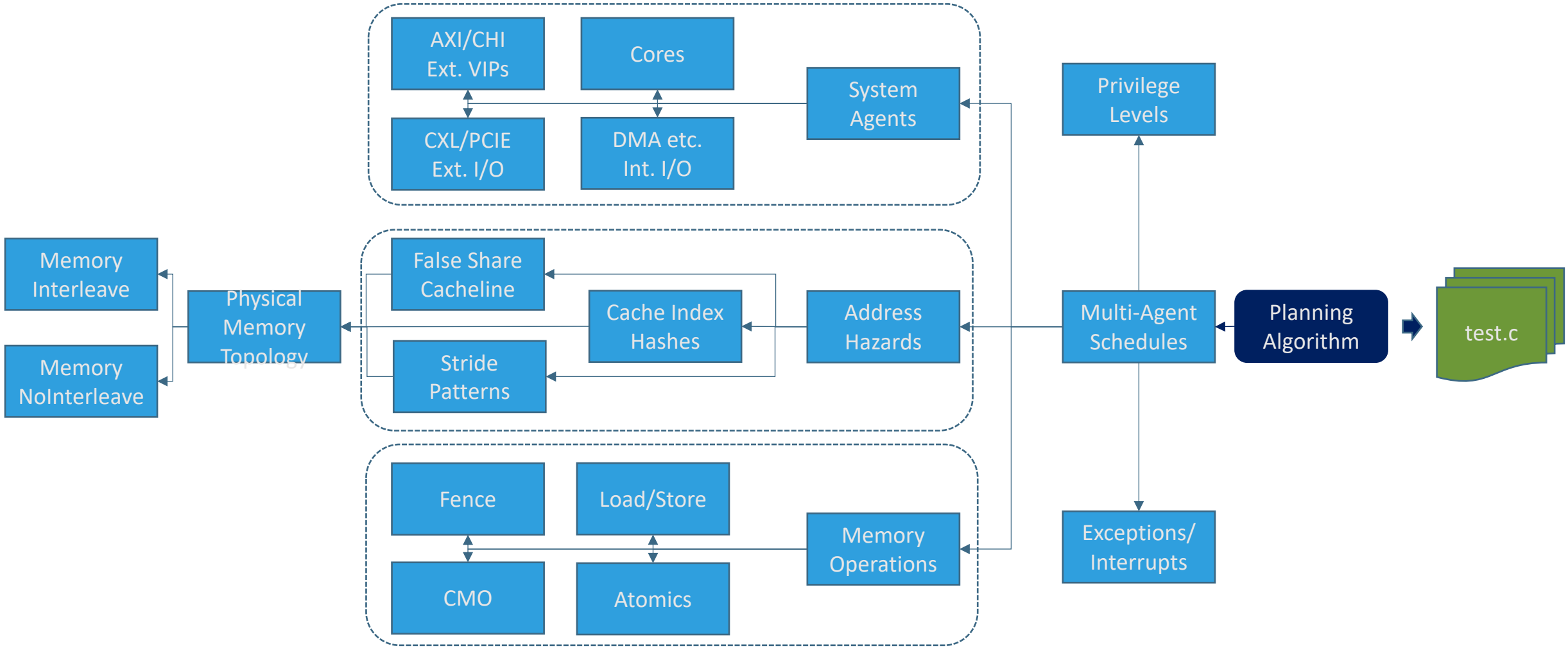
Random Memory Tests	Test Cores/Fabrics/Memory controllers across DDR, OCRAM, FLASH etc
Random Register Tests	Read/write test to all uncore registers
System Interrupts	Randomized interrupts through CLINT
Multi-core execution	Concurrent operations on fabric and memory
Memory ordering	For weakly order memory protocols
Atomic operation	Across all memory types
System Coherency	Cover all cache transitions, evictions, snoops
System Paging/IOMMU	System memory virtualization
System Security	Register and Memory protection across system
Power Management	System wide sleep/wakeup and voltage/freq scaling

Breker
RISC-V SoC-Integrity
SystemVIP

- **End-to-End use cases**
- **Early Firmware Testing**
- **Performance-Power Profiling**

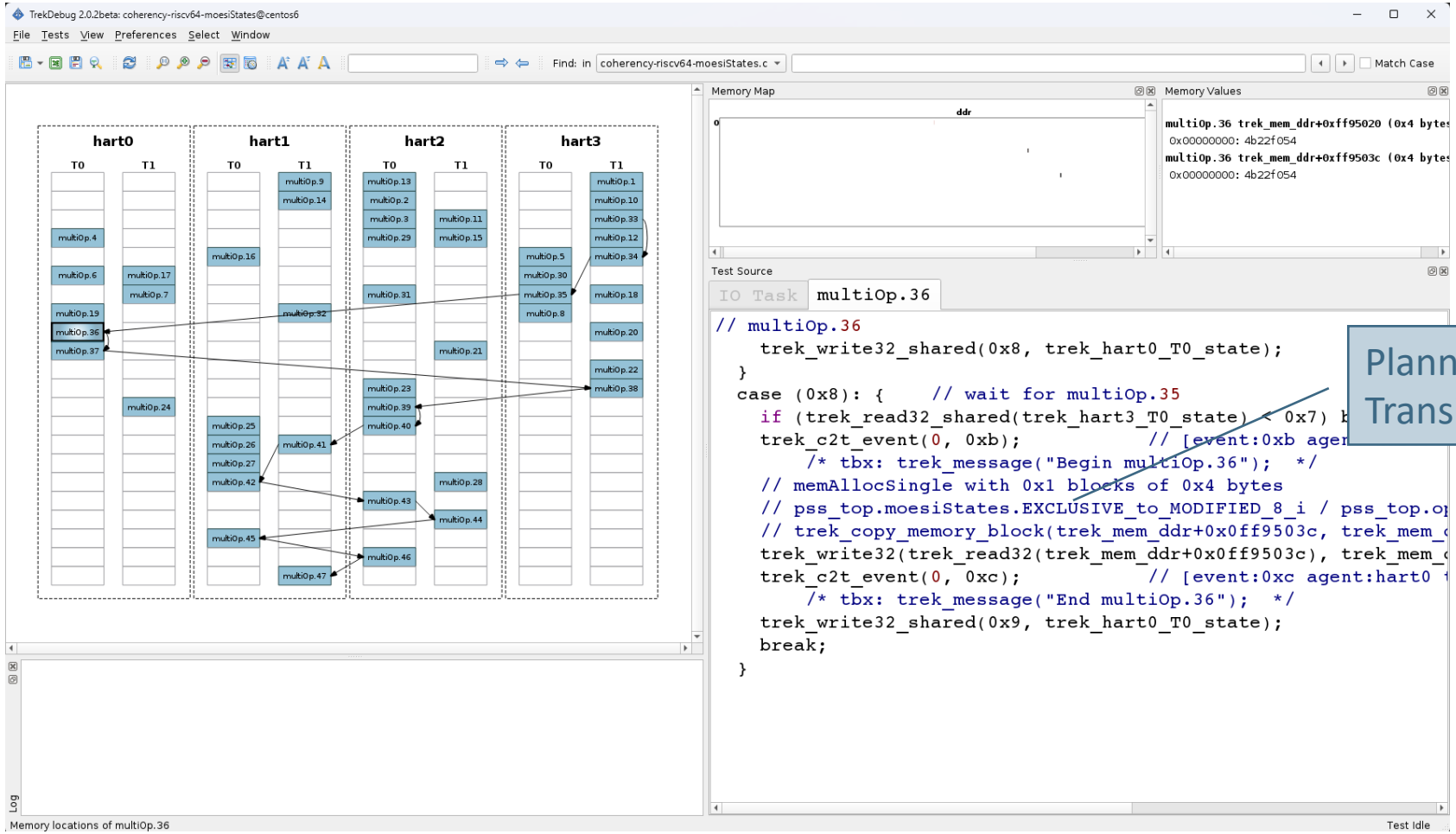
© Breker Verification Systems, Inc. All rights reserved.

RISC-V SoC Integrity TrekApp



© Breker Verification Systems, Inc. All rights reserved.

RV64 MultiCore MoesiStates

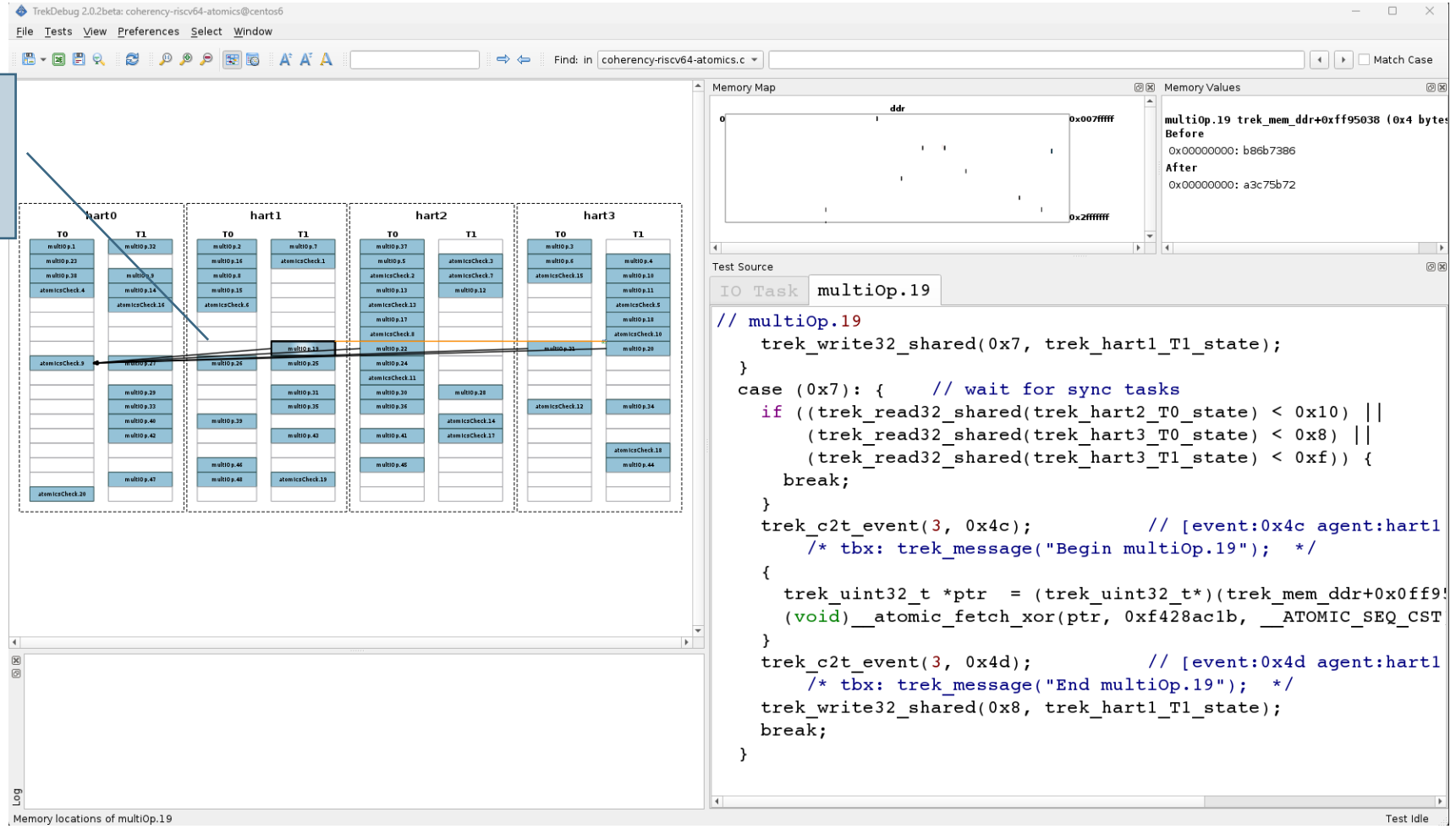


Planned Cache State Transitions

© Breker Verification Systems, Inc. All rights reserved.

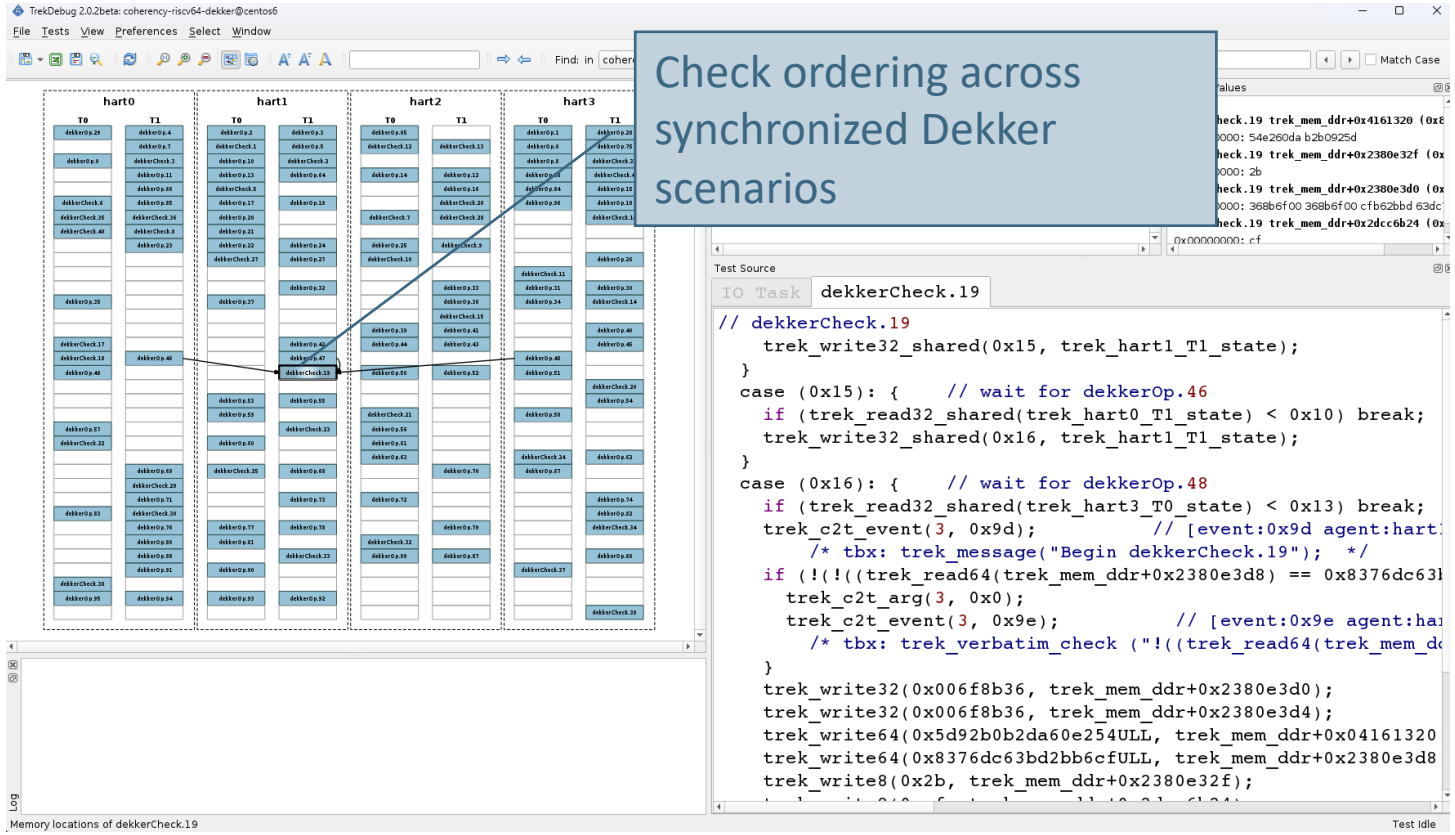
Atomics Testing

Check result is aggregate of synchronized atomic operations



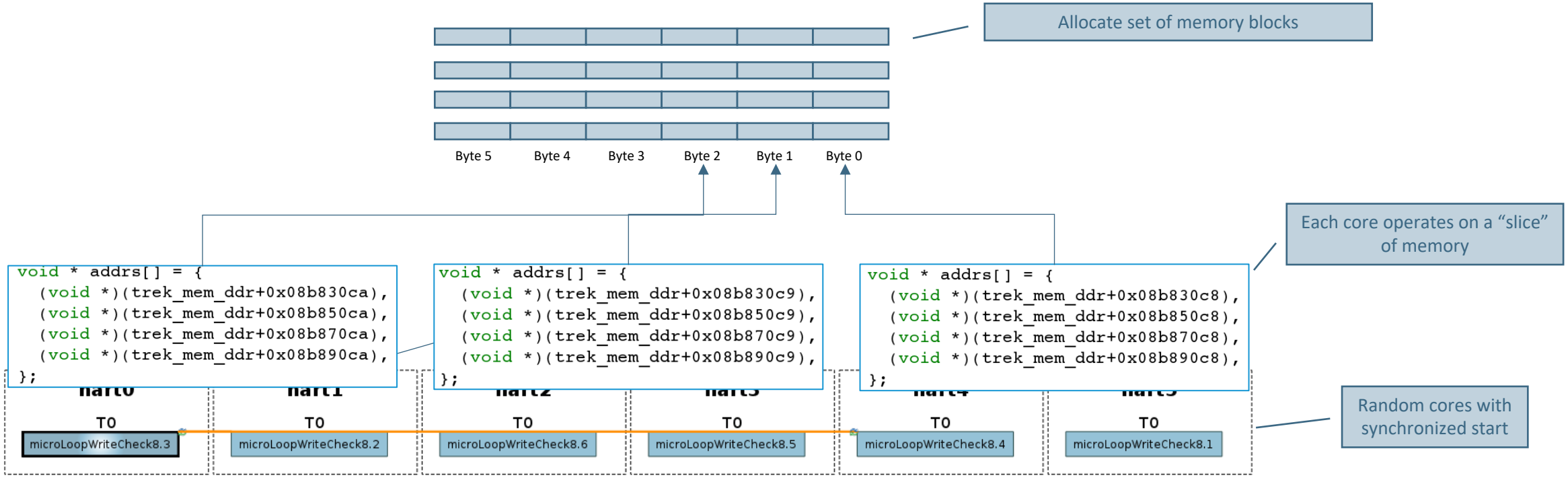
© Breker Verification Systems, Inc. All rights reserved.

Dekker Memory Ordering



© Breker Verification Systems, Inc. All rights reserved.

False-Share Memory Stress Tests




```
int trek_microloop_write_check8( void * addrs[], int count, trek_uint8_t pattern){
  int errorCount = 0;
  int ii;
  for ( ii = 0; ii < count; ++ii){
    trek_write8(pattern, addrs[ii]);
  }
  for ( ii = 0; ii < count; ++ii){
    if (trek_read8(addrs[ii]) != pattern) {
      ++errorCount;
      trek_runtime_error("trek_microloop_write_check8", addrs[ii], pattern, trek_read8(addrs[ii]));
    };
  }
  return errorCount;
}
```

```
for ( int ii = 0; ii < 1000; ++ii ){
  errorCount += trek_microloop_write_check8(addrs, 4, 184);
}
```

Each core has free running loop

© Breker Verification Systems, Inc. All rights reserved.

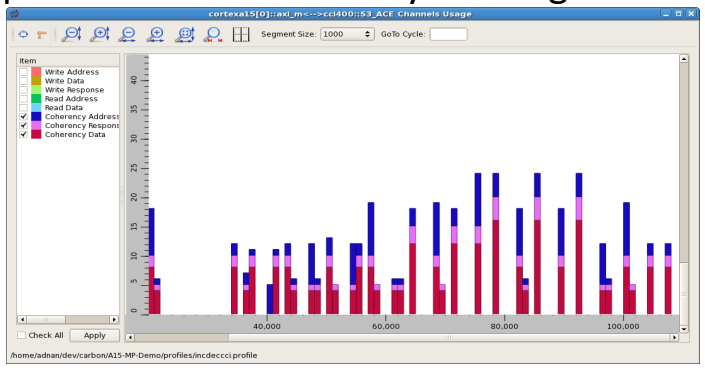
High Coverage and Bug Hunting

Recent examples of bugs discovered in real designs

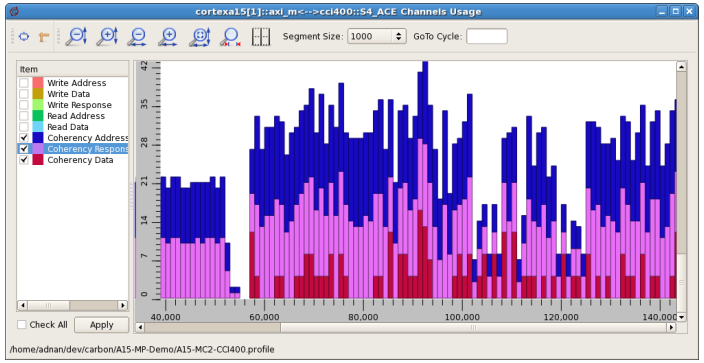
- RISC-V spec misunderstanding between core vendor and user
- Coherent Mesh Network (CMN) programming issues
- Misconfigured ARM CMN pin to enable coherent traffic
- DDR model unable to handle AXI "wrap" transactions.
- Common cache line access reveals deadlock
- Custom instruction bugs discovered by stress tests
- Results mismatch with ultrawide address strides
- Incorrect exception for guest virtual address[63:38] = 0x1ffffff
- Bad mcause value for guest physical address[63:31] != 0x0

SystemVIP Test Suite Synthesis Coverage Comparison

Typical directed coherency coverage



… vs. Breker automated coherency tests



© Breker Verification Systems, Inc. All rights reserved.

Thanks for Listening!
Any Questions?