2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION CHINA

accellera SYSTEMS INITIATIVE

Ensuring System-Level Coherency: A System-Level Framework for Verification and Measurement

Qingli He

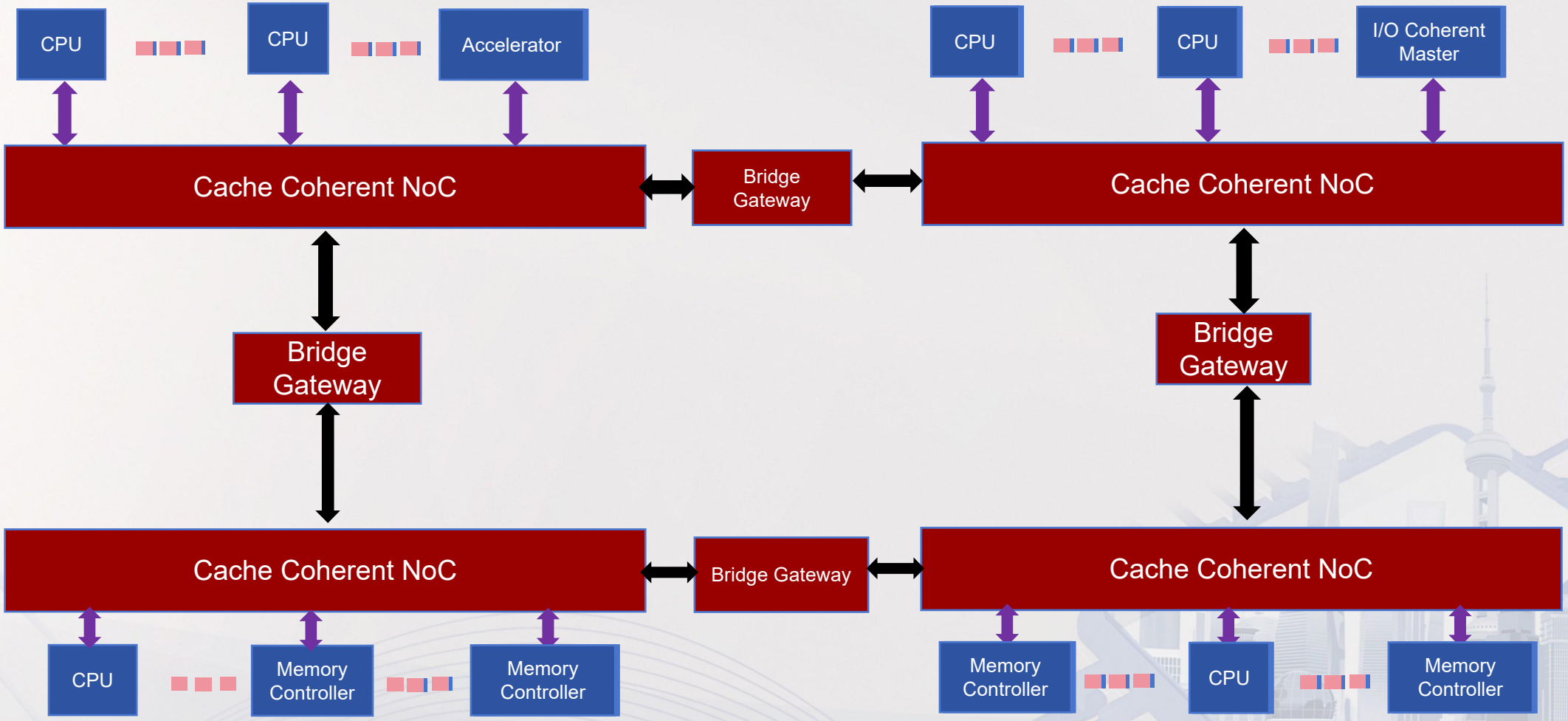
Staff Application Engineering

Agenda

- Challenges of Cache Coherency in SoC Verification
- Scalable Testbench Generation
- Rapid Functional Stimulus Generation & Coverage Closure
- System Level Coherency Checkers
- Smart Debugging to improve Turnaround time (TAT)
- End-to-End Automated Performance Verification

Cache Coherent SoC Design

Cascaded Coherent interconnects across multiple chips



Verification Challenges of Cache Coherent SoC

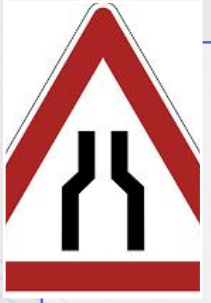
System Coherency Complexity

Testbench Scalability

System Level Functional Scenarios

System Coherency

Smart Debug

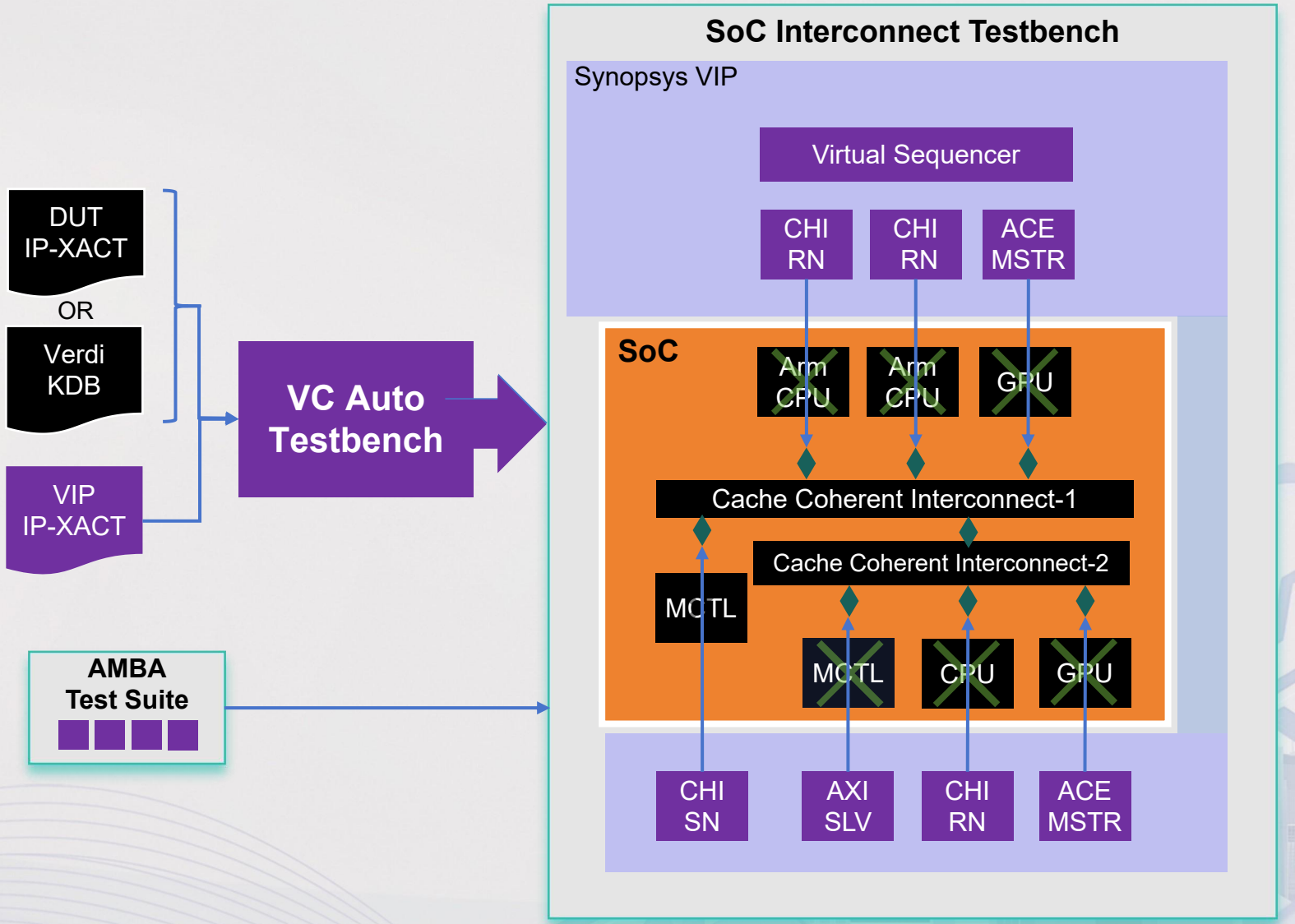


Performance Bottleneck

Scalability : Automated SoC Testbench Generation

Reduce Time-to-first-test from Weeks to Hours

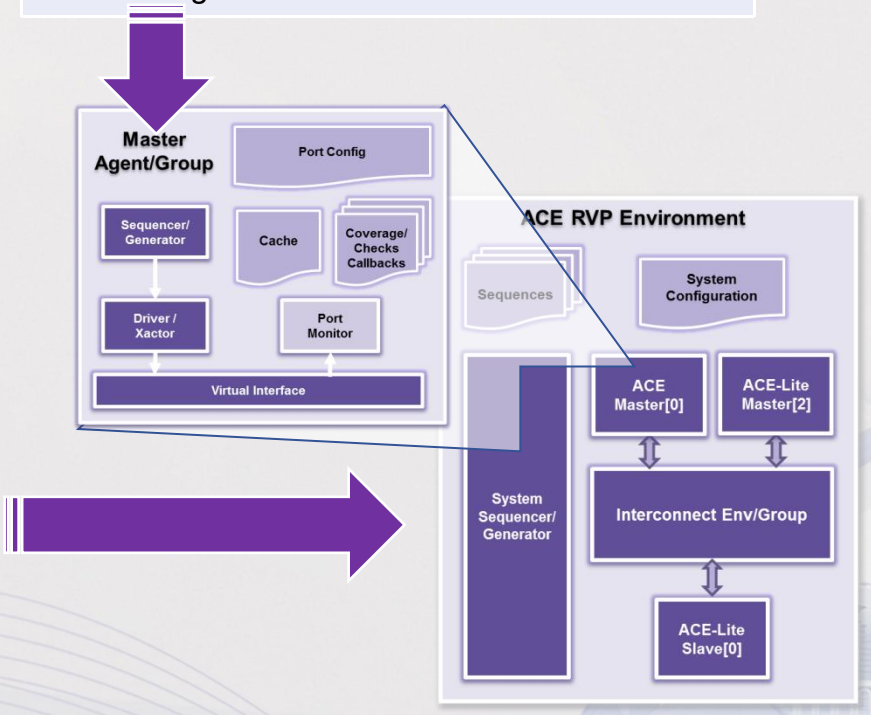
- SoC environment generation reading the DUT topology
- Both passive & active RTL replacement



Predefined Scenarios for Coherency Verification

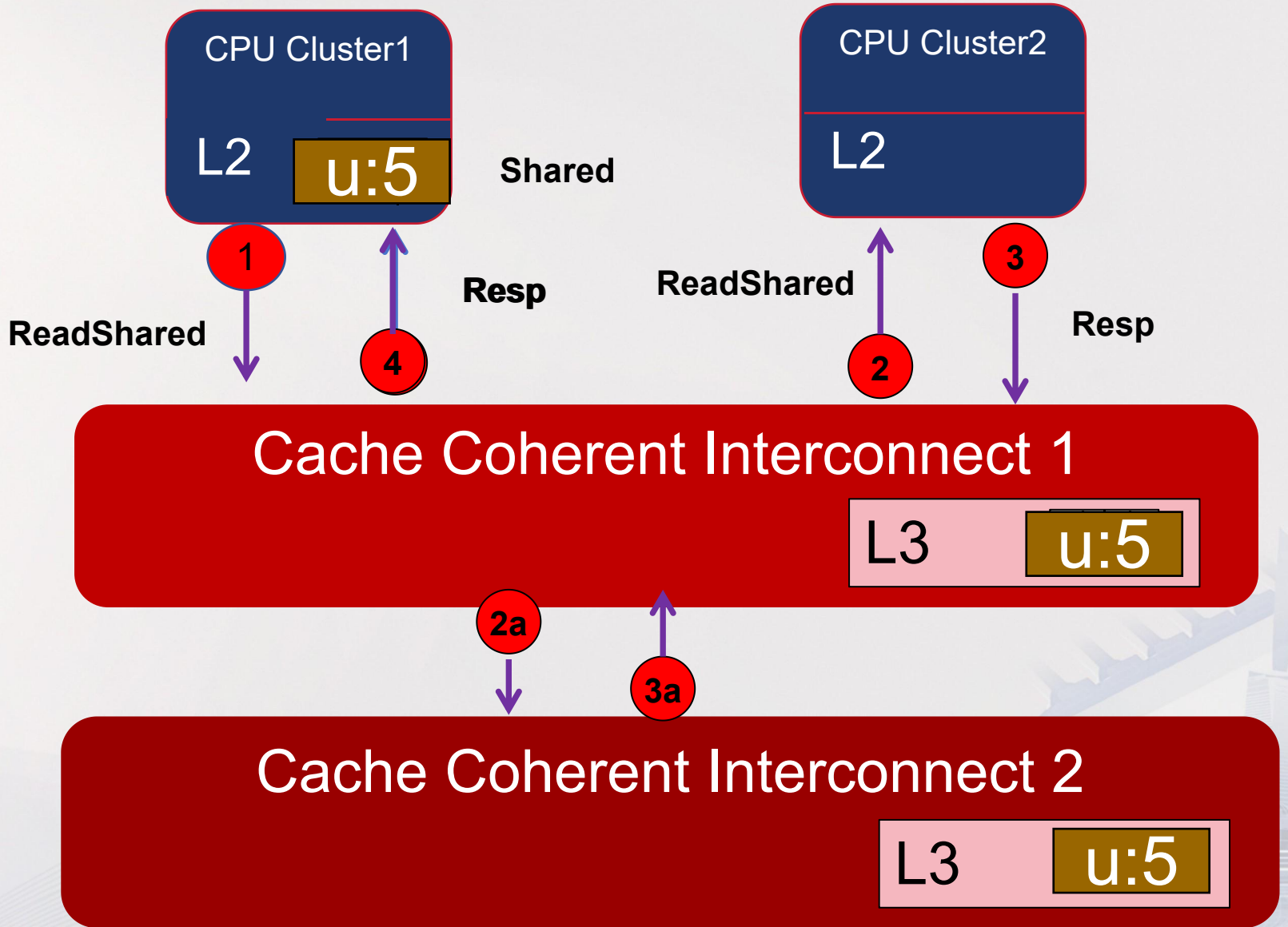
Transaction type	Pre-defined Coherent Sequences
Non-snooping	ReadNoSnp
	WriteNoSnp
Coherent	ReadClean
	ReadNotSharedDirty
	ReadShared
	ReadUnique
	CleanUnique
	MakeUnique
	ReadOnce
	WriteUnique
	WriteLineUnique
Memory update	WriteBack
	WriteClean
	WriteEvict
	Evict
Cache Maintenance	CleanShared
	CleanInvalid
	MakeInvalid
Others	DVM
	Barrier
	Snoop During Memory Update
	Overlapping Address
	Exclusive Reads
	Exclusive Writes

System Level Sequences
Connectivity sequences
Multi chip Coherent Random Traffic
Blocking Alternate Load & Stores
Multichip Concurrent transactions
Exclusive accesses
Unaligned Address



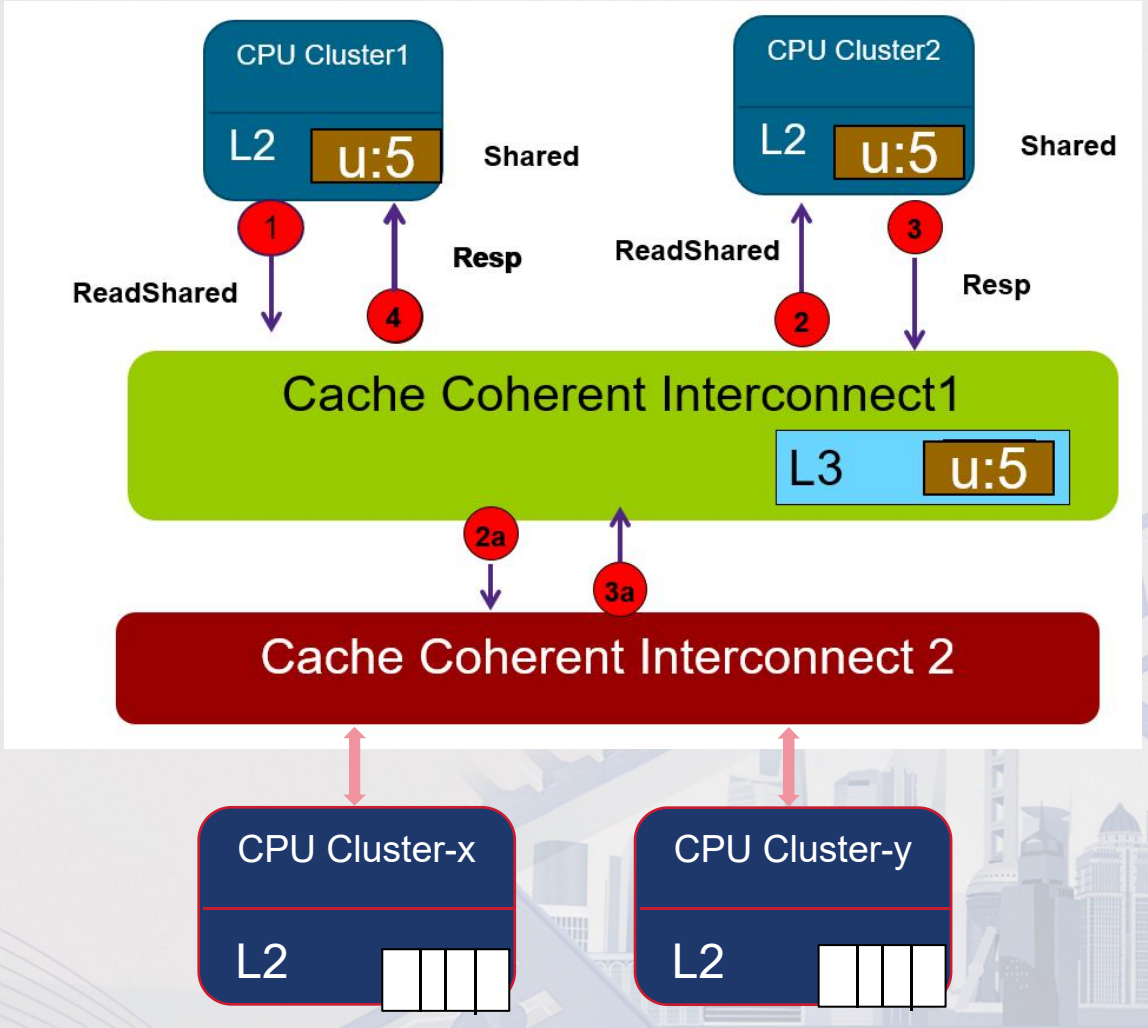
Cache Coherency across Interconnects

Performing coherent operation in shareable location



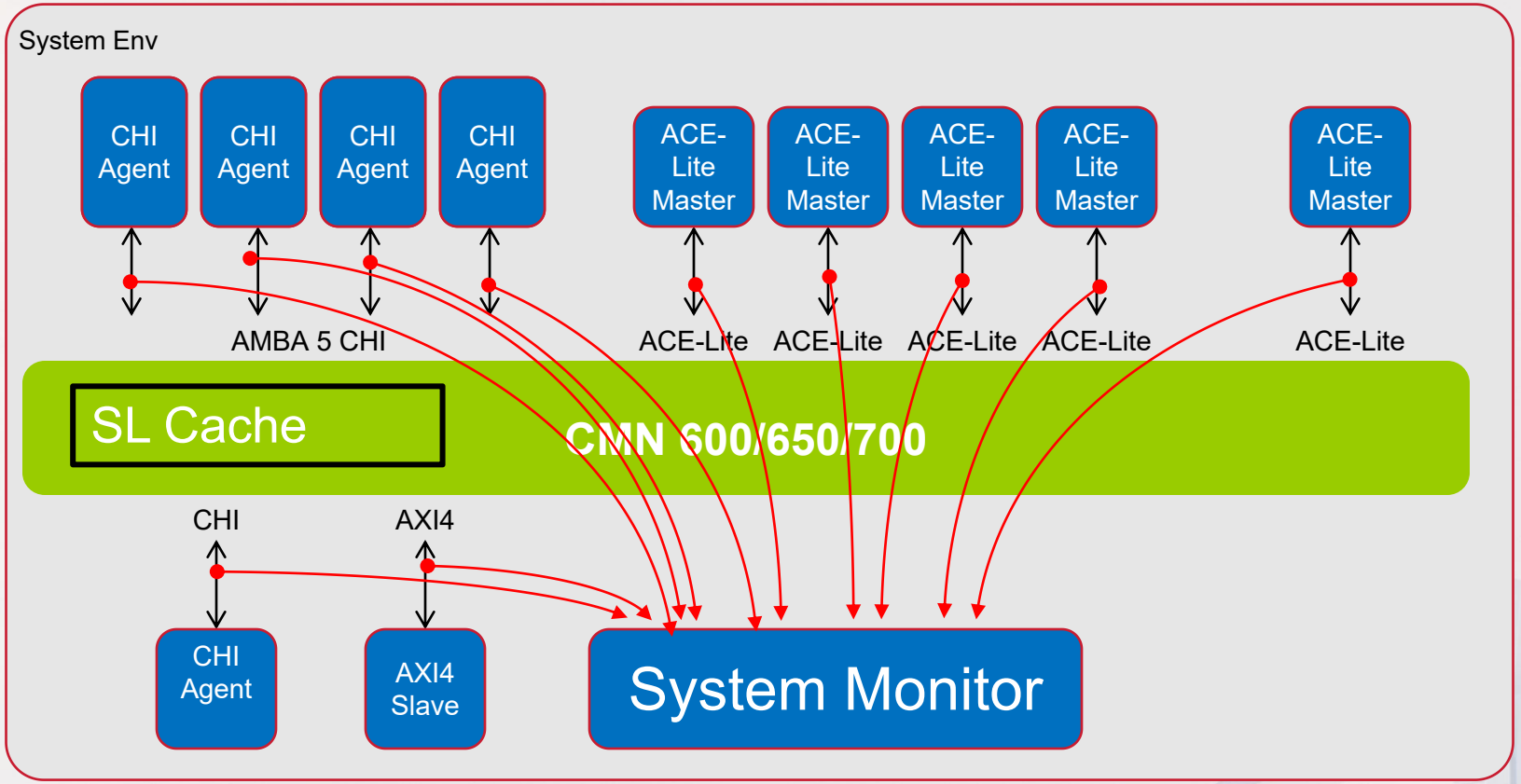
System Checks for Coherency

- Maintain coherency across caches
- Coherency across master cache between chips
- Issuing Snoop transactions
- Sequencing transactions
- Data integrity between snoop and coherent transactions
- Coherency across master cache and Interconnect cache



Single Chip -CHI System Monitor

Perform Check across CMN600/650/700

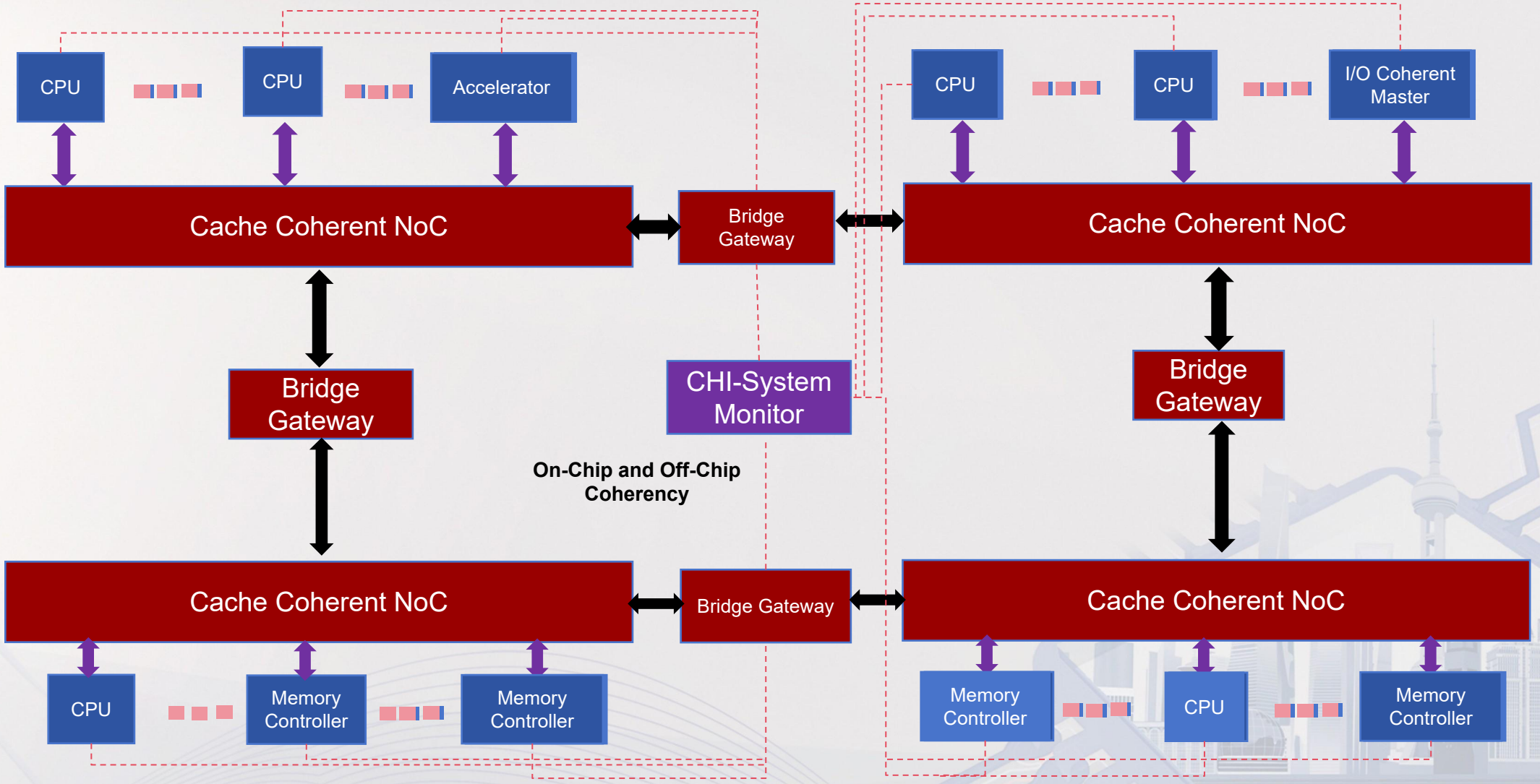


Single Chip CHI System Monitor

- Performs system checks across interconnect ports
- Main categories of System Level checks:
  - Correctness of mapping between coherent transaction and snoop transactions
  - Correctness of sequencing between coherent and snoop transactions
  - Correctness of response to coherent transactions based on response to snoop transactions
  - Data integrity between coherent transactions and snoop transactions
  - Coherency between master caches
  - Data integrity across transaction data
  - Slave Routing Check

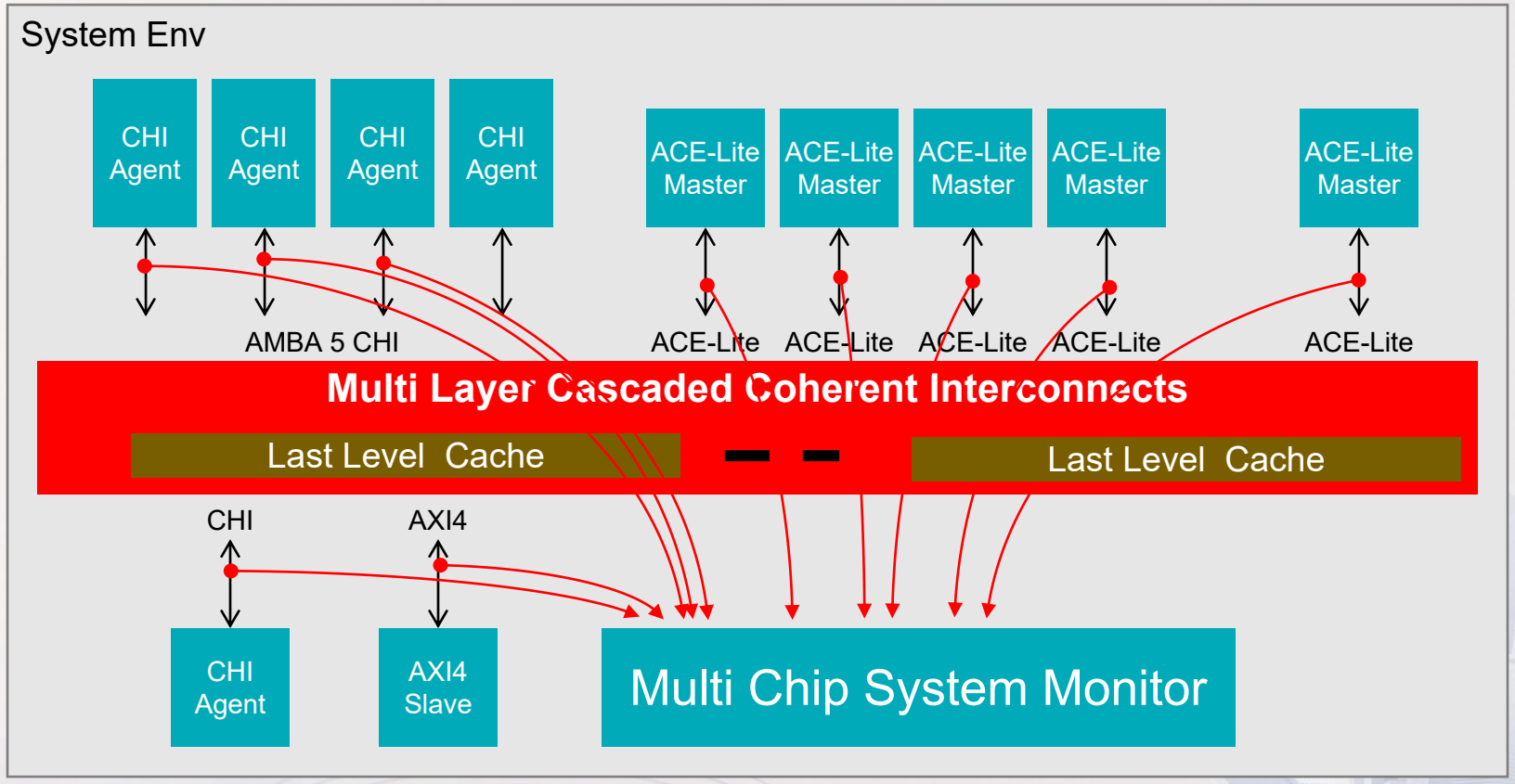
System Coherency Challenge across Chips

Cascaded Coherent interconnects across multiple chips



Multi-Chip Coherent System Monitor

Performs System Level checks across interconnect ports

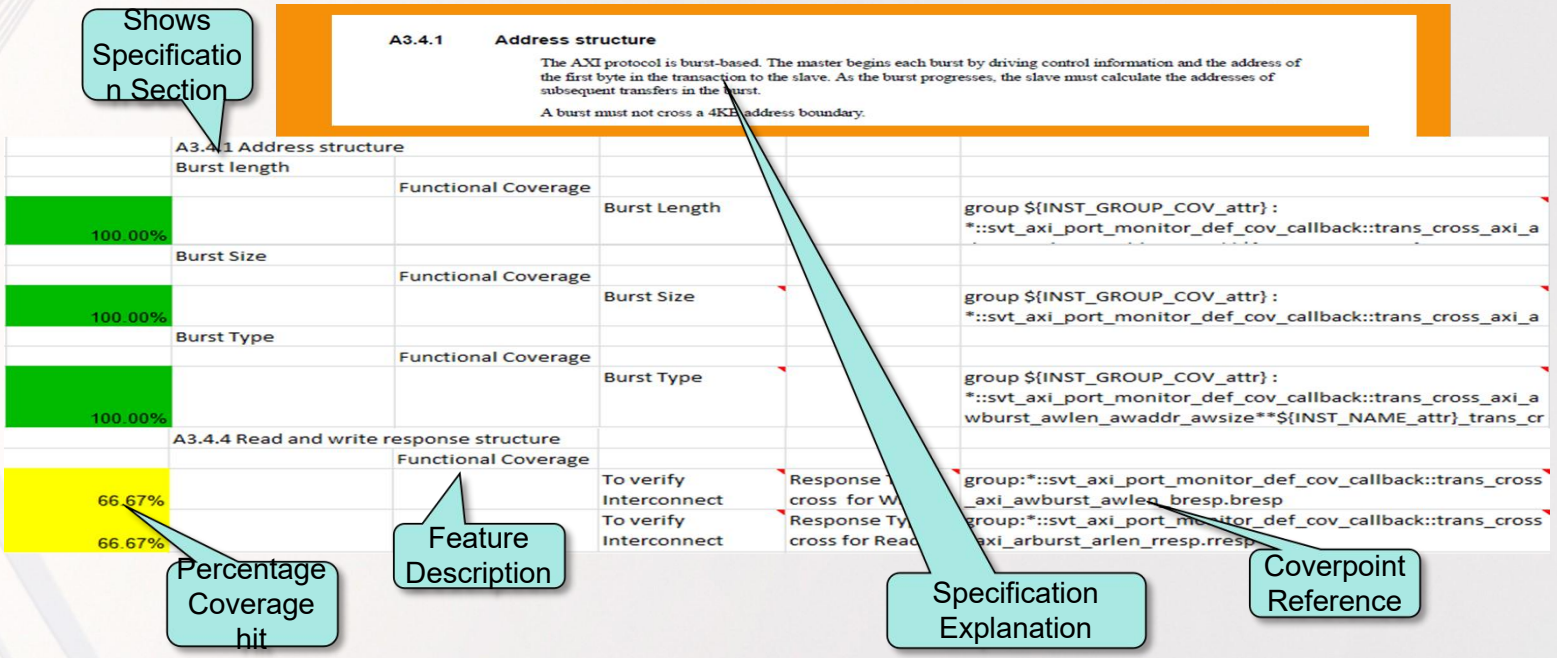


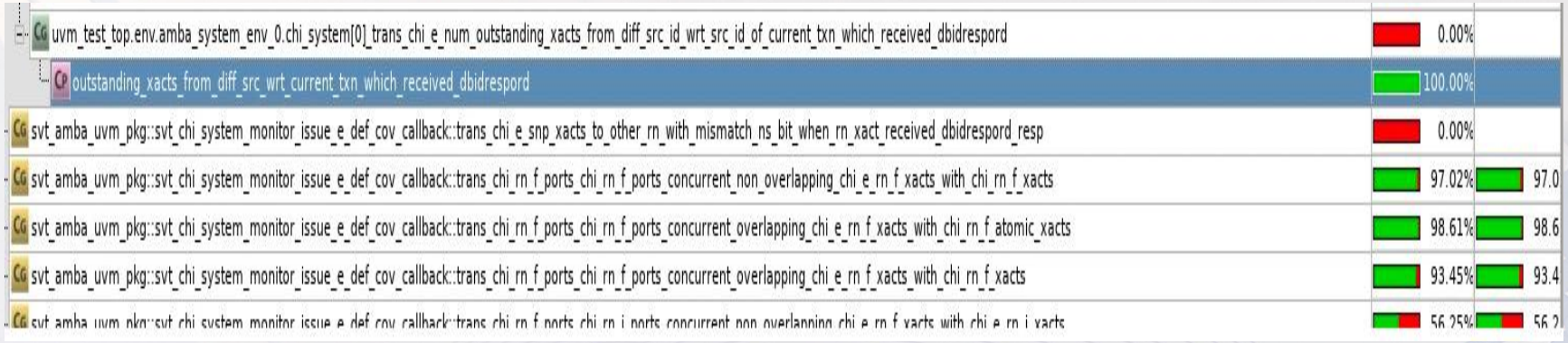
Multi Chip CHI System Monitor

- Performs system checks across interconnects
- Main categories of System Level checks:
  - Correctness of mapping between coherent transaction and snoop transactions
  - Correctness of sequencing between coherent and snoop transactions
  - Correctness of response to coherent transactions based on response to snoop transactions
  - Data integrity between coherent transactions and snoop transactions
  - Coherency between master caches
  - Data integrity across transaction data
  - Slave Routing Check

Coverage Closure across the System

Protocol Specification to Verification Plan Closure





Verification plan mapped to Functional Requirements

- Mapped to specification sections
- Targeted Application sub-plans
- Back annotate coverage data to show progress
- Identify Coverage holes

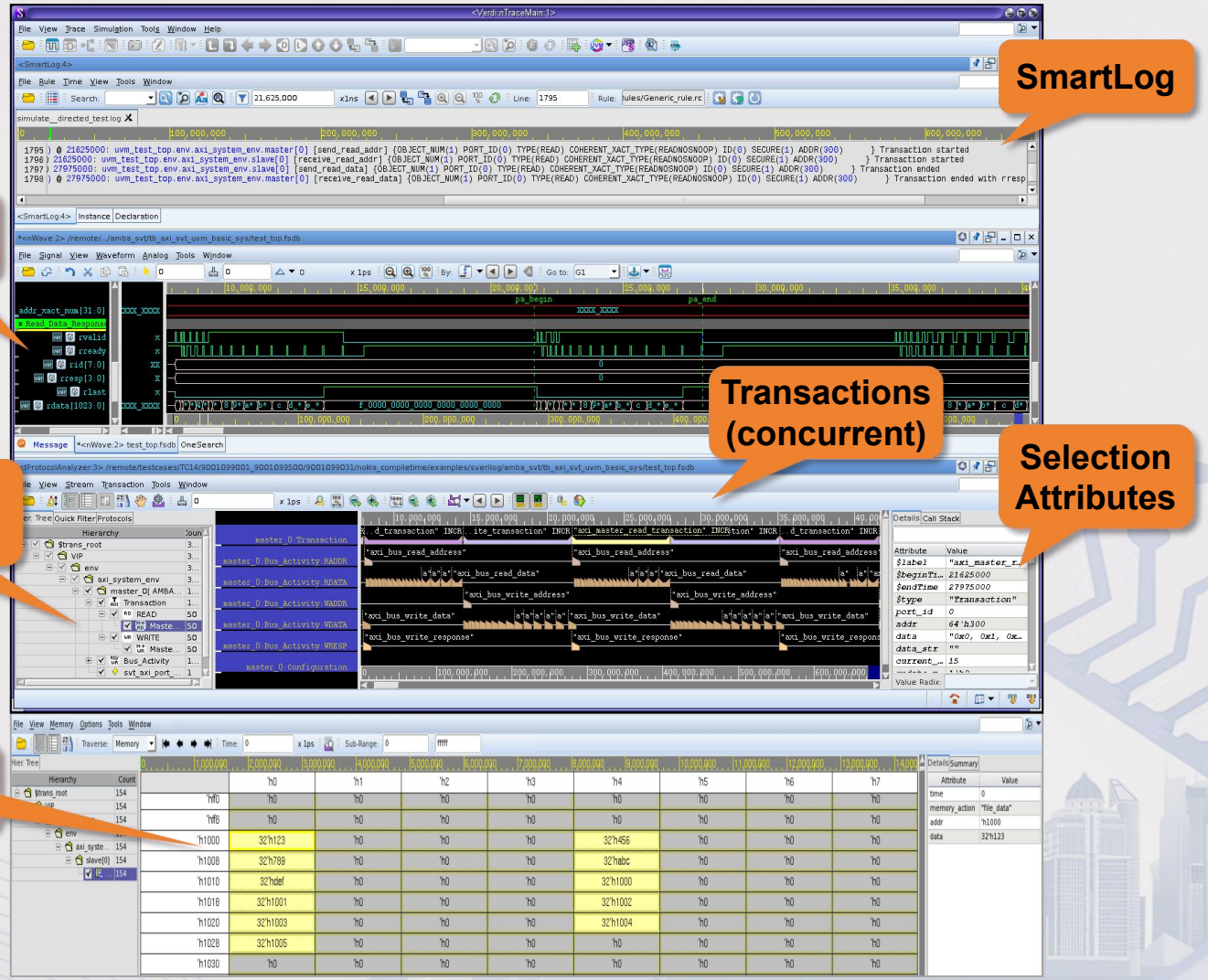
Configuration aware Functional coverage model

- Scenario coverages for connectivity
- Application specific scenario coverage
- Predefined transaction coverage

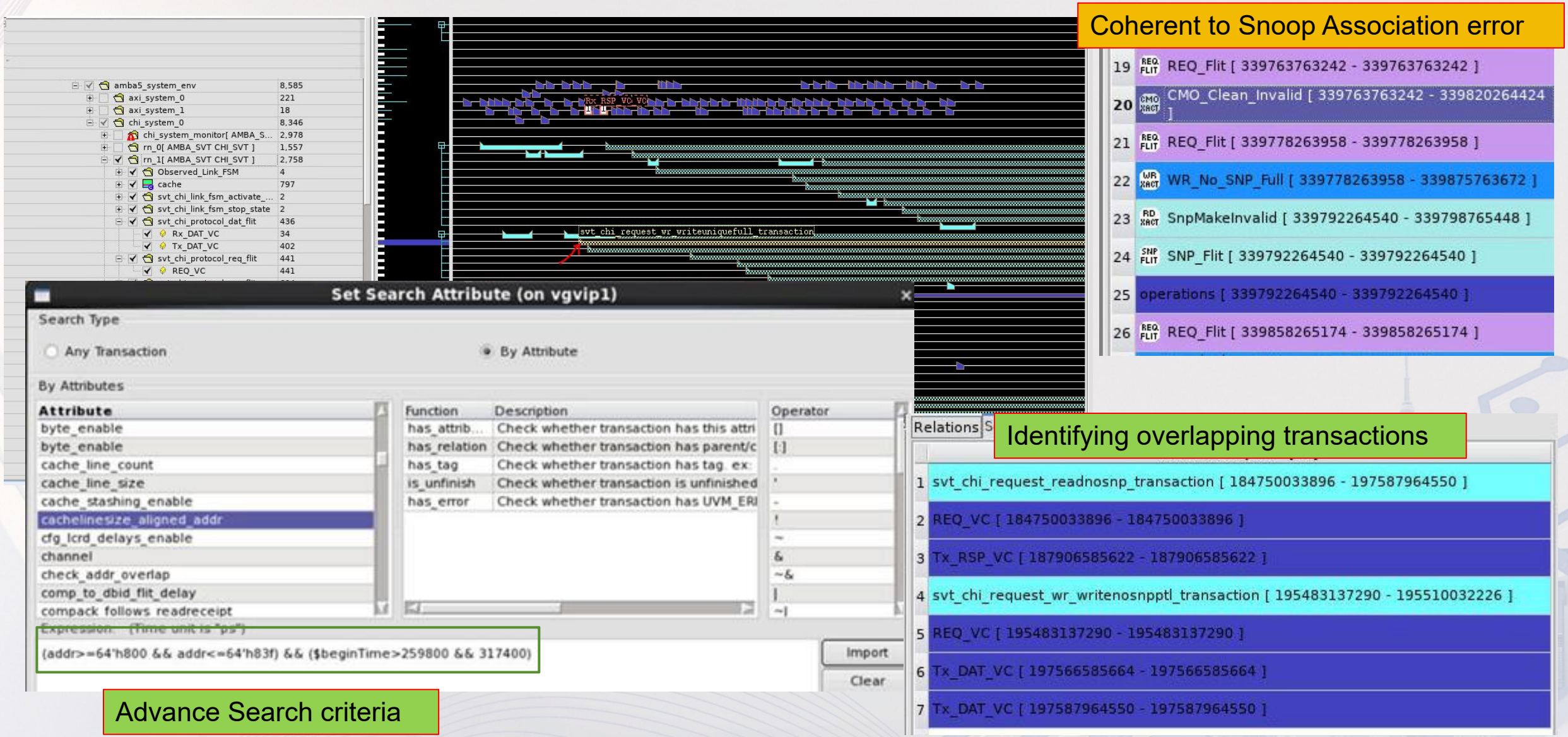
Quickly Identify the Source of Bugs

Verdi Protocol Analyzer and Memory Protocol Analyzer

- Protocol-aware transactions & attributes
- Synchronize time with waveform viewe SmartLog, etc.
- Link transactions to signals
- Find bugs quicker
- Gain insight into memory operations
- Flexible interactive or post-processing mode

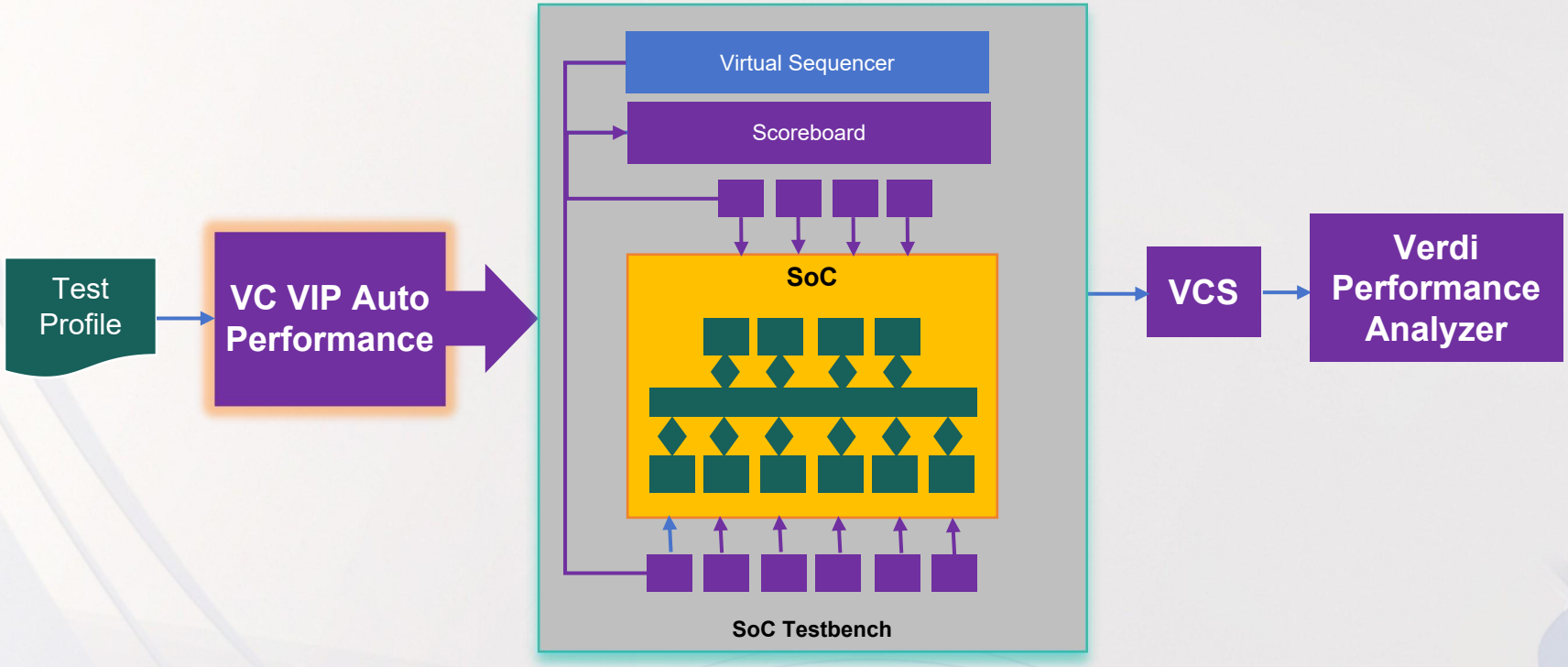


Smart Debug using Protocol Analyzer



Complete Performance Verification

Using ARM Adaptive Traffic Profile



User provides test profile

- Defines test grouping, sequencers, traffic profiles & resources

VC VIP AutoPerformance generates AMBA traffic to match profile

AMBA VIP built-in traffic rate adapter & arbiter drive profile on DUT

User analyzes performance simulation results with Verdi Performance Analyzer

Defining a Performance Test

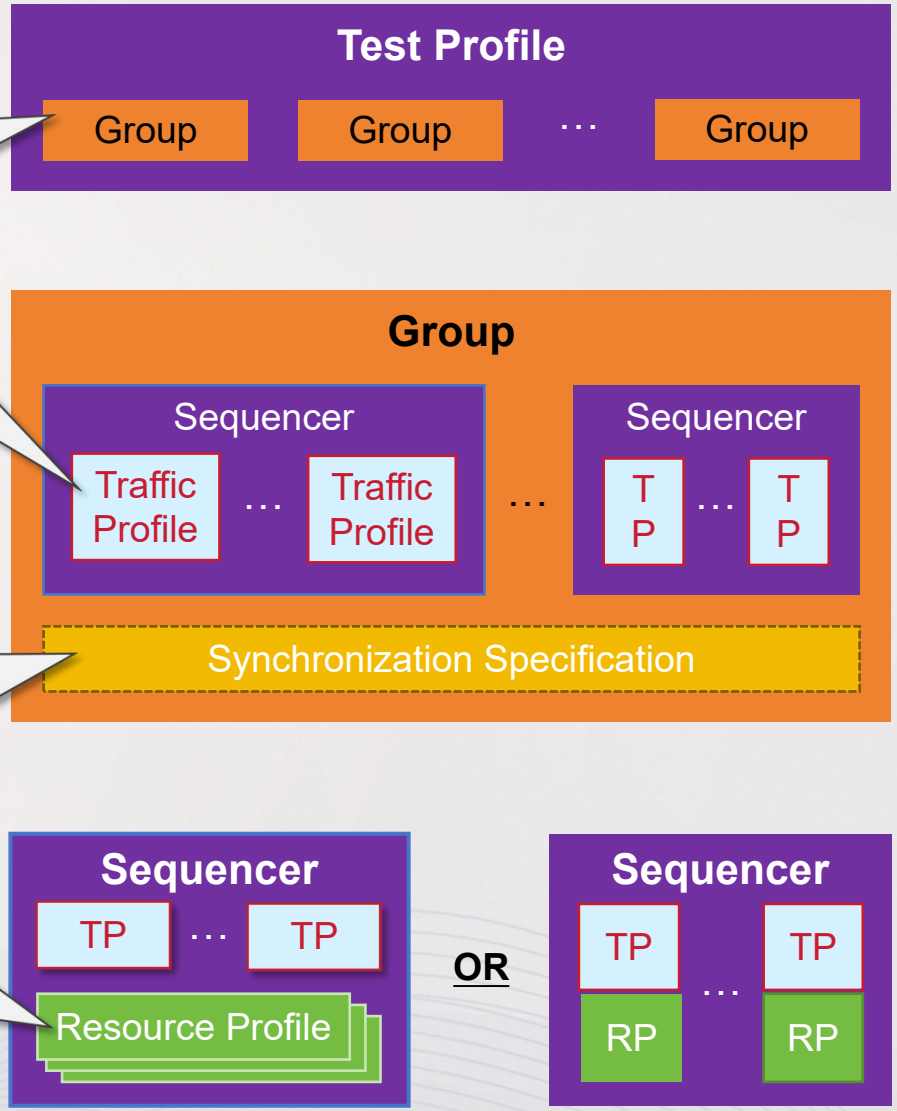
VC VIP AutoPerformance

Each **Group** is executed sequentially

The **Traffic Profiles** in a given **Group** are executed concurrently

A **Synchronization Specification** coordinates the execution of **Traffic Profiles** within a Group

Multiple **Resource Profiles** can be associated with a sequencer




```
<traffic_profile xmlns="http://www.synopsys.com/vc_speedtest_axi_traffic_profile">
  <!-- This initiates device memory WRITE transactions to
       sequential address from 32'b0100_0000 -->
  <master>
    <axi
      total_num_bytes="4096"
      xact_size = "64"
      xact_action = "STORE"
      xact_gen_type = "FIXED"
      xact_type = "WRITENOSNOOP"
      cache_gen_type = "RANDOM"
      cache_type_min = "4'b0000"
      cache_type_max = "4'b0001"
      prot_gen_type = "FIXED"
      prot_type_fixed = "SECURE"
      addr_gen_type = "SEQUENTIAL"
      base_addr = "'h2004_0000"
      addr_xrange = "'h2006_0000"
      addr_twodim_stride = "'h200C_0000"
      id_gen_type = "FIXED"
      id_min="16'h6"
      id_max="16'h6"
      qos_gen_type = "RANDOM"
      qos_min="8'h4"
      qos_max="8'hf"
      data_gen_type = "RANDOM"
      data_min = "64'h0"
      data_max = "64'hffff_ffff"
      artv = "FIXED"
      artv_min = "1"
      artv_max = "1"
      awtv = "RANDOM"
      awtv_min = "1"
      awtv_max = "4"
      rbr = "0"
      rla = "1"
      awv = "1"
      wbv = "1"
      br = "0"
      ba = "1"
    />
```

CHI/ACE Traffic Profile

Traffic Profile Performance Test

cache_type range indicates device type transactions

```
<?xml version="1.0" encoding="utf-8"?>
<traffic_profile
xmlns="http://www.synopsys.com/vc_speedtest_axi_traffic_profile">
<master>
    <axi
      total_num_bytes="4096"
      xact_size = "64"
      xact_action = "LOAD"
      xact_gen_type = "FIXED"
      xact_type = "READNOSNOOP"
      cache_gen_type = "RANDOM"
      cache_type_min = "4'b0000"
      cache_type_max = "4'b0001"
      prot_gen_type = "FIXED"
      prot_type_fixed = "SECURE"
      addr_gen_type = "SEQUENTIAL"
      base_addr = "'h4000_0000"
      addr_xrange = "'h4002_0000"
      addr_twodim_stride = "'h400C_0000"
      id_gen_type = "FIXED"
      …
     />
  </master>
</traffic_profile>
```

Initiates device memory WRITE transactions to sequential address from 32'h4000_0000

addr_twodim_stride is not applicable since addr_gen_type is sequential user could also opt not to specify it at all

Test Profile CHI & AXI Control

```
<group name="memory writes">
      <sequencer instance="uvm_test_top.env.amba_system_env_0.chi_system[0].rn[0].rn_xact_seqr"
                 name = "rn_0_sequencer">
<traffic_profile path="mem_normal_writes_sequential_1_profile.xml"
                          name="mem_normal_writes_sequential_1_profile"/>
      </sequencer>
      <sequencer instance="uvm_test_top.env.amba_system_env_0.chi_system[0].rn[1]. rn_xact_seqr"
                 name = "rn_1_sequencer">
        <traffic_profile path="mem_normal_writes_sequential_2_profile.xml"
                         name="mem_normal_writes_sequential_2_profile"/>
      </sequencer>
      <sequencer instance="uvm_test_top.env.amba_system_env_0.axi_system[0].master[2].sequencer"
                 name = "master_2_sequencer">
         <traffic_profile path="mem_normal_writes_sequential_2_profile.xml"
                          name="mem_normal_writes_sequential_2_profile"/>
      </sequencer>
                       <!-- Synchronization specification -->
      <synchronization_specification>
        <output_event name="processor_1_end_of_profile"
                      output_event_type="end_of_profile"
                      sequencer_name="rn_0_sequencer"
                      traffic_profile_name="mem_normal_writes_sequential_1_profile" />
      </synchronization_specification>   <!-- Group: memory reads -->
      …
   </group>
</test_profile>
```

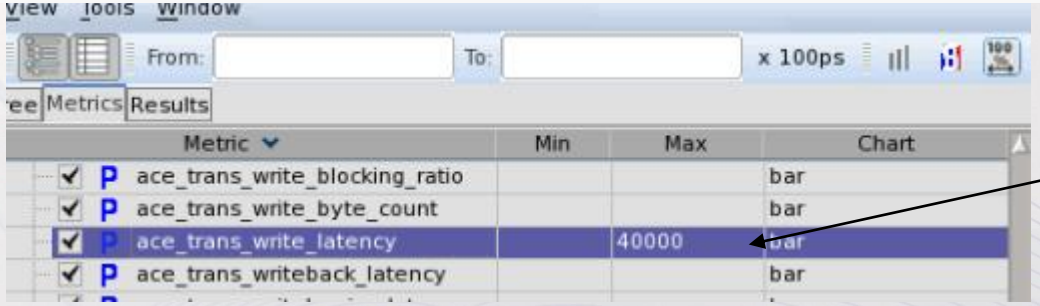
Define memory writes

Sequencers for writes

Synchronizatio n

Write Latency Observed at a Slave

This port of the Interconnect is connected to the Memory Controller

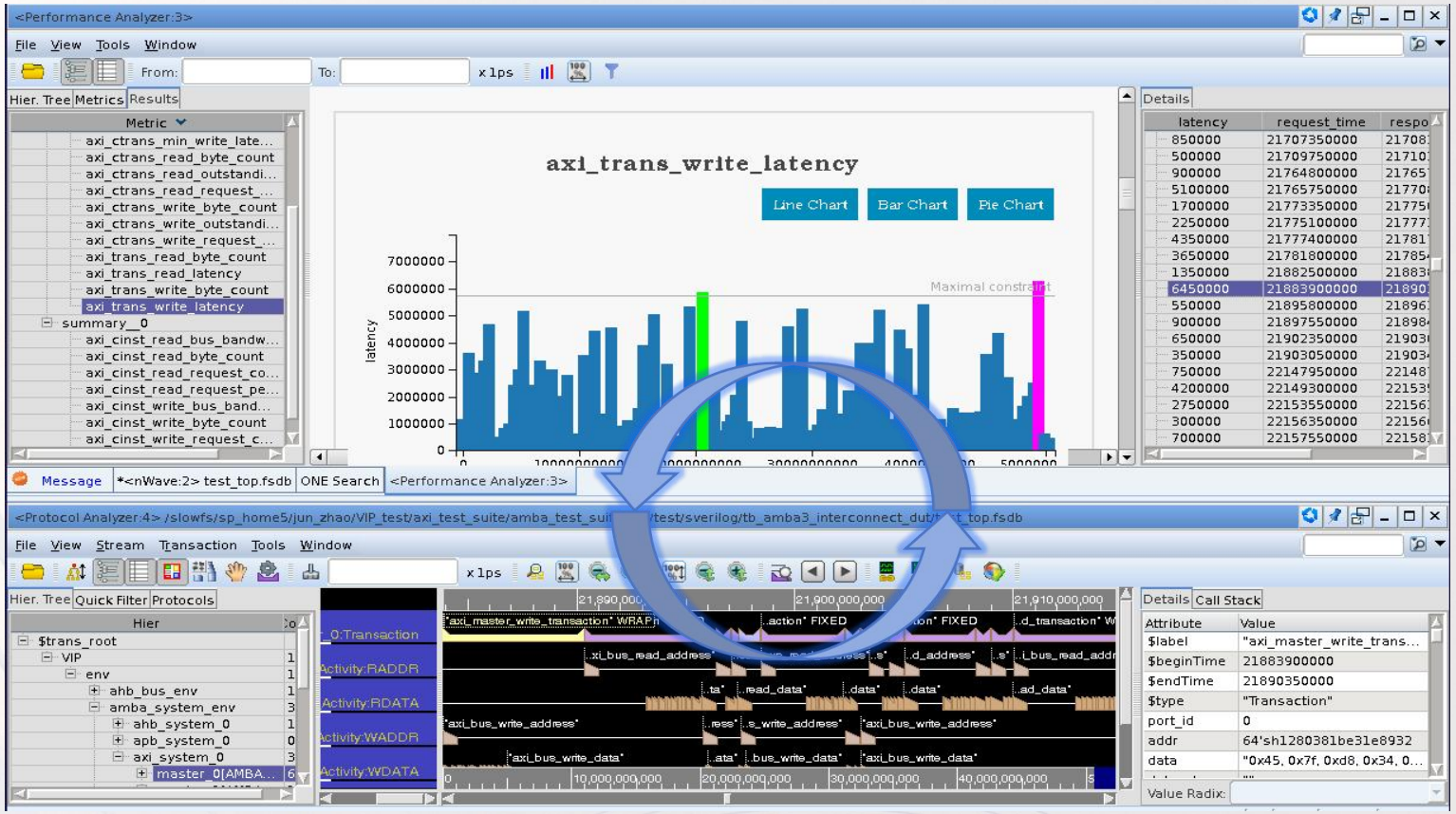


Max Latency as per architecture : 40000

Violation Slow Slave!

Analyze & Debug Performance Violations

Verdi Performance Analyzer for latency, bandwidth & throughput violations



Pre-defined metrics

- Latencies, bandwidths, counts, etc.

Supports user-defined metrics

- Based on SQL query statements

Apply constraints to detect violations

GUI charts to visualize distributions & violations

Trace violations to the related transactions

Export performance reports and charts

Batch mode support to regress multiple tests