2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Using Portable Stimulus to Verify Cache Coherency in a Many-Core SoC

Adnan Hamid, Breker Verification Systems, Inc.

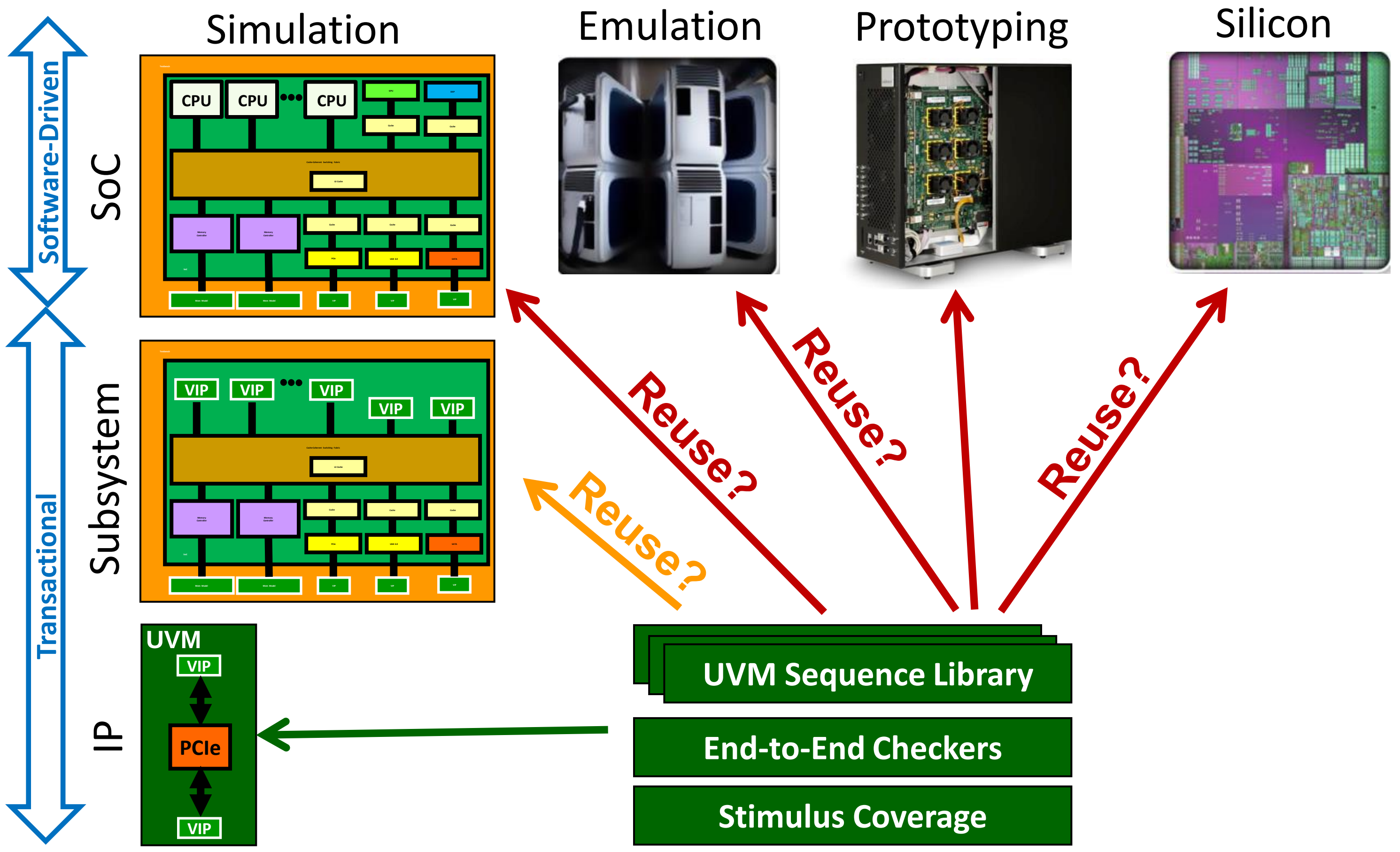
Raja Pantangi, Cavium, Inc.

BREKER™ CAVIUM

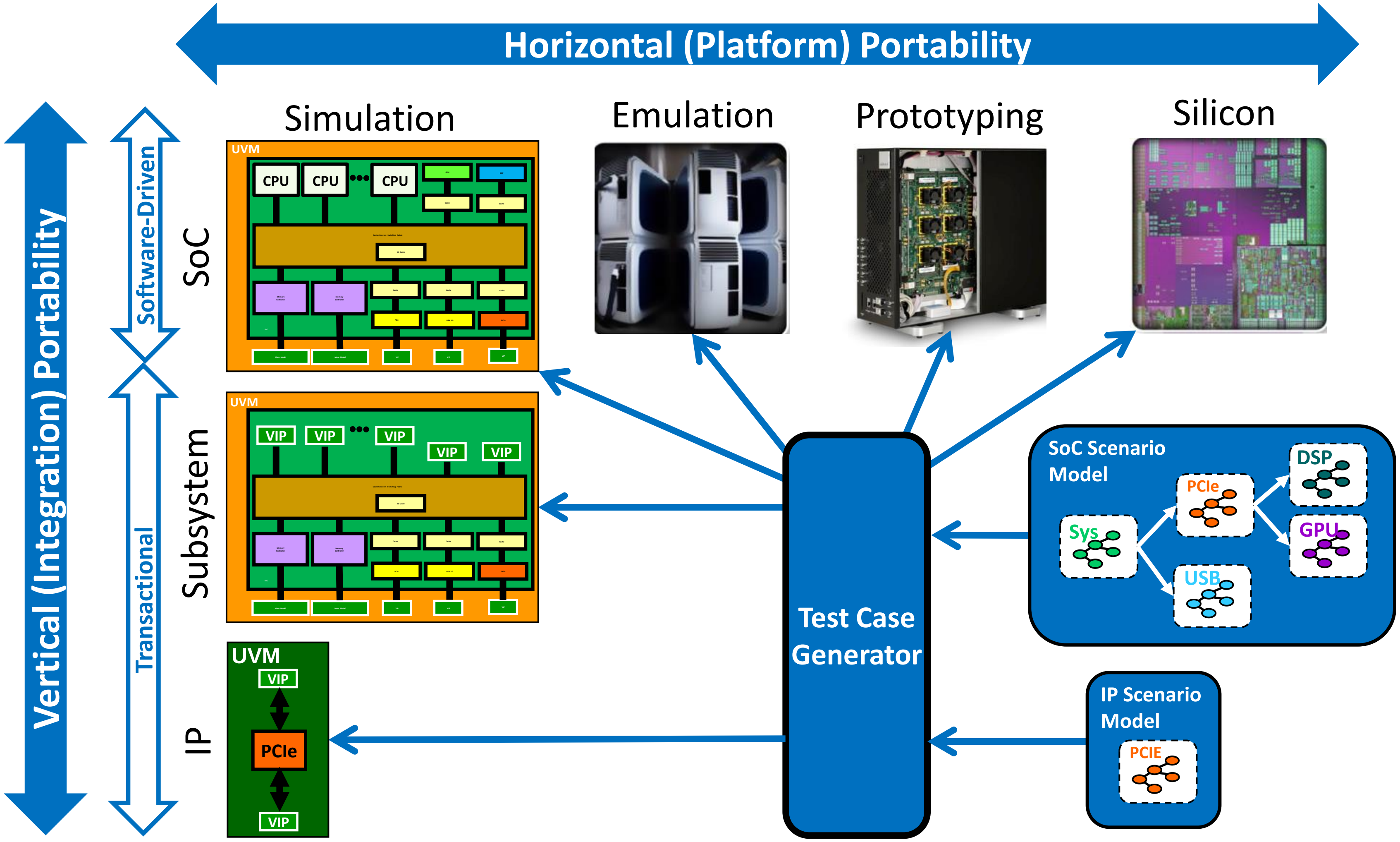
Agenda

- Why Portable Stimulus?
- Cache Coherency Verification Challenges
- Automated Cache Coherency Verification
- Test Case (Scenario Model) Portability
- Real-World Application at Cavium
- Cavium Project Results and Learnings
- Conclusions

Why Portable Stimulus ?



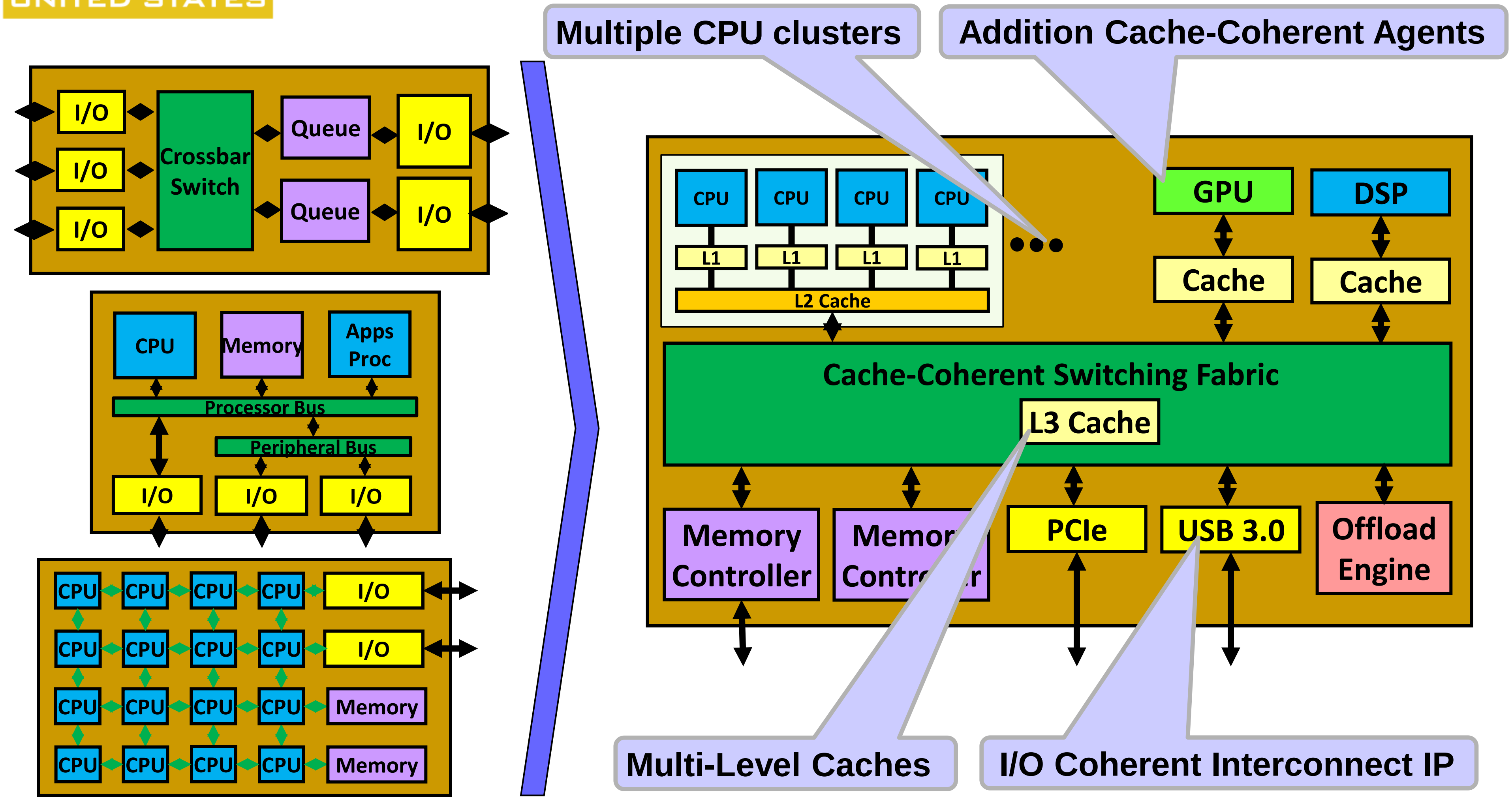
The Portable Stimulus Vision



Accellera & Portable Stimulus

- Industry has a clear goal for test portability
  - Accellera Portable Stimulus Working Group (PSWG)
  - "Vertical" portability from IP to system
  - "Horizontal" portability from simulation to silicon
- Upcoming standard will be the next step from UVM
  - Graph-based models capture verification intent
  - Graphs plug together directly for vertical reuse
  - Transactional tests generated for UVM testbench
  - Software-driven tests generated and run on SoC embedded processors on all verification platforms

SoC Cache Coherency



- Cache coherency is now an SoC-level challenge

Key Verification Challenges

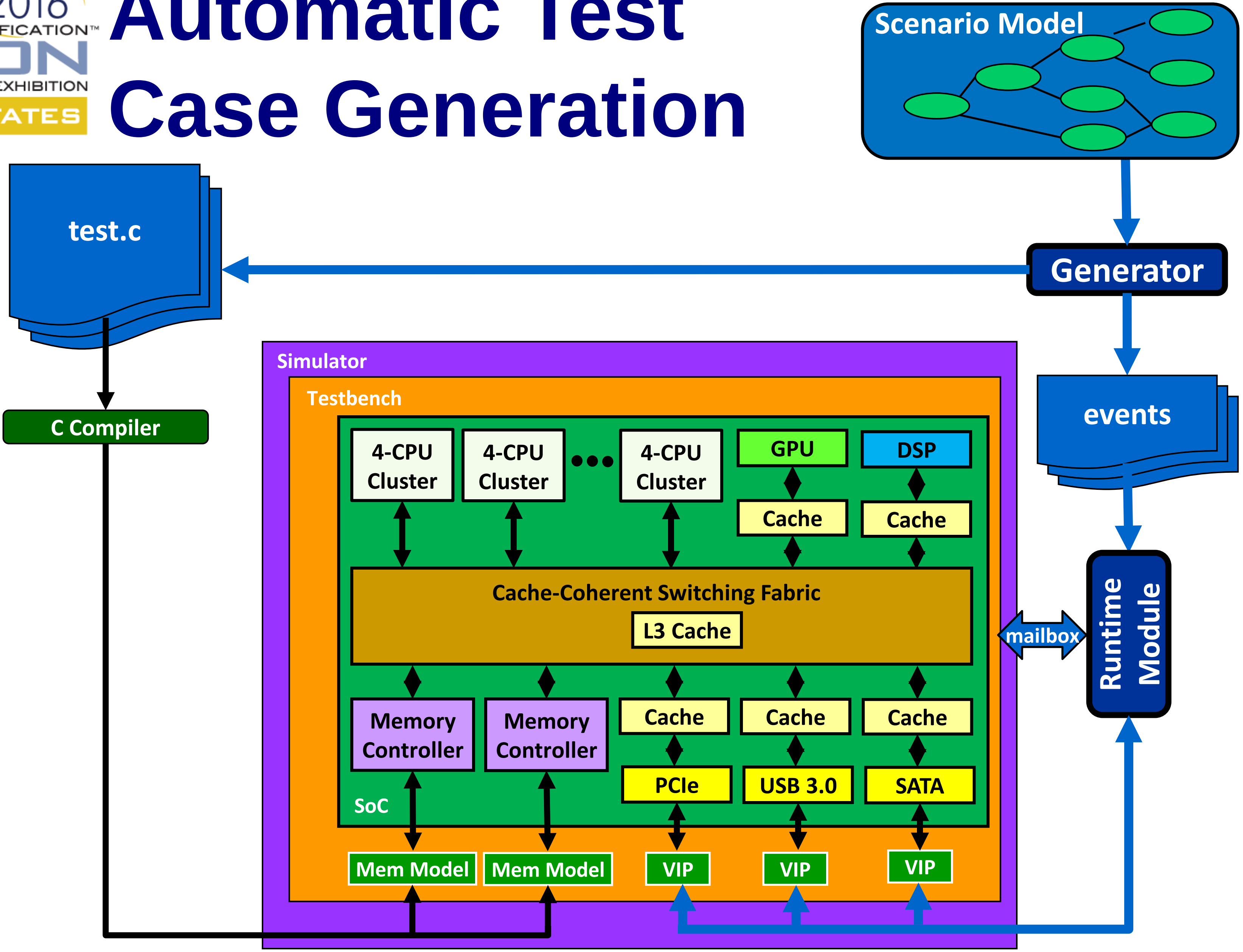
- Memory ordering
  - Needed for proper inter-processor communication
  - Example:
    - CPU0 writes data A followed by flag B
    - CPU1 waits for flag B, then reads data A
    - CPU1 *must* get data value A from CPU0
- Non-determinism
  - Impossible to predict exact cache transitions
    - Except in very simple cases
  - Focus on interesting cases and measure coverage

More Verification Challenges

- Single processor coherency state transitions
- Multi-processor coherency state transitions
- Crossing cache line boundaries
- Cache line evictions
- Page table properties
- Physical memory types
- Processor instructions
- Single vs. block operations
- Memory ordering tests
- Ordered vs. unordered tests
- Power scaling

Must Cross *All* These Variables

Automatic Test Case Generation



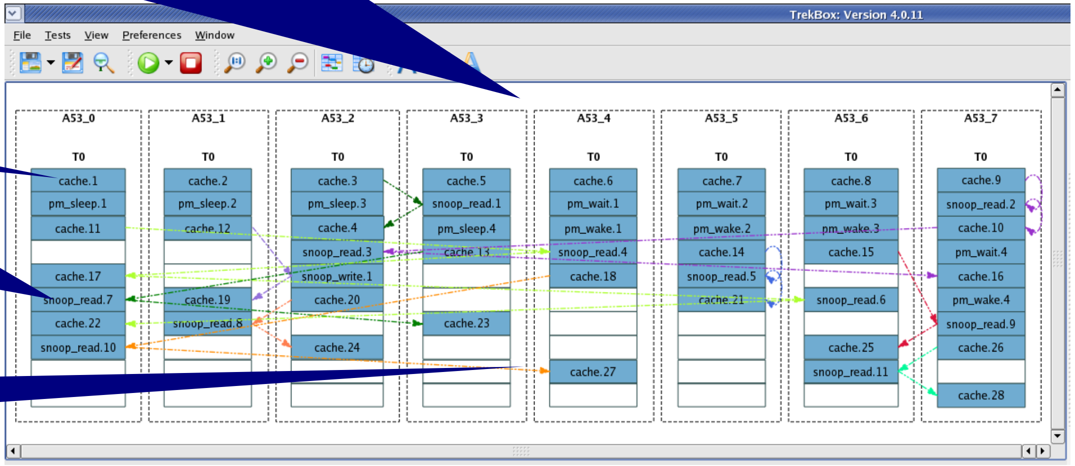
Multi-Processor Scheduling

Automatically Generated Concurrent Test Case for 8 Processors

Individual Writes, Reads, and Snoops Causing Cache Transitions

Self-Checking Test Case: All Reads Compare to Expected Values

Test Case Threads Interleaved and Synchronized across all Processors



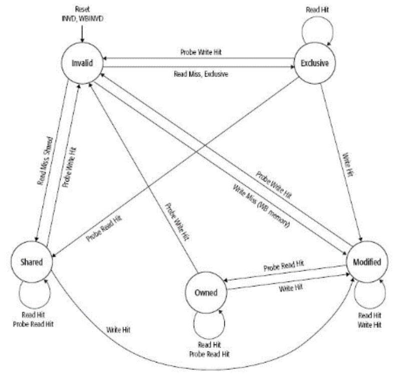
- Test cases generated from C++ scenario models
  - Powerful graph-based modeling representation
  - General solution for many verification challenges
  - Ideal for cache coherency since graphs are pre-built

Graphs Model Cache States

State Transition Table

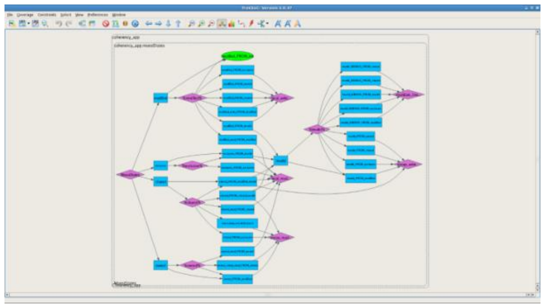
Current State				Cmd			Next State				Outputs / Expects
C0	C1	L2	CCU	Agent	Op	Coh	C0	C1	L2	CCU	
I	I	I	I	C1	LD	1	I	E	E	E	fetch
I	E	E	E	C0	ST	1	M	S	S	S	snoop C1
M	S	S	S	C1	LDEX	1	I	E	E	E	snoop C0
M	S	S	S	VIP	ST EX	1	I	I	I	I	invalidate C1

State Machine

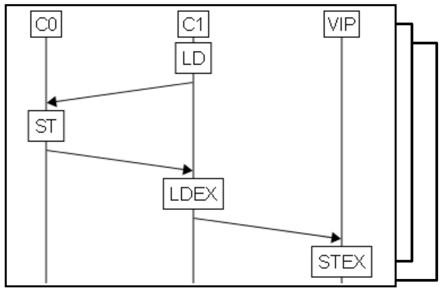


Graph

Map

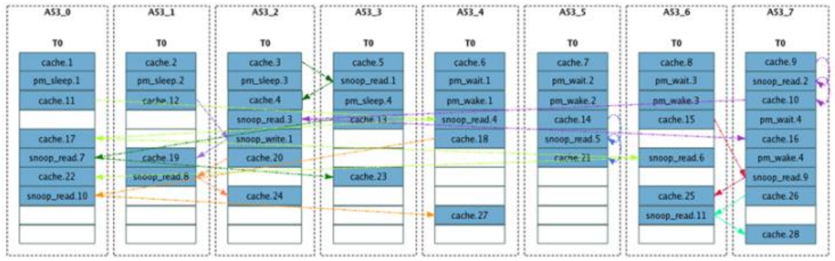


N Transition Scenarios



N Graph Traversals

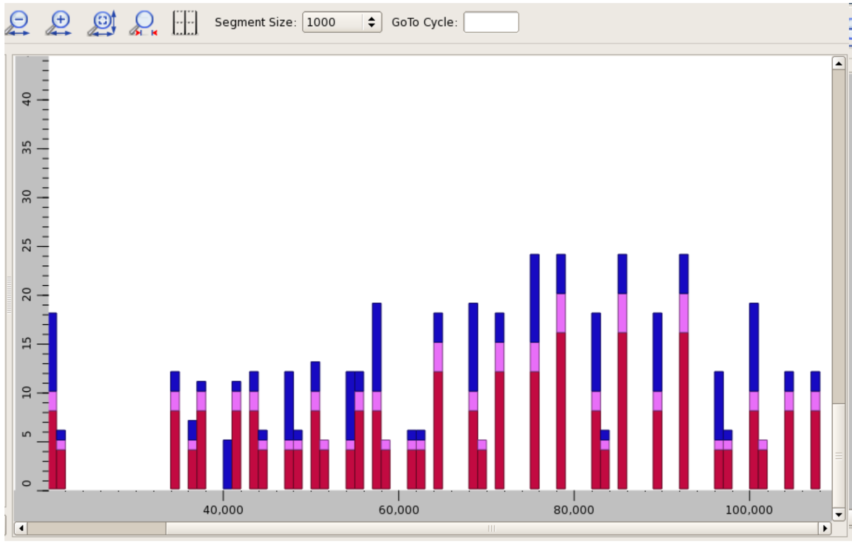
Concurrent Scenario Test Case



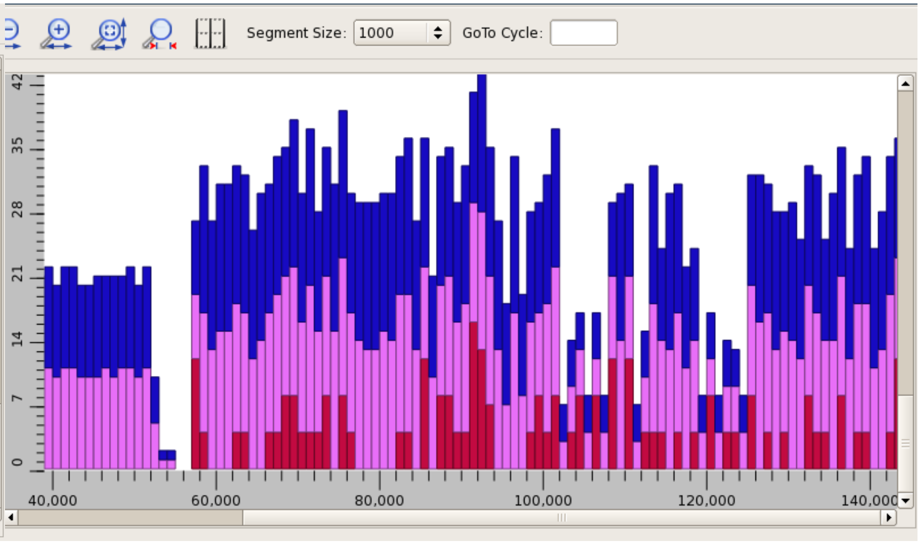
Generate

Better Verification Coverage

Metrics from Hand-Written Tests

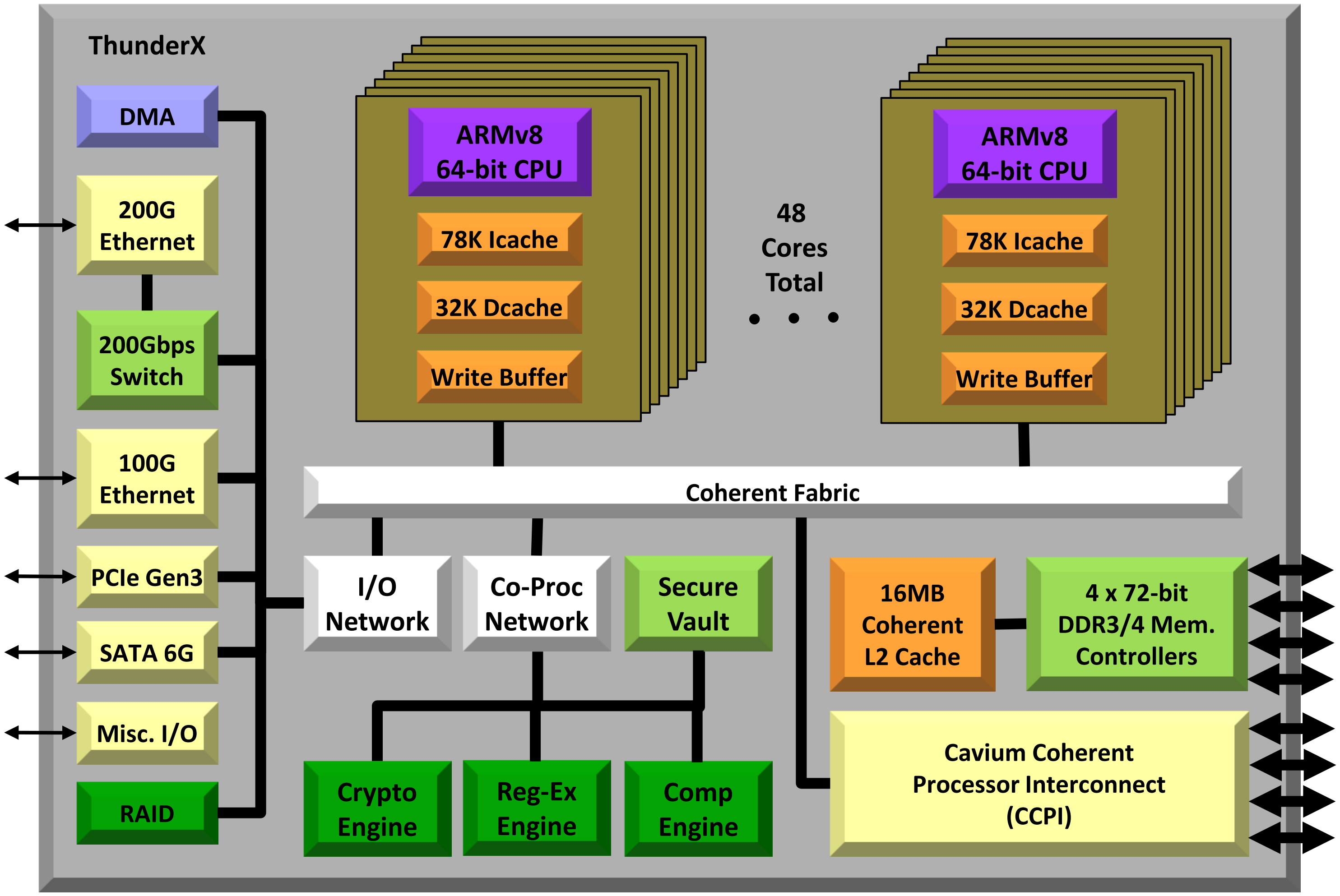


Metrics from Generated Test Cases



- Results from design with 8 ARM Cortex-A53 CPUs
  - Metrics for traffic and usage of key resources
  - Better coverage = much more thorough verification

Real-World Case Study



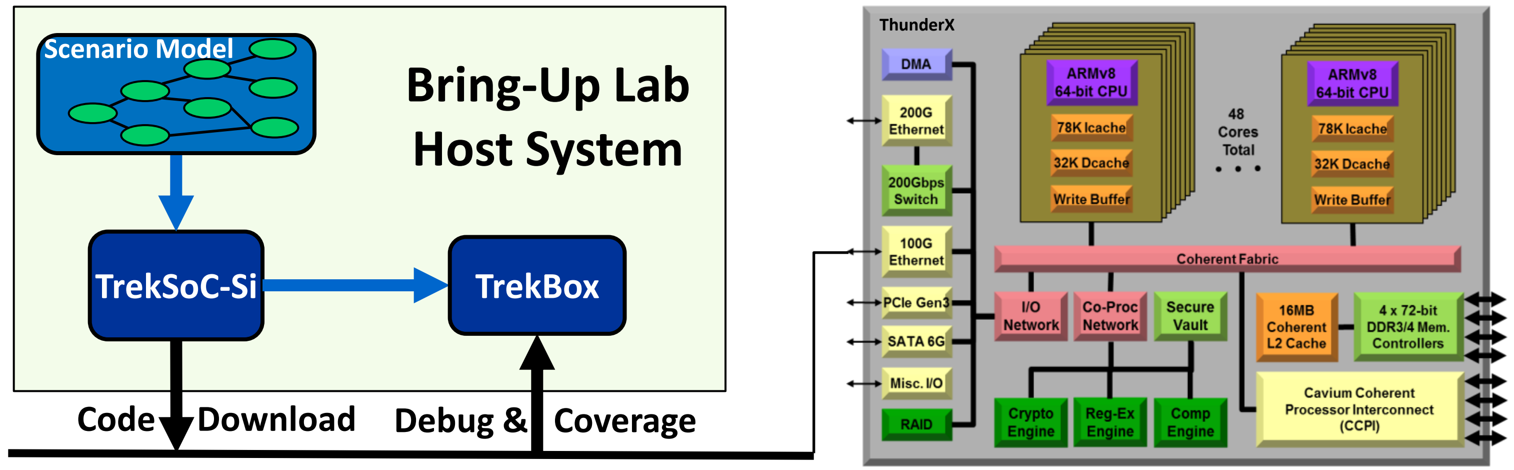
Cavium Verification Project

- Automatically generated test cases verified cache coherency for Cavium ThunderX product family
  - Up to 48 custom CPUs complaint with ARMv8
  - Four DDR3/4 memory controllers
  - 100/200 Gb Ethernet, PCI Express (PCIe), SATA, etc.
  - Cache coherent across dual sockets (two chips)
    - Cavium Coherent Processor Interconnect (CCPI) link
- Verification performed in the bring-up lab
  - Design too large for full-SoC or multi-SoC simulation
  - All 48 cores never run together pre-silicon

Verification Challenges

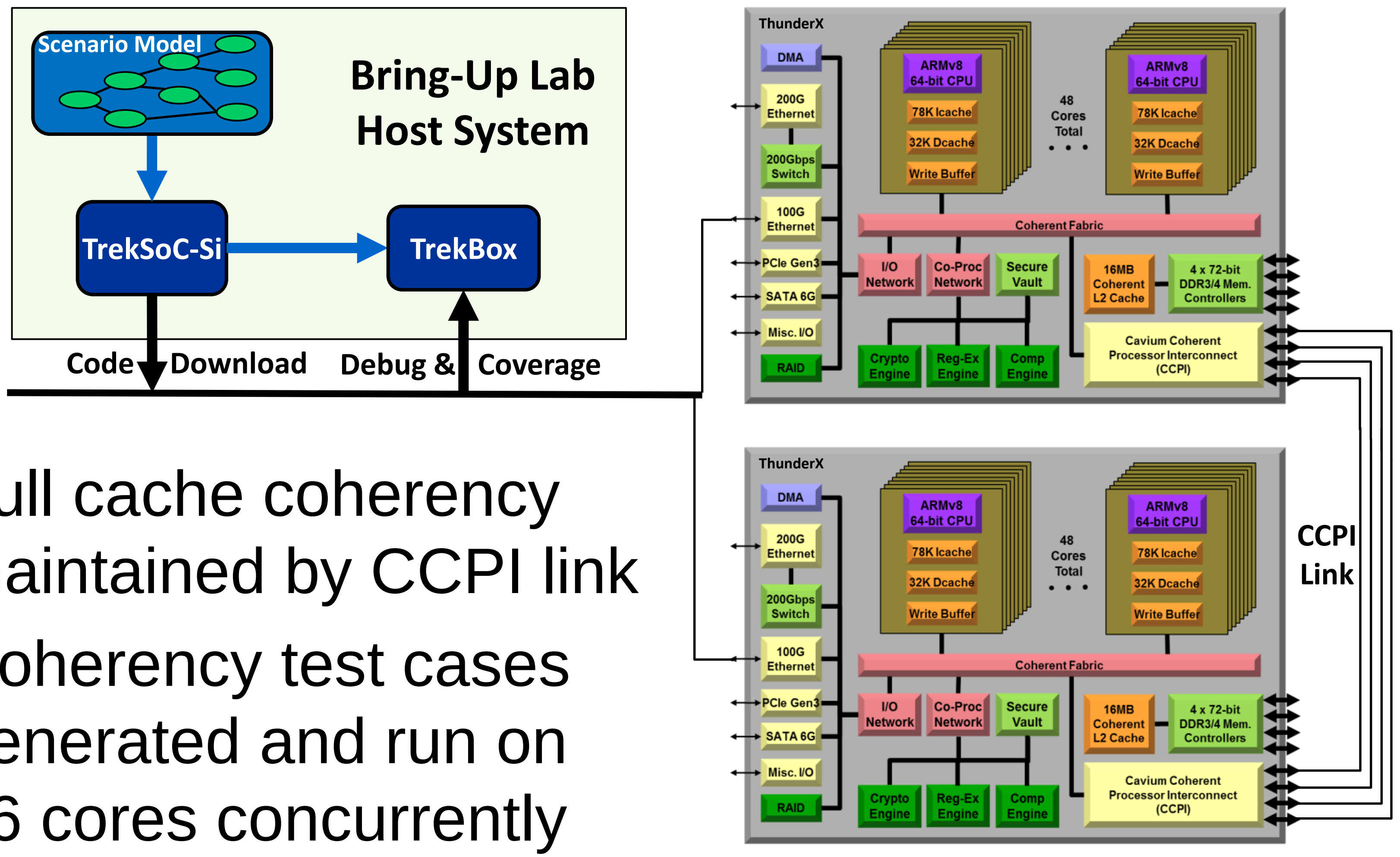
- Most verification is done at the cluster level
- Limited scope for system-level simulations
  - Slower simulation speeds, only a few cores are enabled
  - System-level tests, combination of coherent master traffic combined with non-coherent masters
  - Portability and scaling of stimulus from full chip to pre-silicon platforms
- Post-silicon validation
  - Verify all cache coherency scenarios
  - Verify dual socket configurations, with CCPI link
  - Be able to reproduce post-silicon failures, on pre-silicon platforms, without environment setup-related issues
- Stimulus portability
  - Flexibility to port the stimulus across platforms

Verification Process



- Step 1: Run code on all 48 cores at the same time
  - Validate models, generation, download, debug
- Step 2: Verify cache coherency across all 48 cores and all memory channels
- Step 3: Add IP-focused system randomization tests

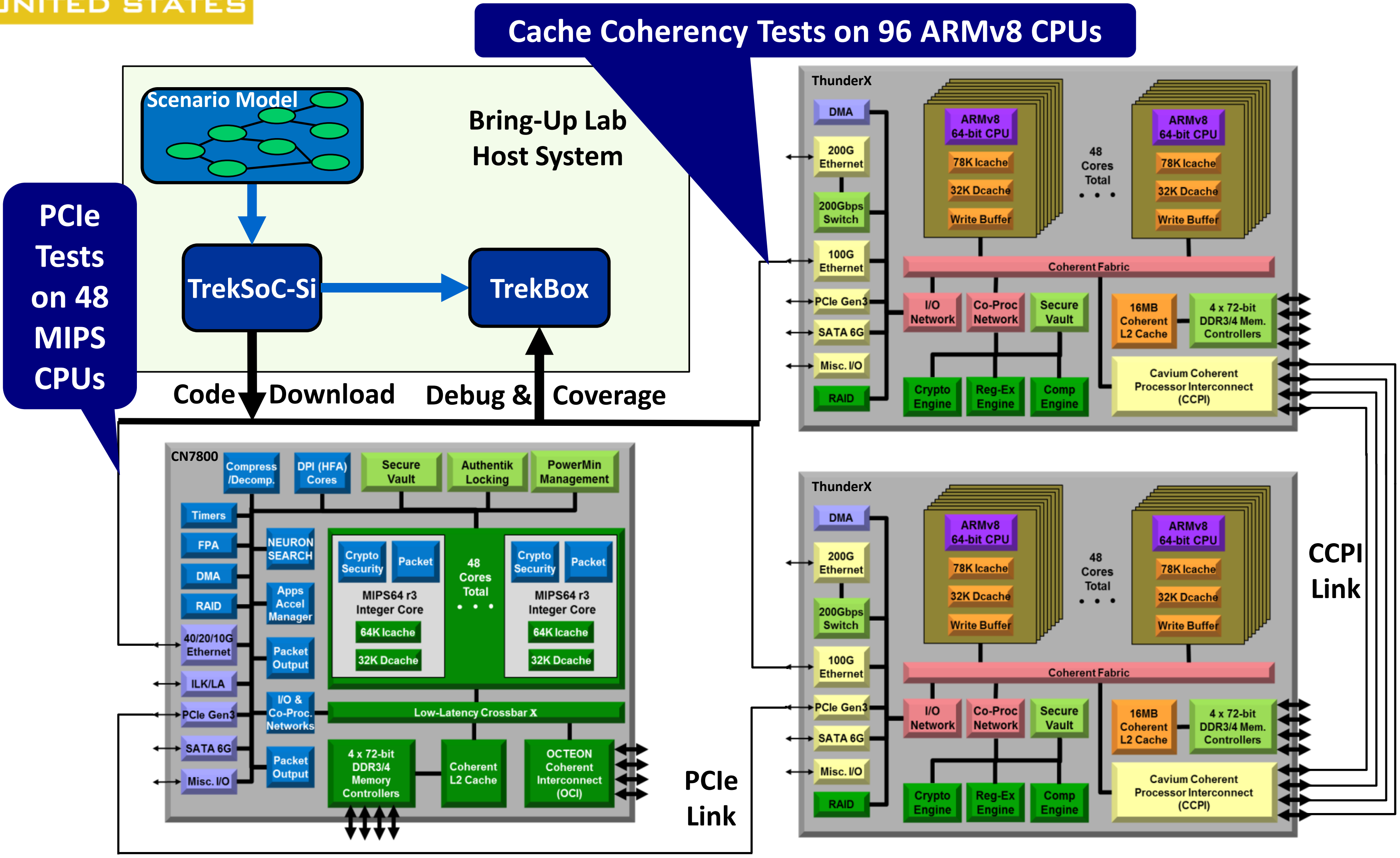
Step 4: Verify Dual-Socket



- Full cache coherency maintained by CCPI link
- Coherency test cases generated and run on 96 cores concurrently

DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES 2016

Step 5: Verify 144 CPUs



Cavium Project Results

- Cache coherency verification process successful
  - Running full system in the bring-up lab was required
    - Thorough verification a must before release to customers
  - ThunderX family now announced and shipping
- Cache coherency issues found at multiple levels
  - Some due to bring-up lab setup (download, etc.)
  - Others in Cavium driver-level software
- Some test cases running in simulation and emulation
- Cavium regards automatically generated test cases as essential for cache coherency verification

Cavium Learnings

- Off-the-shelf test generation and customization flexibility
  - Helps spend most of the engineer’s time in debugging the failure, rather than creating the stimulus
  - Flexibility: tweak the stimulus to create interesting scenarios
- Portability of the stimulus across platforms made easier
  - Be able to port the stimulus from post silicon to pre-silicon or from full chip to pre-silicon
- Scalability
  - Be able to scale up the stimulus scope from simulation to pre-silicon
- Debug capabilities
  - Pinpoint the failure to a particular thread and particular transaction
  - Quickly narrow down the problem, mask or unmask certain instructions or functions, etc.

Industry Conclusions

- This project was a landmark proof point for graph-based scenario models and “portable” test cases
  - Concurrent generation and run on 144 CPU cores
  - Support for SoCs and multi-SoC systems
  - Support for heterogeneous CPUs (ARM and MIPS)
  - Test cases portable: simulation↔emulation↔silicon
- Impact on Accellera PSWG will be significant
  - PSWG planning to use graph-based models
  - C++ is sufficient to develop these models
  - Real-world validation that <u>the Accellera vision is reality</u>

Thanks for Listening!

Questions?