2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION TAIWAN

Fast IP/SoC Test Generation with SystemVIPs and Test Suite Synthesis

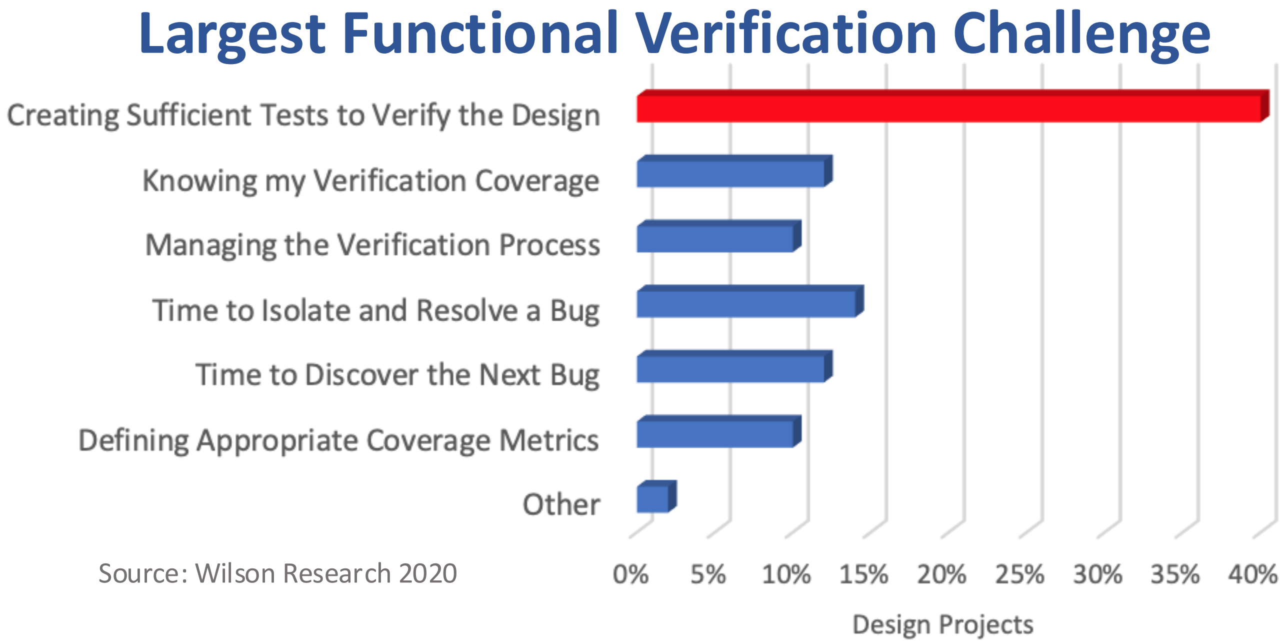
Andy Stein

Breker Verification Systems

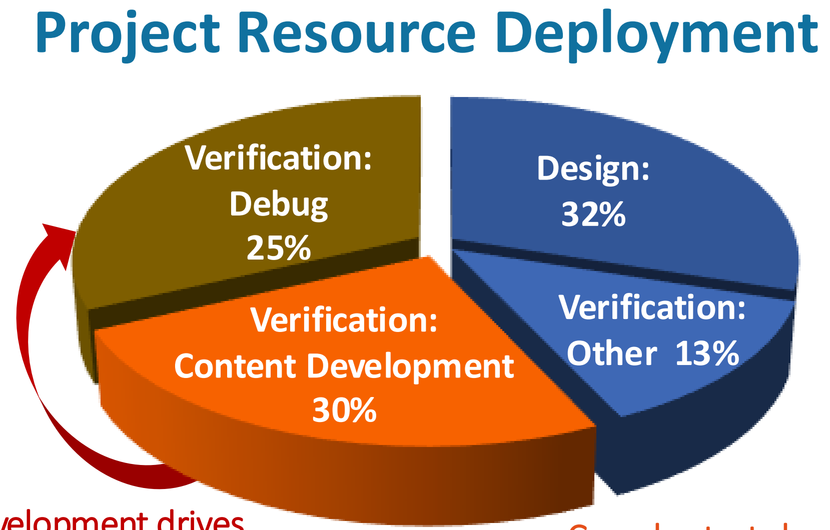
BREKER™

accellera SYSTEMS INITIATIVE

Explosive Challenge of Verification



Source: Wilson Research 2020

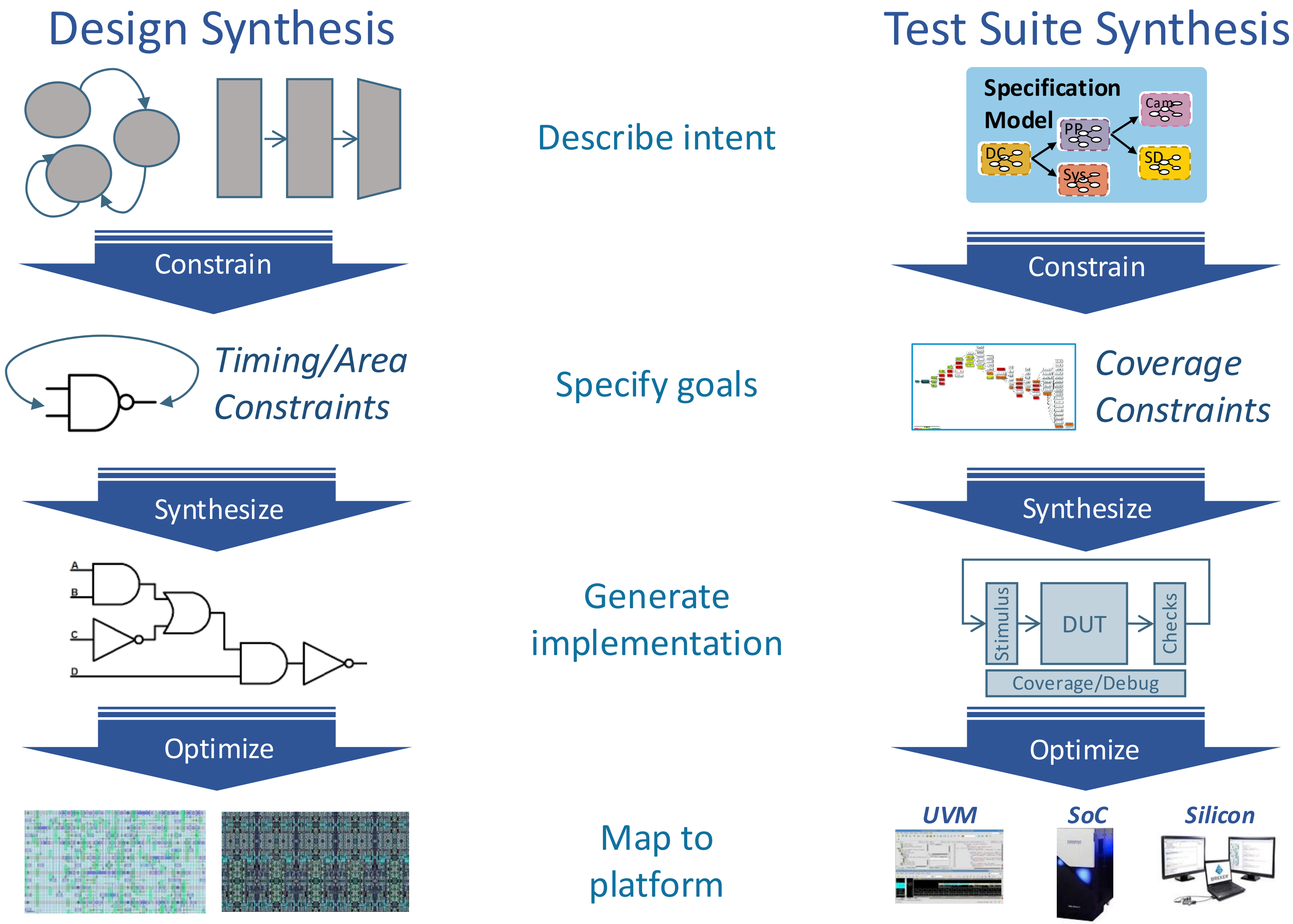


Test development drives debug

Complex tests hard to get right

Can we solve this issue through automation, abstraction & reuse?

Test Suite Synthesis… Analogous to Logic Synthesis



Design Synthesis

Test Suite Synthesis

Specification Model

Describe intent

Constrain

Constrain

Timing/Area Constraints

Specify goals

Coverage Constraints

Synthesize

Synthesize

Generate implementation

Optimize

Optimize

Map to platform

UVM

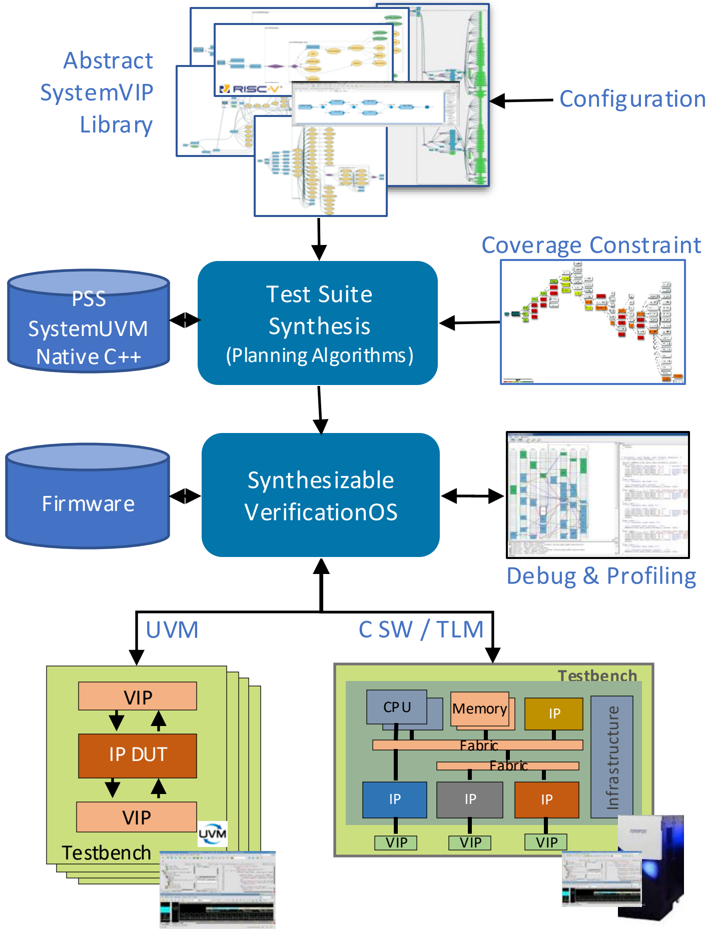
SoC

Silicon

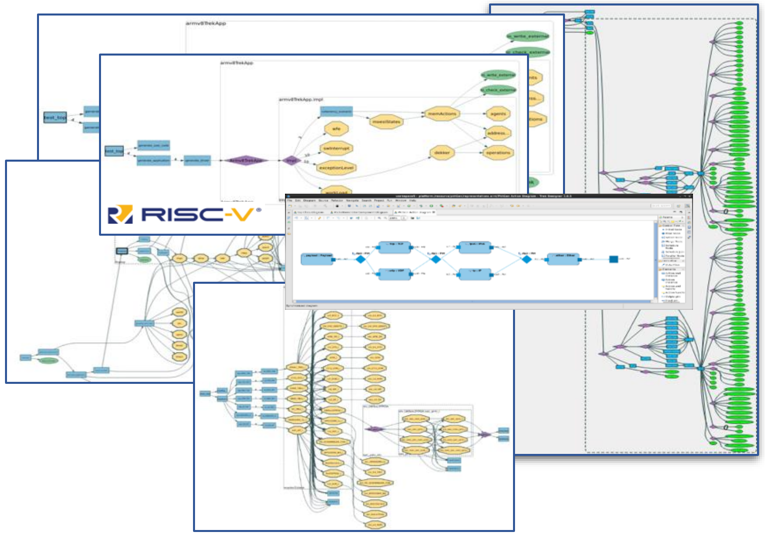
Breker Test Suite Synthesis

Amplifying SystemVIP Solutions

- SystemVIP pre-packaged, automated, self-checking scenario verification with broad test content range
- Test Suite Synthesis drives high-coverage, corner-case bug hunting, concurrent torture testing, and cross-coverage test combinations
- Fully portable through verification process, easily extendable and configurable integration, includes debug and coverage analysis



Breker SystemVIP Library & Solutions



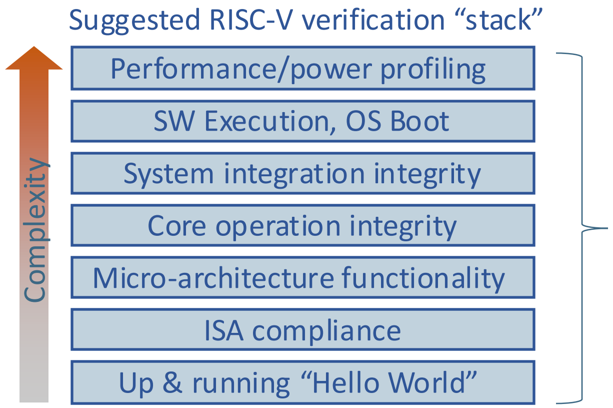
Leading SystemVIP Library Components

- *Cache-System Coherency* verifies cache and system-level coherency in a multiprocessor SoC
- *Arm SoCReady* testing for typical bugs across Arm SoCs, particularly given latest Arm architectures
- *RISC-V SoCReady* Ensuring RISC-V SoCs are compliant and operate correctly with cores from varies sources
- *RISC-V CoreAssurance* provides fast, pre-packaged tests for RISC-V Core and SoC integrity issues

Other SystemVIP-Based Automated Solutions

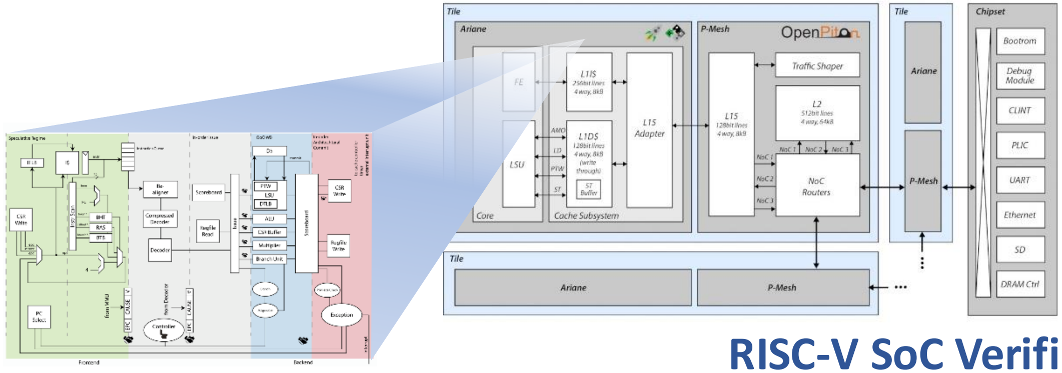
- *Early Firmware Validation* testing firmware in parallel to IP & Core
- *Automotive ISO 26262 Safety* for Systematic, Random and STL test sets
- *System End-to-End Testing* SystemUVM for scaling PSS IP tests to sub-systems and SoC
- The *Power Management* automates power domain switching verification
- The *Security* automates testing of hardware access rules for HRoT fabrics
- The *Networking & Interface* automates packet generation, CXL, UCIe interface tests

RISC-V SoC Integrity Verification



Suggested RISC-V verification "stack"

- Performance/power profiling
- SW Execution, OS Boot
- System integration integrity
- Core operation integrity
- Micro-architecture functionality
- ISA compliance
- Up & running "Hello World"

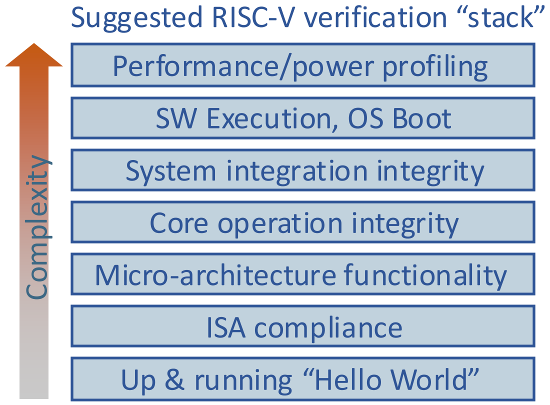


RISC-V SoC Verification Functionality

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
Packet Generation	Generating networking packets for I/O testing
Interface Testing	Analyzing coherent interfaces including CXL & UCIe
SoC Profiling	Layering concurrent tests to check operation under stress
Firmware-First	Executing SW on block or sub-system without processor

RISC-V Core Verification Functionality

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


Suggested RISC-V verification "stack"
Performance/power profiling
SW Execution, OS Boot
System integration integrity
Core operation integrity
Micro-architecture functionality
ISA compliance
Up & running "Hello World"
Complexity

Arm SoC Integrity Verification

"Latest Arm Cortex Processors and v9 architecture has introduced a number of complex issues that require extended SoC level verification."
Leading Arm/Breker customer

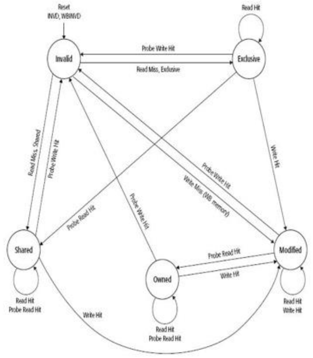
arm

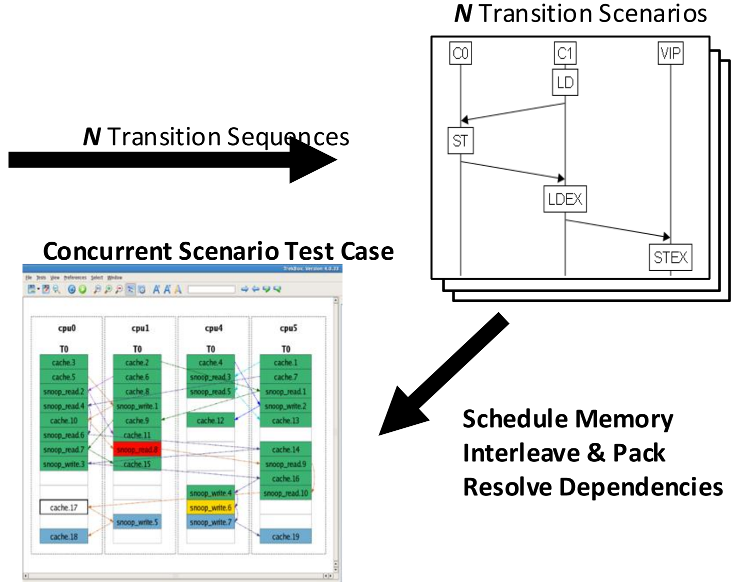
Arm SoC Integrity Functionality

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
Packet Generation	Generating networking packets for I/O testing
Interface Testing	Analyzing coherent interfaces including CXL & UCIe
SoC Profiling	Layering concurrent tests to check operation under stress
Firmware-First	Executing SW on block or sub-system without processor

System Coherency: High Coverage and Bug Hunting

- Automated, configurable coherency tests for broad range of complex scenarios
- Proprietary and public algorithms to maximize coverage and corner case detection
- Operates from IP to system to post silicon, in use at leading semiconductor companies





N **Transition Scenarios**

N Transition Sequences

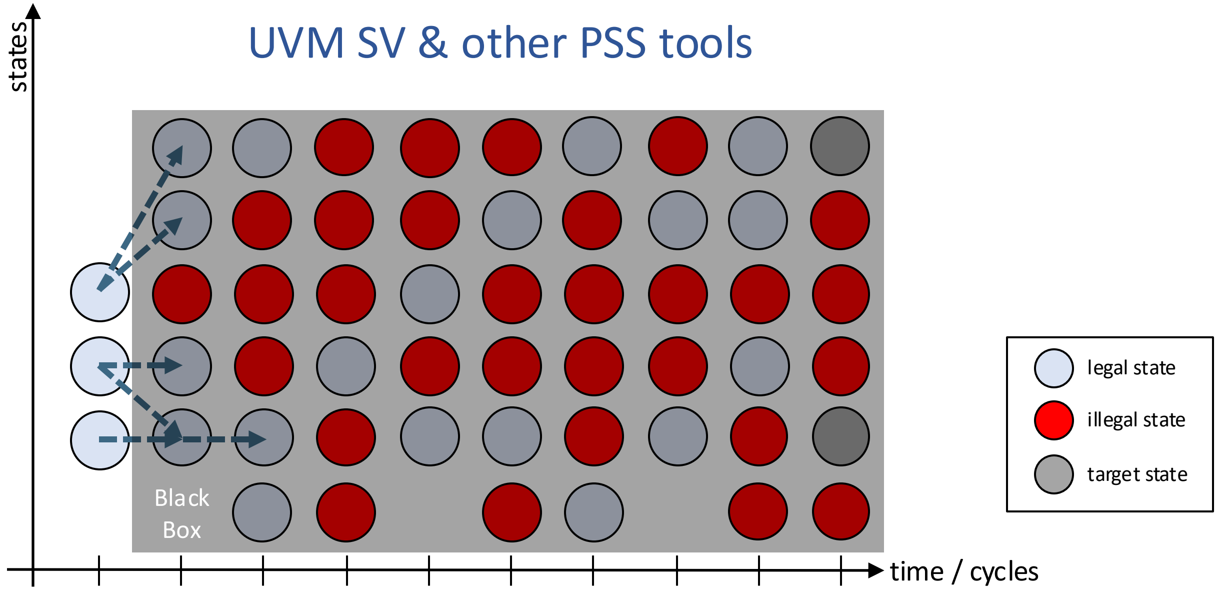
Concurrent Scenario Test Case

Schedule Memory
Interleave & Pack
Resolve Dependencies

- Cache State Transitions
- Snoop / Probe Sources
- Cache Line Sharing Cases
- Capacity Eviction Cases
- Load/Store Operation Size
- False Sharing Cases
- Crossing Cache Line Boundaries
- Memory Ordering Tests
- Concurrent Scenarios
- Multiple Memory Regions

Leveraging AI Planning Algorithms

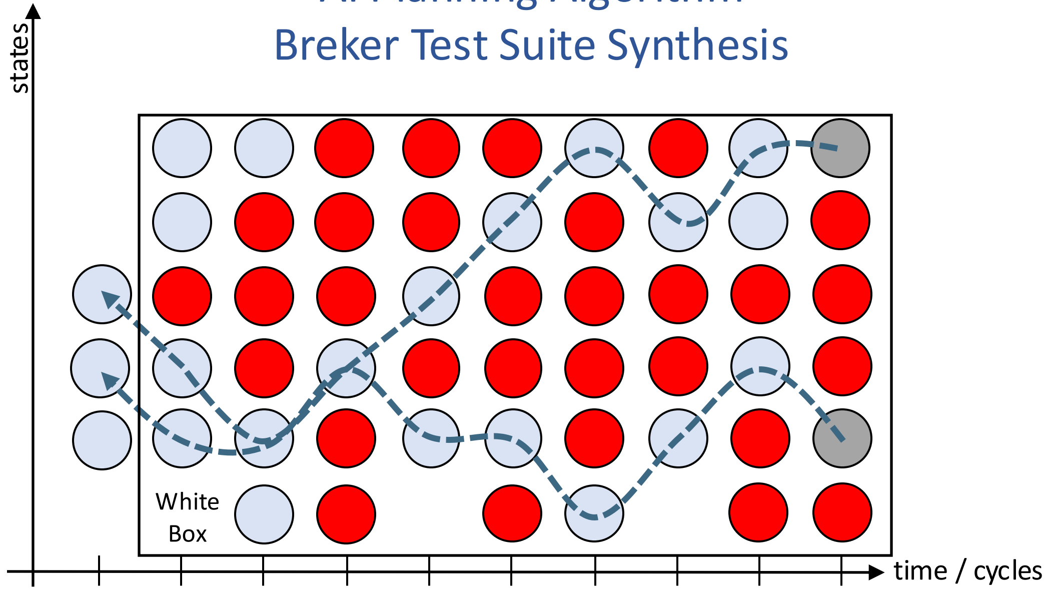
Constrained Random Generation
UVM SV & other PSS tools



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

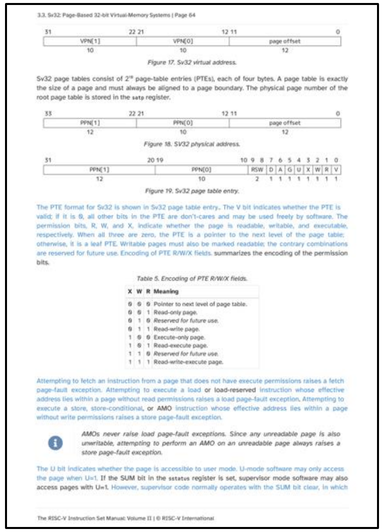
Deep sequential, optimized test discovers complex corner-cases

White Paper Discussing AI Planning Algorithm Test Generation on Breker Website

Manual Spec to Verification Model AI

- Can we leverage AI to read a manual specification and generate a verification plan and abstract graph model?
- New research program for Breker
- Starting to show positive results

TrekAssist Program
Spec to Plan to Graph AI



3.3. Sv32: Page-Based 32-bit Virtual-Memory Systems | Page 64

31	22 21	12 11	0
VPN[1]	VPN[0]	page offset	
10	10	12	

Figure 17. Sv32 virtual address.

Sv32 page tables consist of 2¹⁰ page-table entries (PTEs), each of four bytes. A page table is exactly the size of a page and must always be aligned to a page boundary. The physical page number of the root page table is stored in the satp register.

33	22 21	12 11	0
PPN[1]	PPN[0]	page offset	
12	10	12	

Figure 18. SV32 physical address.

31	20 19	10 9 8	7	6	5	4	3	2	1	0
PPN[1]	PPN[0]	RSW	D	A	G	U	X	W	R	V
12	10	2	1	1	1	1	1	1	1	1

Figure 19. Sv32 page table entry.

The PTE format for Sv32 is shown in Sv32 page table entry.. The V bit indicates whether the PTE is valid; if it is 0, all other bits in the PTE are don't-cares and may be used freely by software. The permission bits, R, W, and X, indicate whether the page is readable, writable, and executable, respectively. When all three are zero, the PTE is a pointer to the next level of the page table; otherwise, it is a leaf PTE. Writable pages must also be marked readable; the contrary combinations are reserved for future use. Encoding of PTE R/W/X fields. summarizes the encoding of the permission bits.

Table 5. Encoding of PTE R/W/X fields.

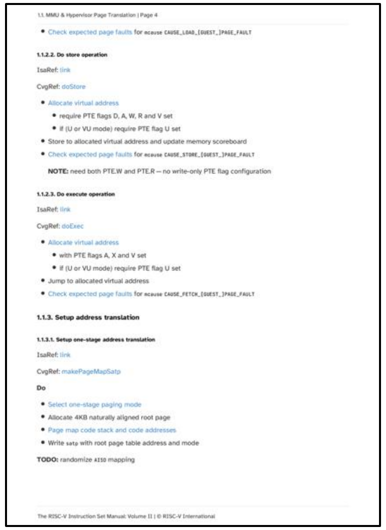
X	W	R	Meaning
0	0	0	Pointer to next level of page table.
0	0	1	Read-only page.
0	1	0	*Reserved for future use.*
0	1	1	Read-write page.
1	0	0	Execute-only page.
1	0	1	Read-execute page.
1	1	0	*Reserved for future use.*
1	1	1	Read-write-execute page.

Attempting to fetch an instruction from a page that does not have execute permissions raises a fetch page-fault exception. Attempting to execute a load or load-reserved instruction whose effective address lies within a page without read permissions raises a load page-fault exception. Attempting to execute a store, store-conditional, or AMO instruction whose effective address lies within a page without write permissions raises a store page-fault exception.

ℹ *AMOs never raise load page-fault exceptions. Since any unreadable page is also unwritable, attempting to perform an AMO on an unreadable page always raises a store page-fault exception.*

The U bit indicates whether the page is accessible to user mode. U-mode software may only access the page when U=1. If the SUM bit in the sstatus register is set, supervisor mode software may also access pages with U=1. However, supervisor code normally operates with the SUM bit clear, in which

The RISC-V Instruction Set Manual: Volume II | © RISC-V International



1.1. MMU & Hypervisor Page Translation | Page 4

- Check expected page faults for `mcause CAUSE_LOAD_[GUEST_]PAGE_FAULT`

1.1.2.2. Do store operation

IsaRef: link

CvgRef: doStore

- Allocate virtual address
  - require PTE flags D, A, W, R and V set
  - if (U or VU mode) require PTE flag U set
- Store to allocated virtual address and update memory scoreboard
- Check expected page faults for `mcause CAUSE_STORE_[GUEST_]PAGE_FAULT`

NOTE: need both PTE.W and PTE.R — no write-only PTE flag configuration

1.1.2.3. Do execute operation

IsaRef: link

CvgRef: doExec

- Allocate virtual address
  - with PTE flags A, X and V set
  - if (U or VU mode) require PTE flag U set
- Jump to allocated virtual address
- Check expected page faults for `mcause CAUSE_FETCH_[GUEST_]PAGE_FAULT`

1.1.3. Setup address translation

1.1.3.1. Setup one-stage address translation

IsaRef: link

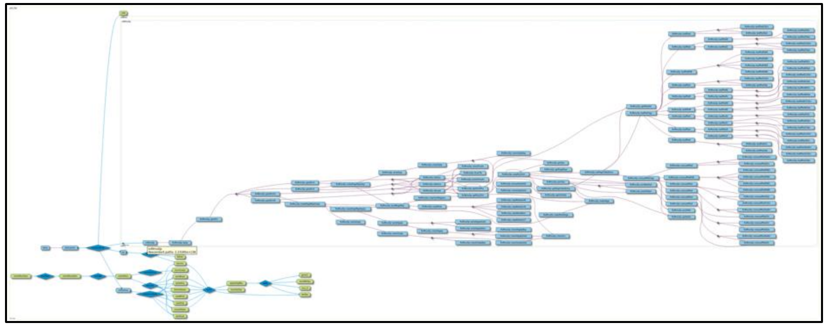
CvgRef: makePageMapSatp

Do

- Select one-stage paging mode
- Allocate 4KB naturally aligned root page
- Page map code stack and code addresses
- Write `satp` with root page table address and mode

TODO: randomize `ASID` mapping

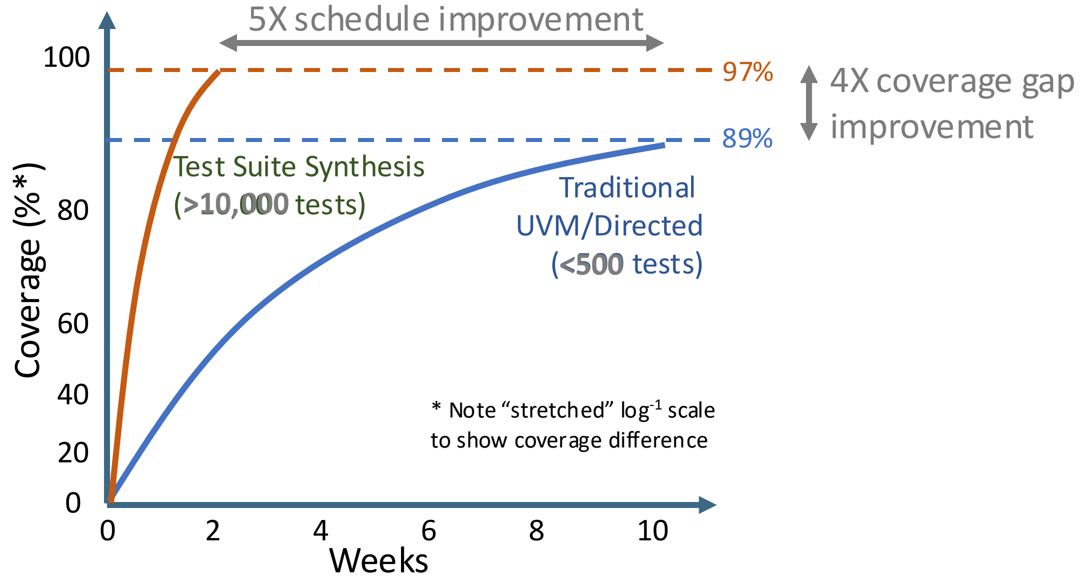
The RISC-V Instruction Set Manual: Volume II | © RISC-V International



Typical Results

BROADCOM

- Broadcom case study
- Complex cell phone SoC
- UVM/C tests augmented existing UVM testbench



* Note "stretched" log^{-1} scale to show coverage difference

Metric	Manual	Synthesis	Improvement
Test Authoring Time	2.5 months	2 weeks	5X
Unique, High-Impact Tests Generated	<500	>10,000	20X
Coverage Gap (100% - Coverage)	11%	3%	4X

Accelerated, High Coverage Test Content

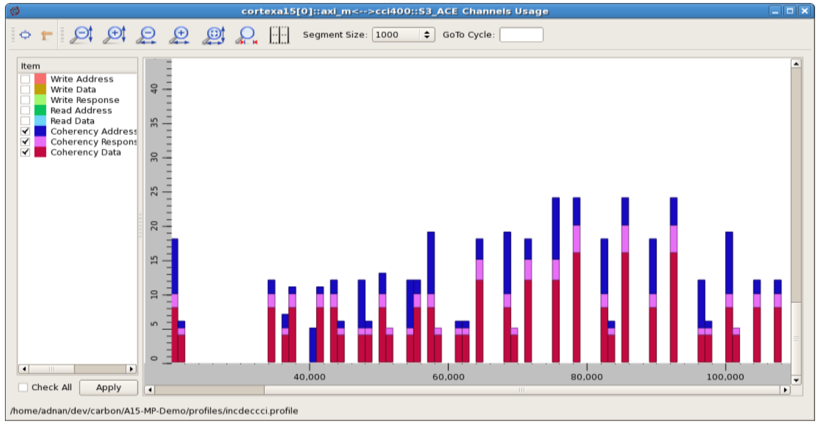
Breker SystemVIP: High Coverage and Bug Hunting

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
- 7 Results mismatch with ultrawide address strides
- 8 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



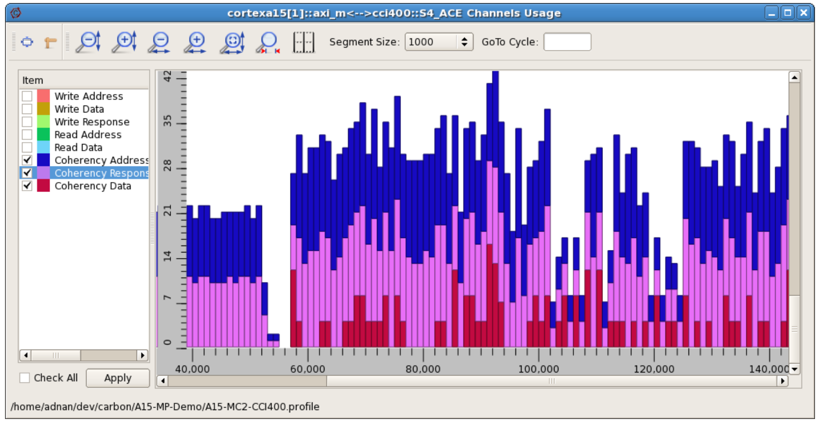
cortexa15[0]::axi_m<-->cci400::S3_ACE Channels Usage

Segment Size: 1000

GoTo Cycle:

/home/adnan/dev/carbon/A15-MP-Demo/profiles/incdeccci.profile

… vs. Breker automated coherency tests



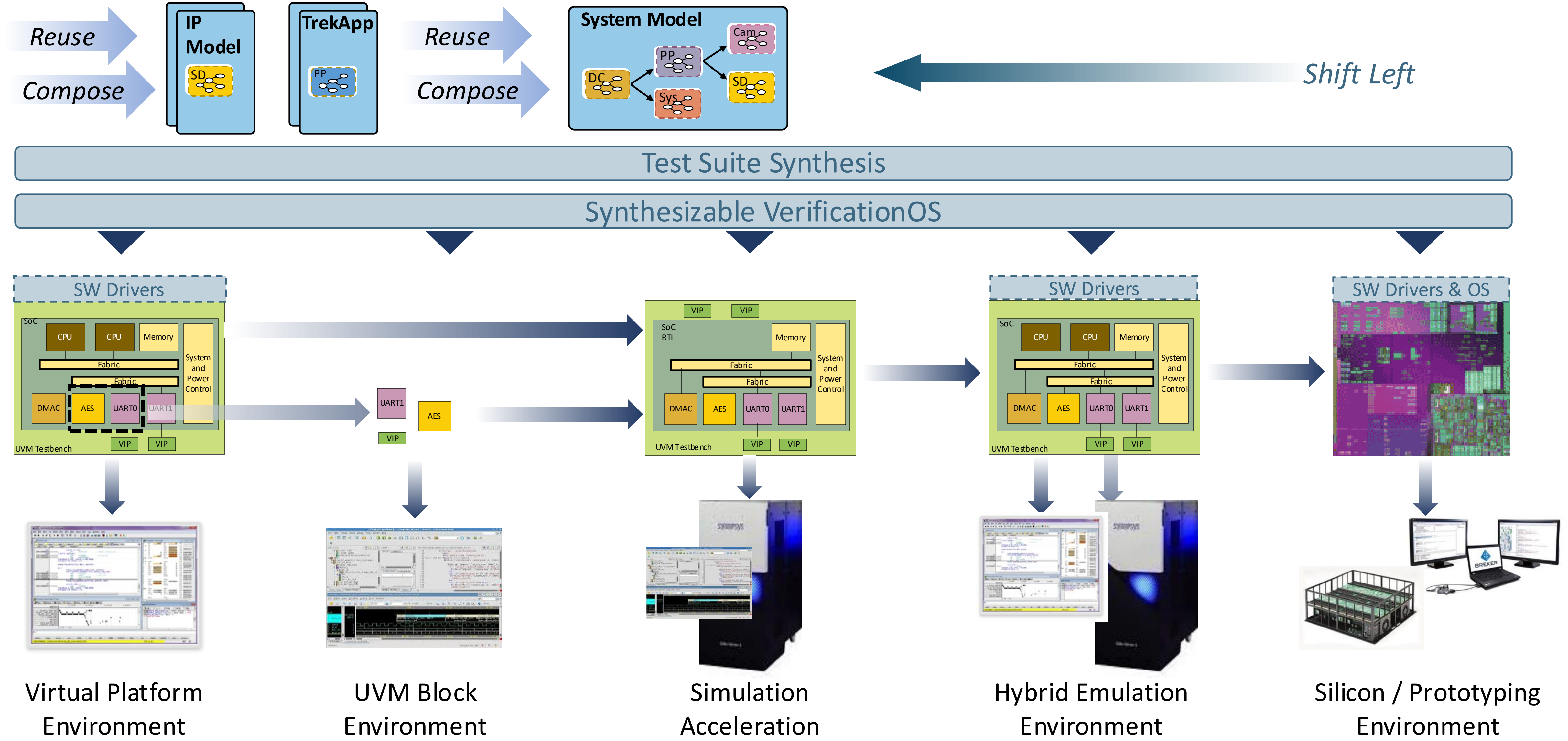
cortexa15[1]::axi_m<-->cci400::S4_ACE Channels Usage

Segment Size: 1000

GoTo Cycle:

/home/adnan/dev/carbon/A15-MP-Demo/A15-MC2-CCI400.profile

Test Reuse and Portability Driving Verification Efficiency



Thanks for Listening!
Any Questions?