



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

Qingli He
Staff Application Engineering

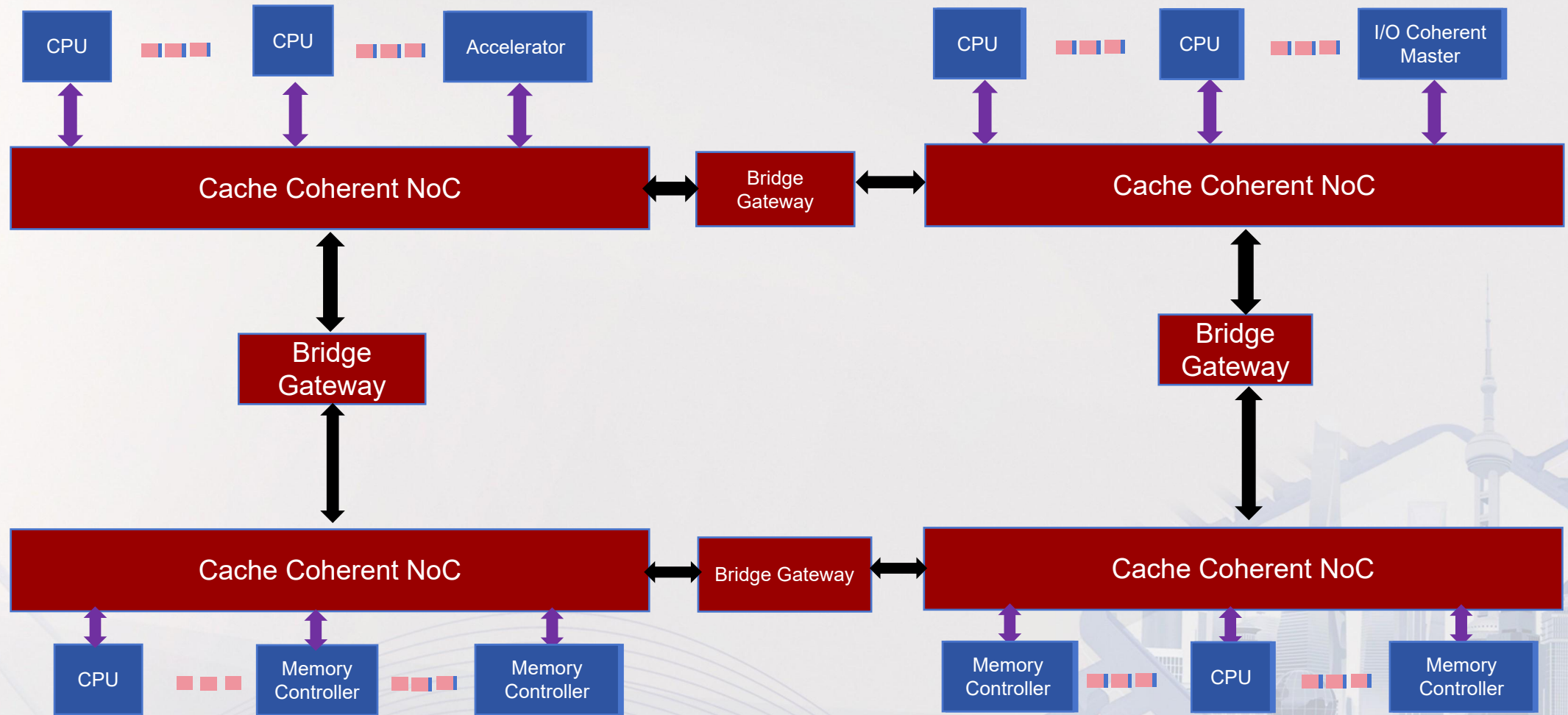


Agenda

- 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



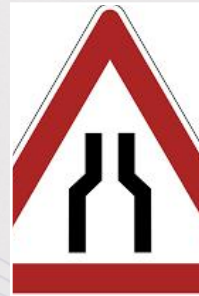
System Level Functional Scenarios



System Coherency



Smart Debug

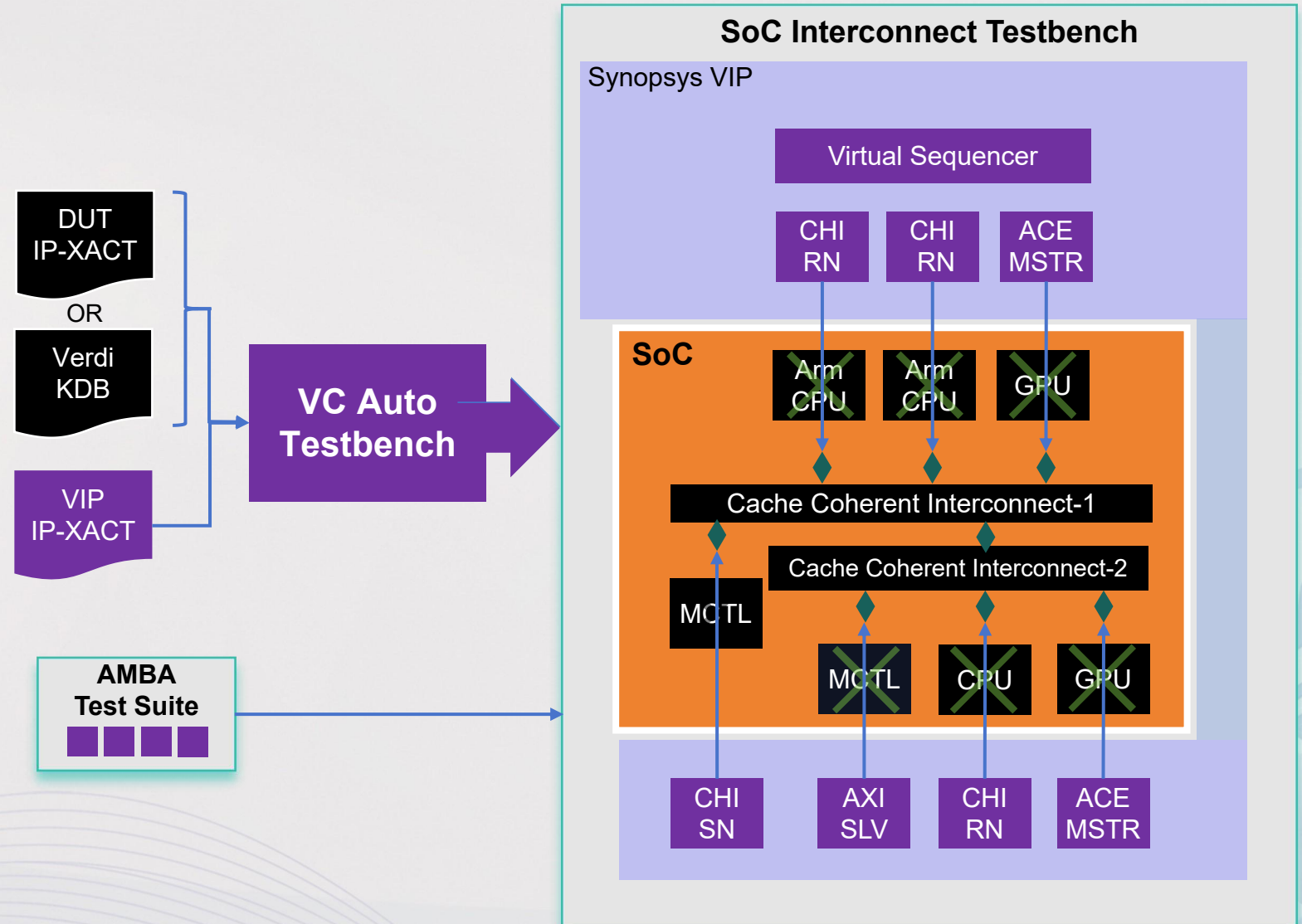


Performance Bottleneck

Scalability : Automated SoC Testbench Generation

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

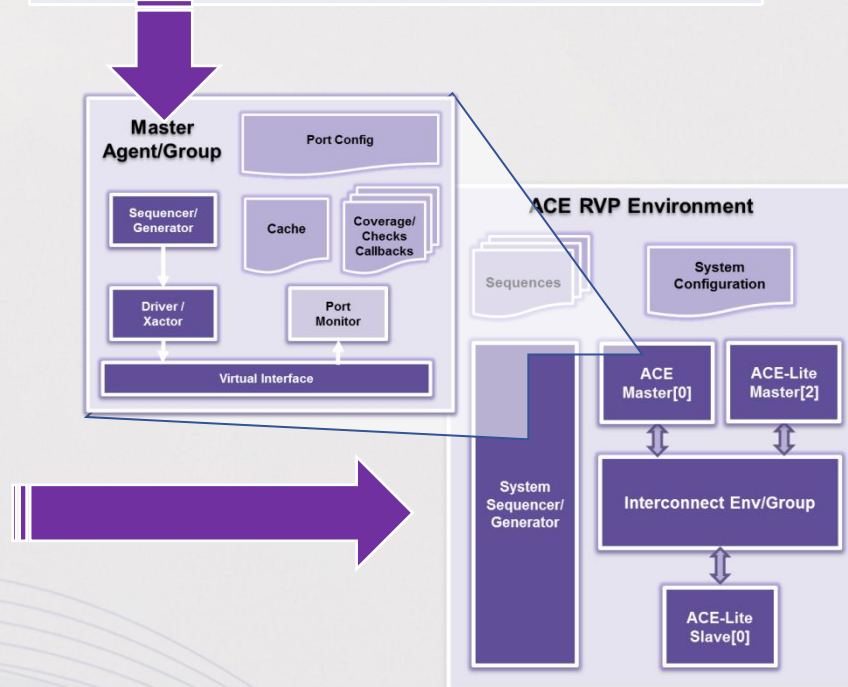
- SoC environment generation reading the DUT topology
- Both passive & active RTL replacement



Predefined Scenarios for Coherency Verification

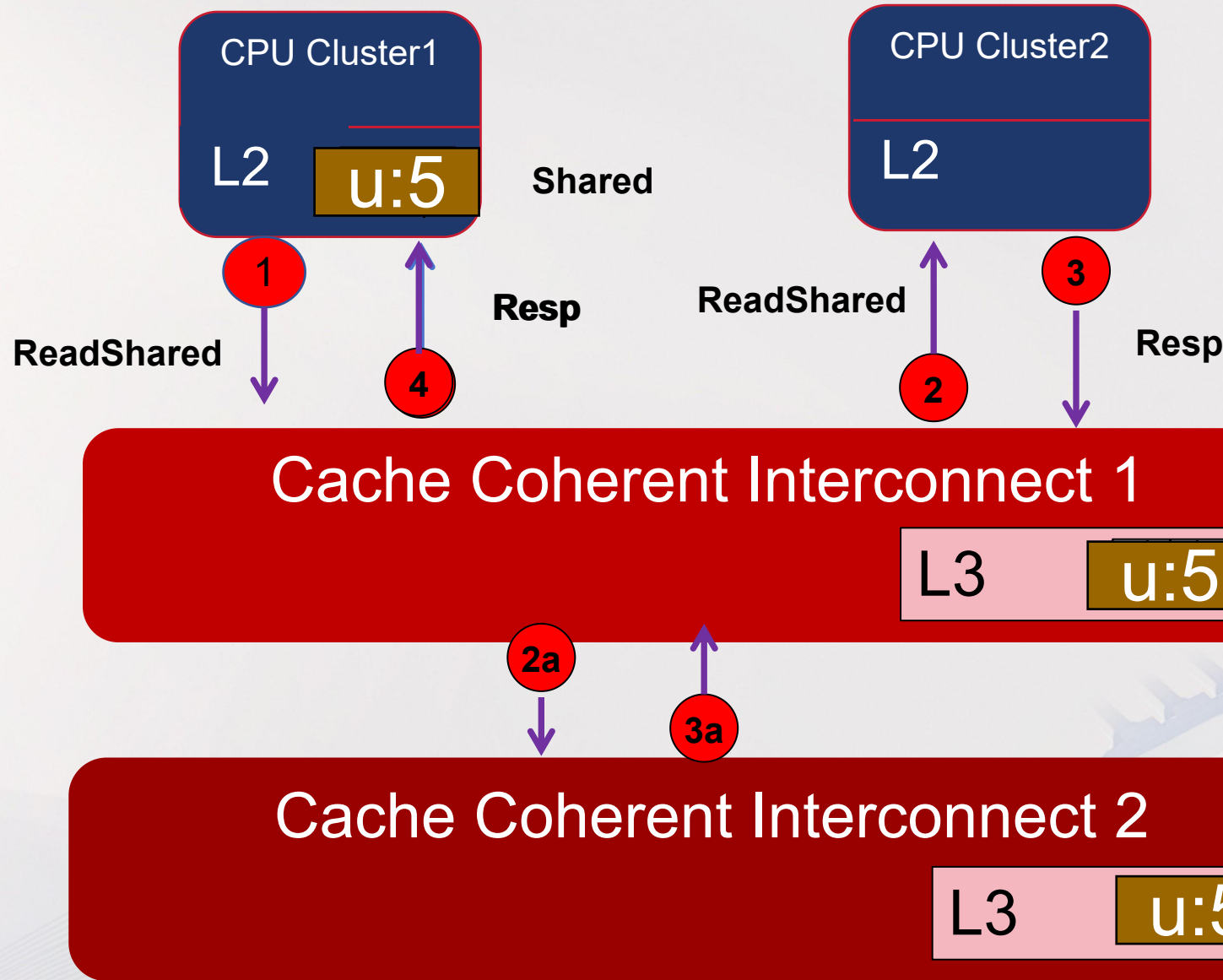
Transaction type	Pre-defined Coherent Sequences
Non-snooping	ReadNoSnp
	WriteNoSnp
Coherent	ReadClean
	ReadNotSharedDirty
	ReadShared
	ReadUnique
	CleanUnique
	MakeUnique
	ReadOnce
	WriteUnique
	WriteLineUnique
Memory update	WriteBack
	WriteClean
	WriteEvict
	Evict
Cache Maintenance	CleanShared
	CleanInvalid
	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

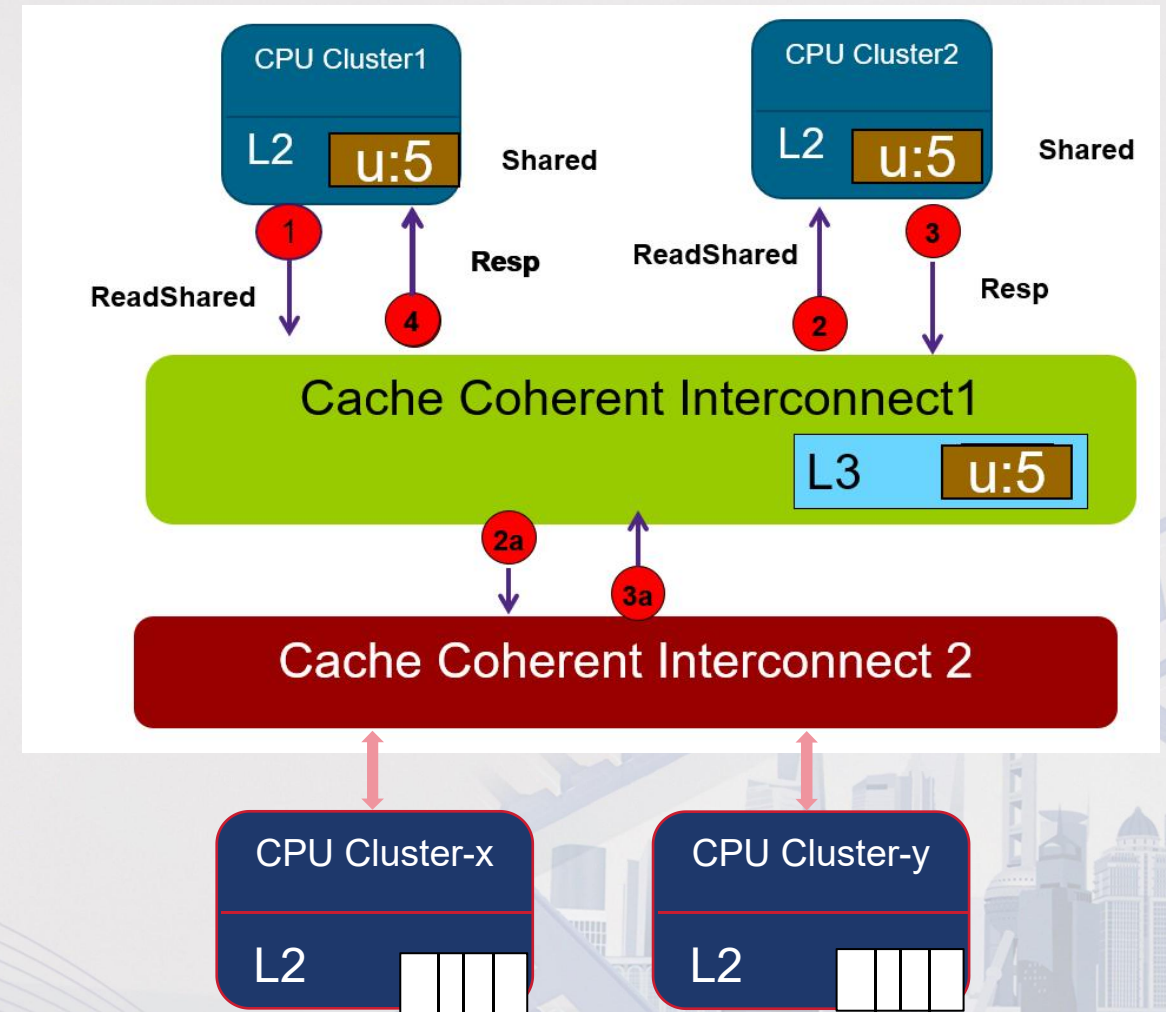
Coherency across master cache
between chips

Issuing Snoop transactions

Sequencing transactions

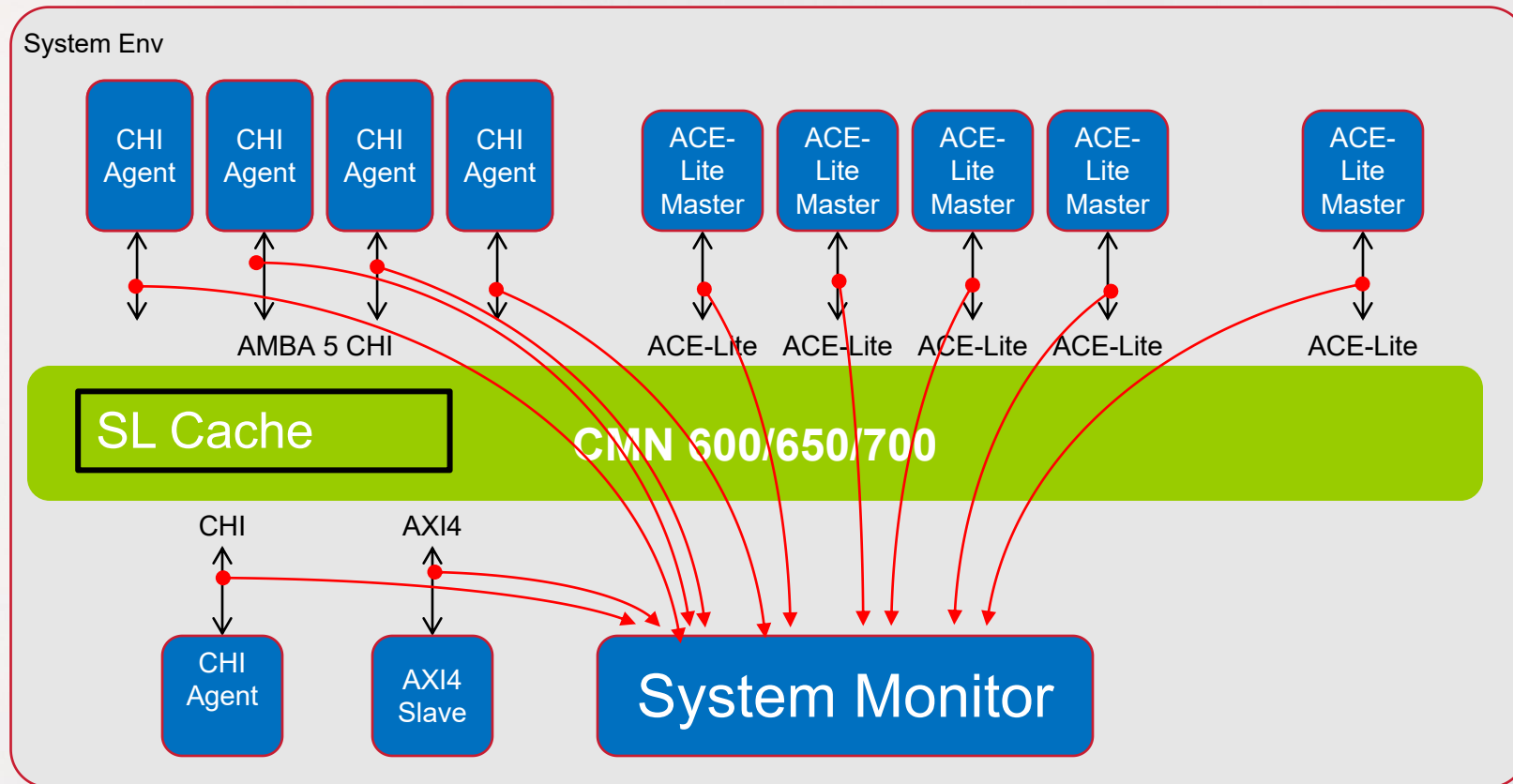
Data integrity between snoop
and coherent transactions

Coherency across master cache
and Interconnect cache



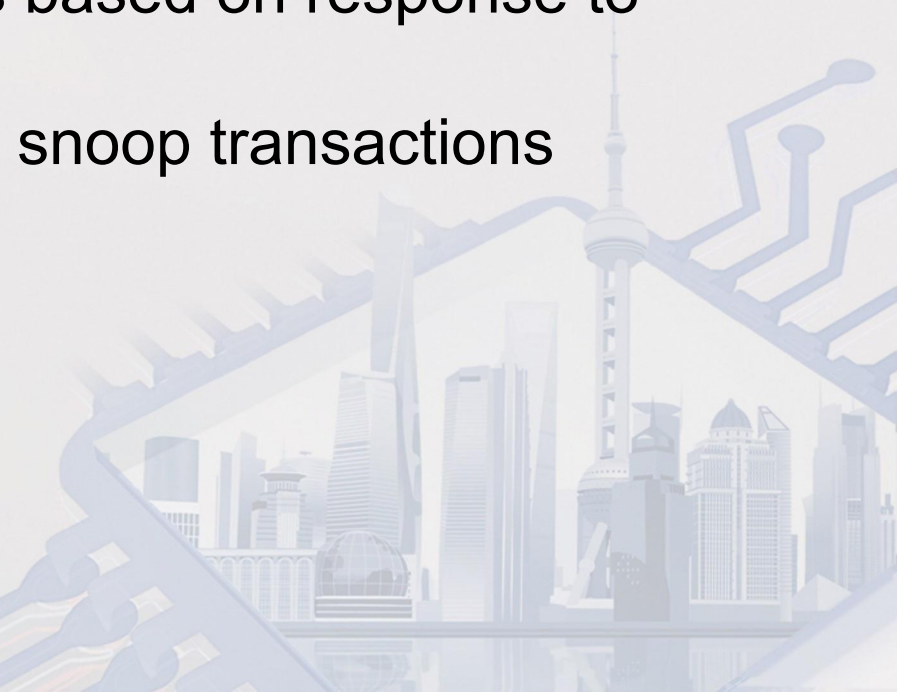
Single Chip -CHI System Monitor

Perform Check across CMN600/650/700



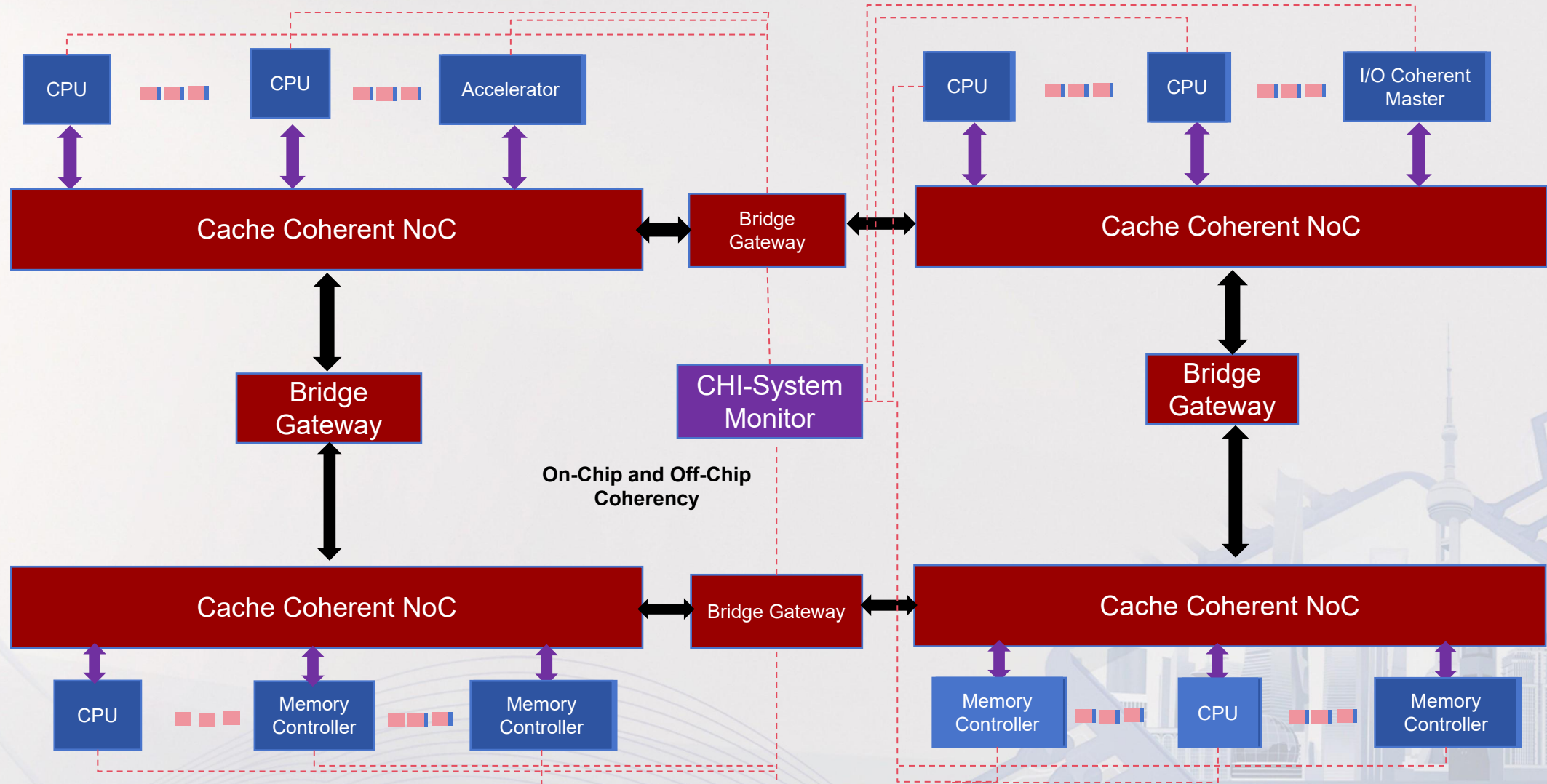
Single Chip CHI System Monitor

- 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



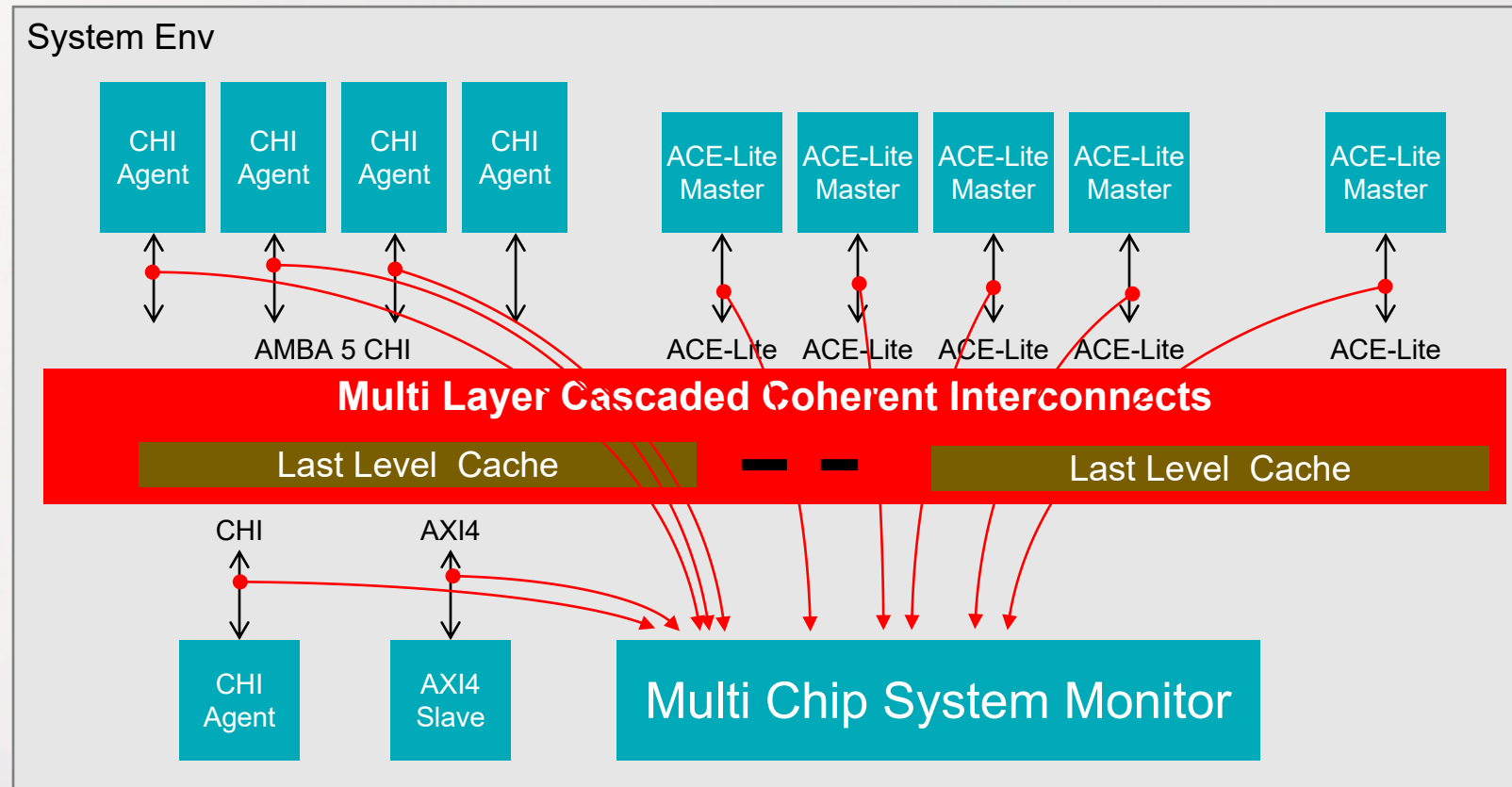
System Coherency Challenge across Chips

Cascaded Coherent interconnects across multiple chips



Multi-Chip Coherent System Monitor

Performs System Level checks across interconnect ports

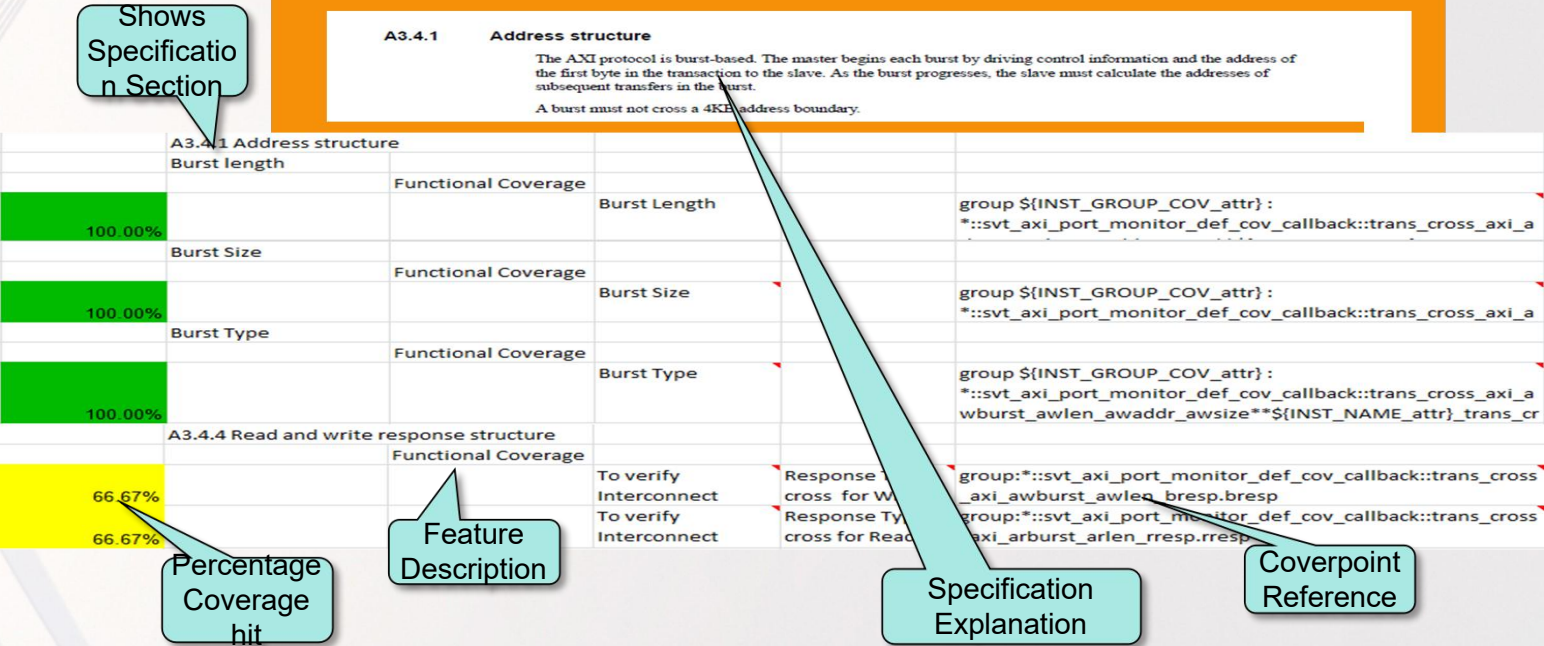


Multi Chip CHI System Monitor

- 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

Cc uvm_test_top.env.amba_system_env_0.chi_system[0].trans_chi_e_num_outstanding_xacts_from_diff_src_id_wrt_src_id_of_current_txn_which_received_dbidrespd	0.00%
Cc outstanding_xacts_from_diff_src_wrt_current_txn_which_received_dbidrespd	100.00%
Cc svt_amba_uvm_pkg::svt_chi_system_monitor_issue_e_def_cov_callback::trans_chi_e_snp_xacts_to_other_rn_with_mismatch_ns_bit_when_rn_xact_received_dbidrespd_resp	0.00%
Cc svt_amba_uvm_pkg::svt_chi_system_monitor_issue_e_def_cov_callback::trans_chi_rn_f_ports_chi_rn_f_ports_concurrent_non_overlapping_chi_e_rn_f_xacts_with_chi_rn_f_xacts	97.02% 97.0
Cc svt_amba_uvm_pkg::svt_chi_system_monitor_issue_e_def_cov_callback::trans_chi_rn_f_ports_chi_rn_f_ports_concurrent_overlapping_chi_e_rn_f_xacts_with_chi_rn_f_atomic_xacts	98.61% 98.6
Cc svt_amba_uvm_pkg::svt_chi_system_monitor_issue_e_def_cov_callback::trans_chi_rn_f_ports_chi_rn_f_ports_concurrent_overlapping_chi_e_rn_f_xacts_with_chi_rn_f_xacts	93.45% 93.4
Cc svt_amba_uvm_pkg::svt_chi_system_monitor_issue_e_def_cov_callback::trans_chi_rn_f_ports_chi_rn_f_ports_concurrent_non_overlapping_chi_e_rn_f_xacts_with_chi_rn_f_xacts	56.25% 56.2

Quickly Identify the Source of Bugs

Verdi Protocol Analyzer and Memory Protocol Analyzer

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

Waveform Viewer

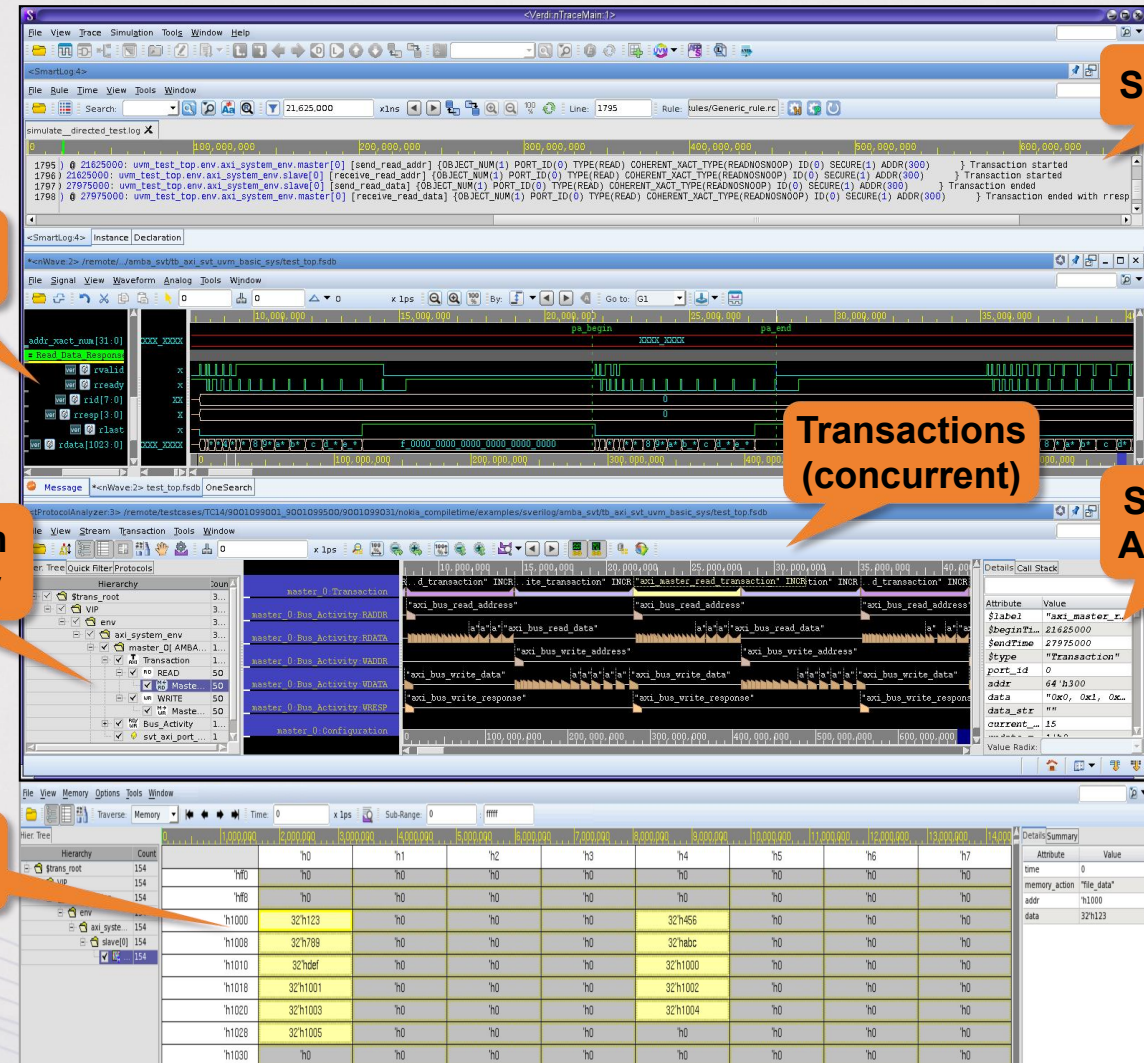
Testbench Hierarchy

Selected Address

SmartLog

Transactions (concurrent)

Selection Attributes



Smart Debug using Protocol Analyzer

ambas_system_env

8,585

axi_system_0

221

axi_system_1

18

chi_system_0

8,346

chi_system_monitor[AMBA_S...

2,978

rn_0[AMBA_SVT_CHI_SVT]

1,557

rn_1[AMBA_SVT_CHI_SVT]

2,758

Observed_Link_FSM

4

cache

797

svt_chi_link_fsm_activate_...

2

svt_chi_link_fsm_stop_state

2

svt_chi_protocol_dat_flit

436

Rx_DAT_VC

34

Tx_DAT_VC

402

svt_chi_protocol_req_flit

441

REQ_VC

441

svt_chi_request_wr_writeuniquefull transaction

Set Search Attribute (on vgvip1)

Search Type

Any Transaction

By Attribute

By Attributes

Attribute

Function

Description

Operator

byte_enable

has_attr...

Check whether transaction has this attri

[]

byte_enable

has_relation

Check whether transaction has parent/c

[[]]

cache_line_count

has_tag

Check whether transaction has tag. ex:

-

cache_line_size

is_unfinish

Check whether transaction is unfinished

'

cache_stashing_enable

has_error

Check whether transaction has UVM_ERI

-

cachelinesize_aligned_addr

!

cfg_lcrd_delays_enable

~

channel

&

check_addr_overlap

~&

comp_to_dbid_flit_delay

|

compact_follows_readreceipt

~|

Expression: (Time unit is "ps")

(addr>=64'h800 && addr<=64'h83f) && (\$beginTime>259800 && 317400)

Import

Clear

Coherent to Snoop Association error

19 REQ FLIT REQ_Flit [339763763242 - 339763763242]

20 CMO XACT CMO_Clean_Invalid [339763763242 - 339820264424]

21 REQ FLIT REQ_Flit [339778263958 - 339778263958]

22 WR XACT WR_No_SNP_Full [339778263958 - 339875763672]

23 RD XACT SnpMakeInvalid [339792264540 - 339798765448]

24 SNP FLIT SNP_Flit [339792264540 - 339792264540]

25 operations [339792264540 - 339792264540]

26 REQ FLIT REQ_Flit [339858265174 - 339858265174]

Identifying overlapping transactions

Relations

1 svt_chi_request_readnosnp_transaction [184750033896 - 197587964550]

2 REQ_VC [184750033896 - 184750033896]

3 Tx_RSP_VC [187906585622 - 187906585622]

4 svt_chi_request_wr_writenosnpptl_transaction [195483137290 - 195510032226]

5 REQ_VC [195483137290 - 195483137290]

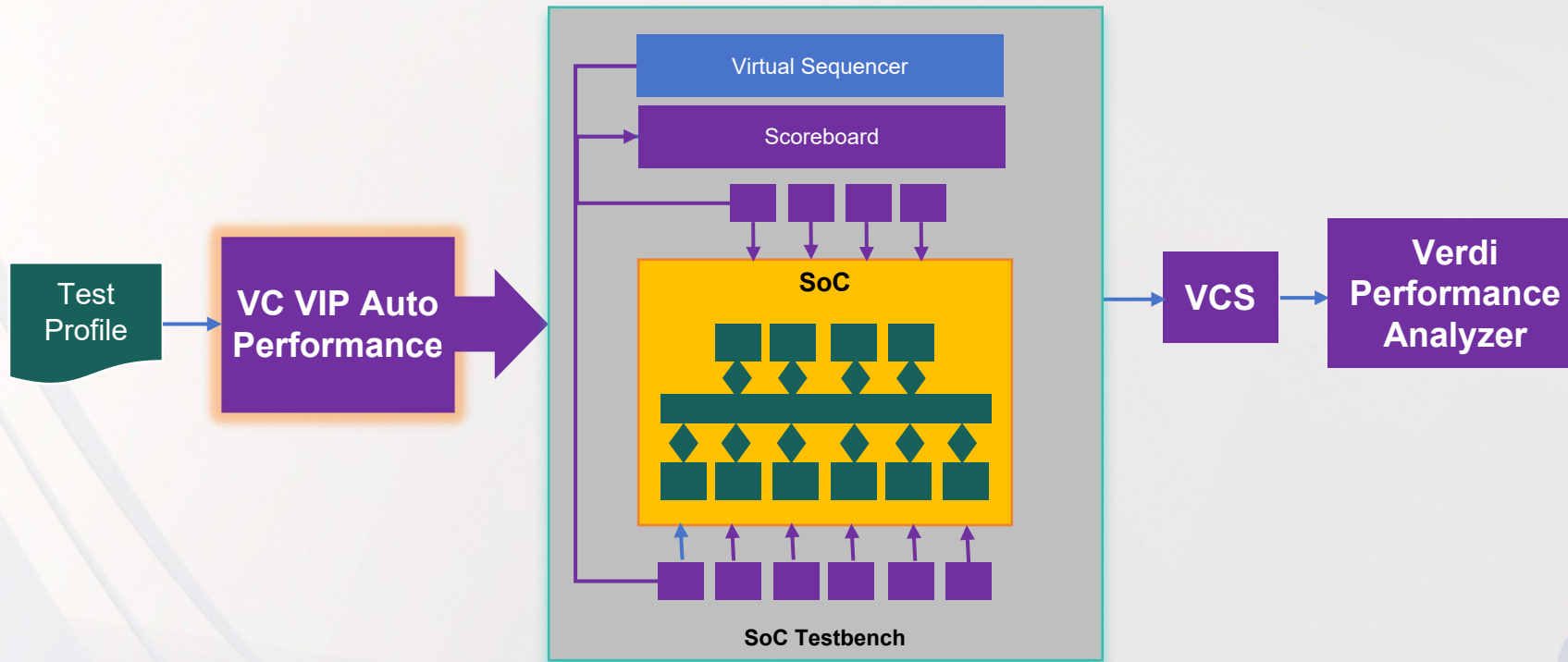
6 Tx_DAT_VC [197566585664 - 197566585664]

7 Tx_DAT_VC [197587964550 - 197587964550]

Advance Search criteria

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

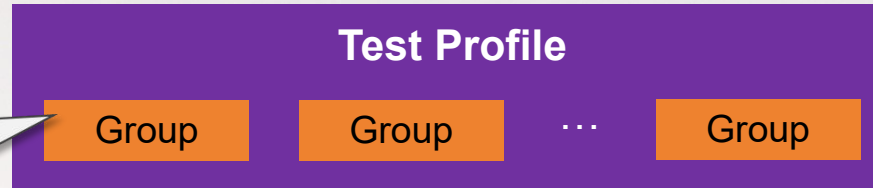
AMBA VIP built-in traffic rate adapter & arbiter drive profile on DUT

User analyzes performance simulation results with Verdi Performance Analyzer

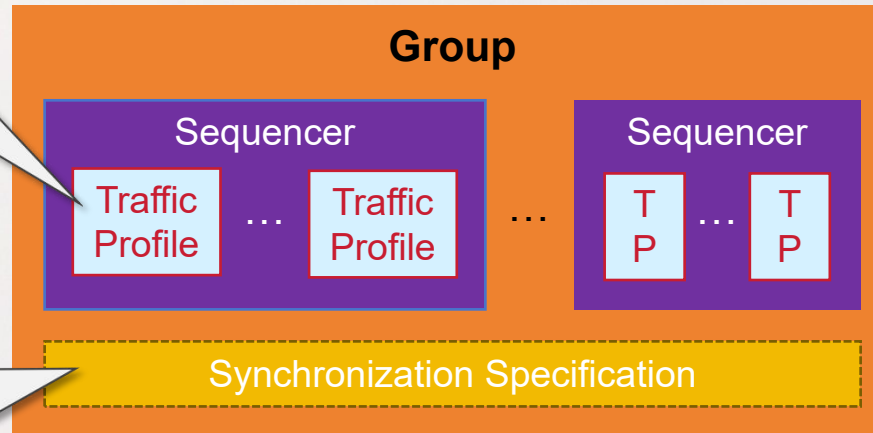
Defining a Performance Test

VC VIP AutoPerformance

Each **Group** is executed sequentially

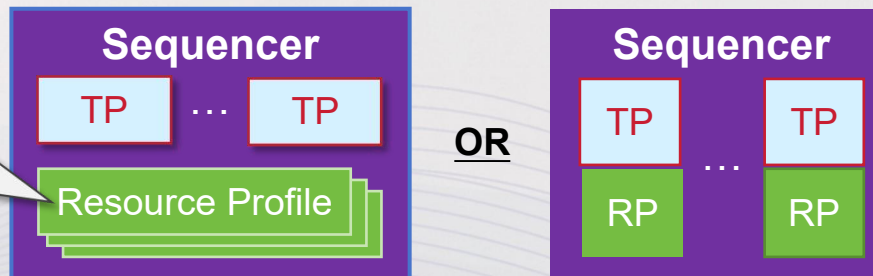


The **Traffic Profiles** in a given **Group** are executed concurrently



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"
    </axi>
  </master>
</traffic_profile>
```

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

Define
memory
writes

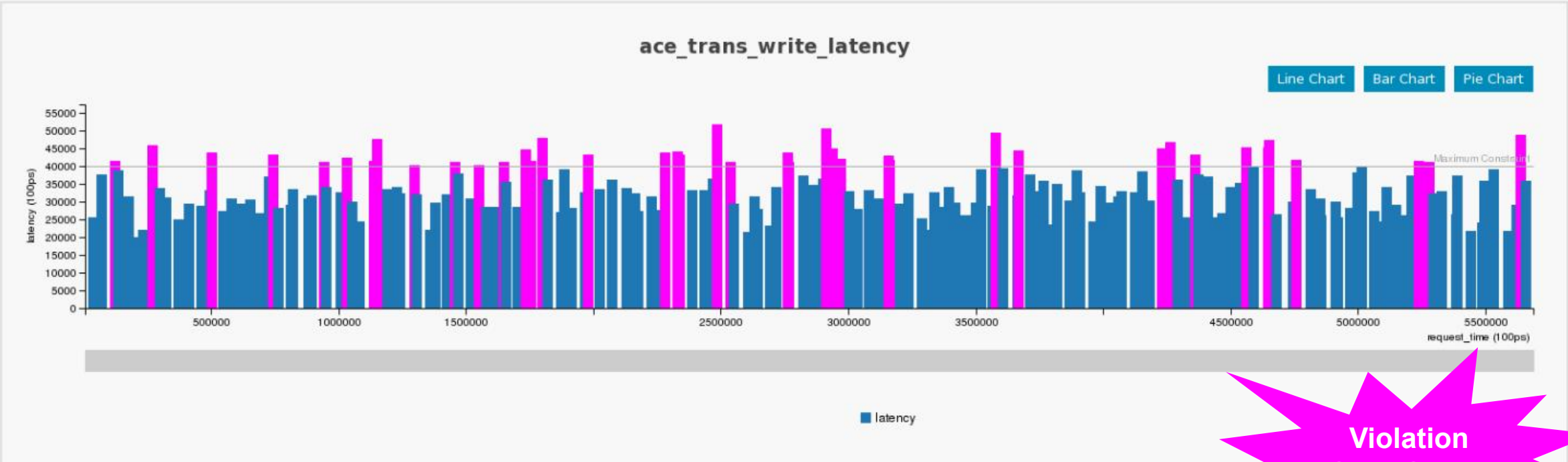
```
<group name="memory writes">
  <sequencer instance="uvm_test_top.env.amba_system_env_0.chi_system[0].rn[0].rn_xact_seqr"
    name = "rn_0_sequencer">
<traffic_profile path="mem_normal_writes_sequential_1_profile.xml"
    name="mem_normal_writes_sequential_1_profile"/>
  </sequencer>
  <sequencer 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>
```

Sequencers
for writes

Synchronizatio
n

Write Latency Observed at a Slave

This port of the Interconnect is connected to the Memory Controller



view tools window

From: To: x 100ps

Metrics Results

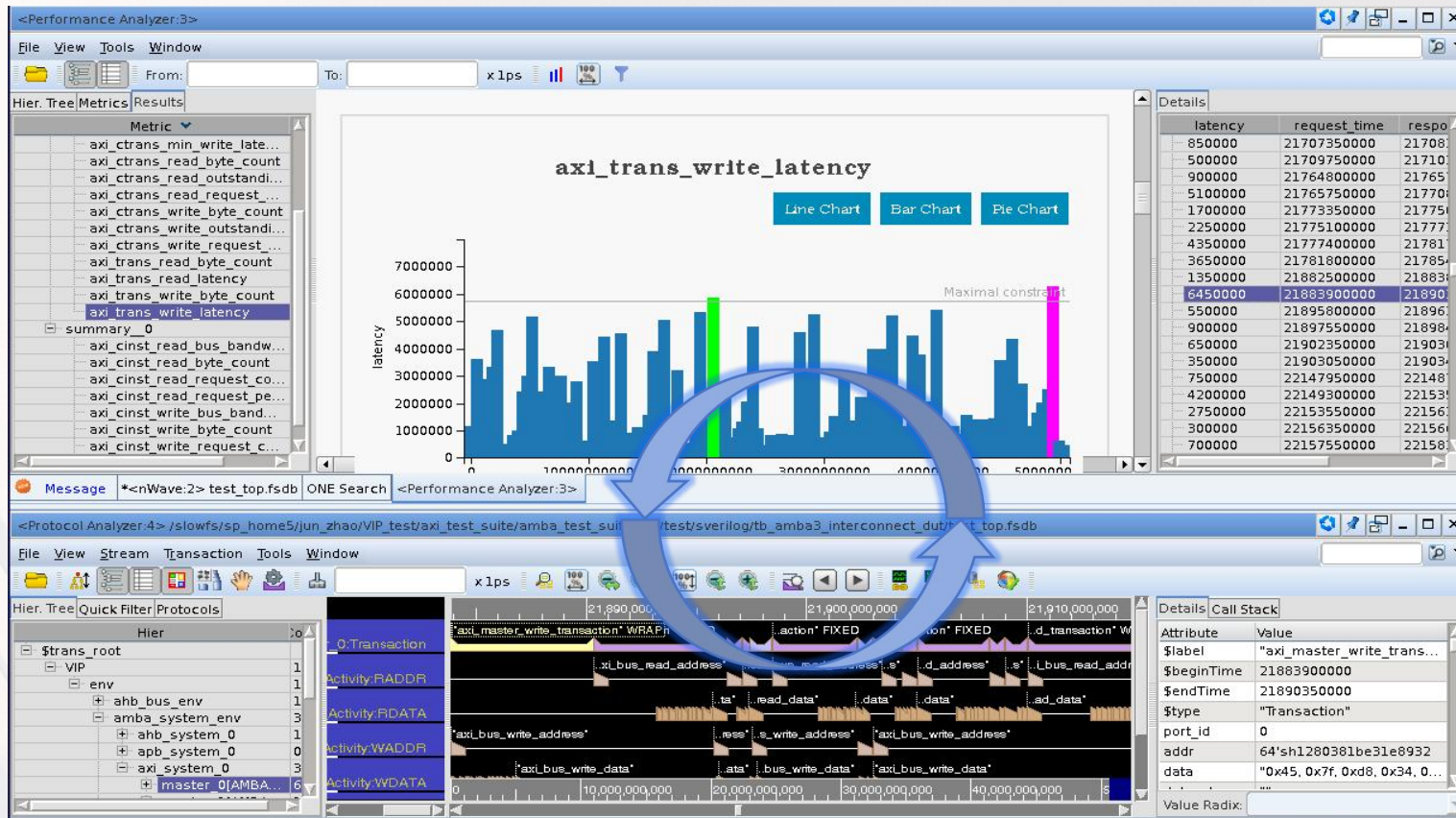
	Metric	Min	Max	Chart
☒	P ace_trans_write_blocking_ratio			bar
☒	P ace_trans_write_byte_count			bar
☒	P ace_trans_write_latency		40000	bar
☒	P ace_trans_writeback_latency			bar

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