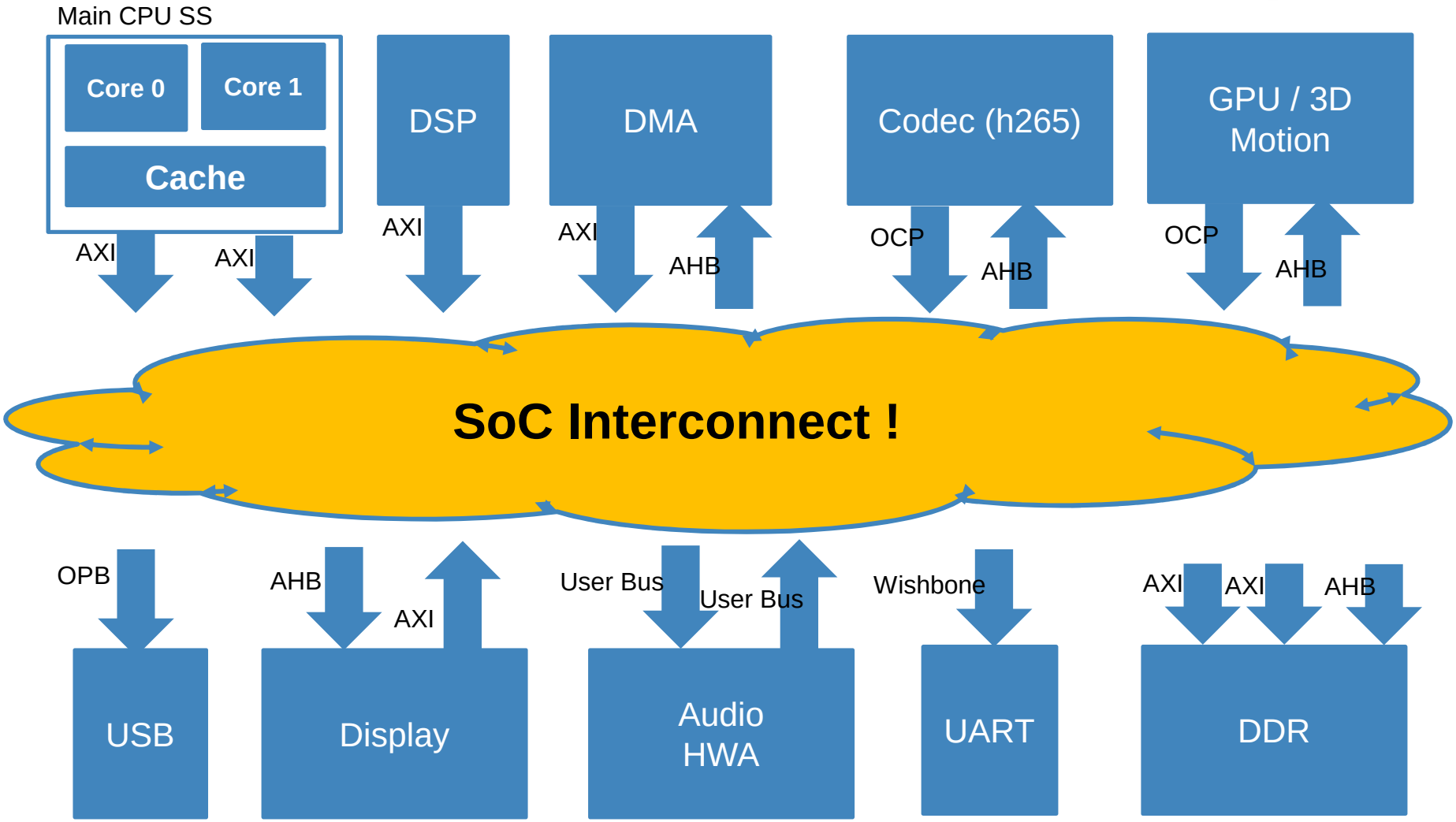
Interconnect Verification Challenge and the need for a generic scoreboard

François Cerisier
Verification Expert
Test and Verification Solutions France

Verification Futures 2013

SoC Interconnect



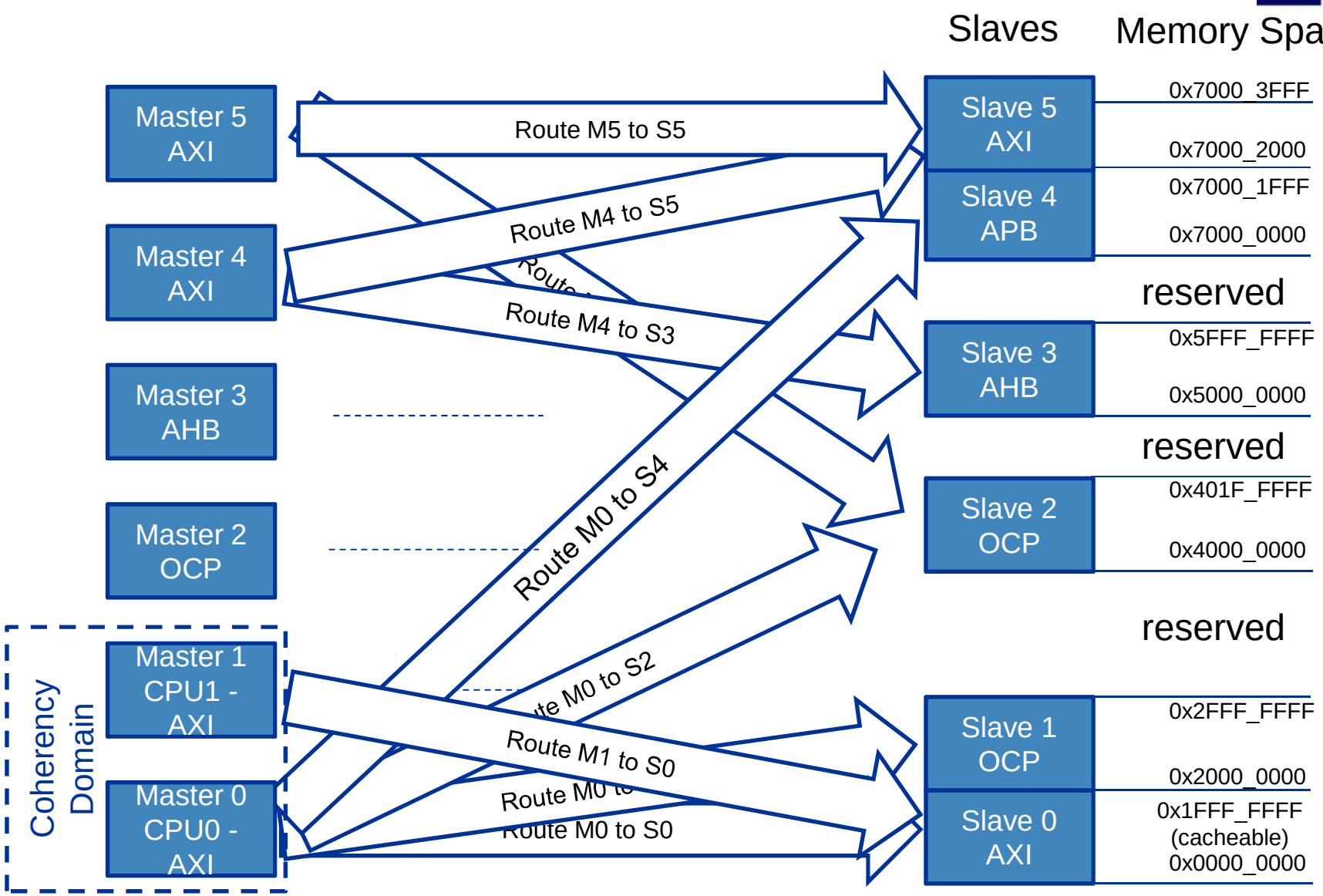
Main Interconnect Characteristics

- **Communication of transactions between Masters and Slaves**
  - Protocols (AXI, AHB, APB, OCP, PLB, OPB, DCR, Wishbone, company corporate bus, …)
  - Bus Widths (8/16/32/64/128)
- **Memory Maps**
  - Shared memory map for all masters ?
  - Memory map clusters ?
  - One Memory map per master ?
- **Address Space**
  - Physical Address Space
  - Virtual Address Space (MMU)
  - System Virtual Memory Address Space (System MMU)
- **Cache Coherency**
  - ACE, ACE-Lite ?
  - Others ?
- **SoC specific features**
  - Dynamic re-configuration (address space, security, routes, …)
  - Error management
    → invalid requests, unmapped addresses
  - Security, DRM, Firewall
    → invalid request depending on security attributes, software execution level
  - Power management, Master/Slave power up/down
    → interrupts, invalid request

The Interconnect Verification Plan

- **Address Map**
  - Are all masters able to access all possible slaves ?
  - … under virtual address mode ?
  - Errors on invalid addresses
- **Protocol Sanity**
  - Are all kinds of transactions supported on each route ?
  - Are bursts/locks supported on each route ?
  - Protocol not broken under stress conditions
- **SoC features**
  - Security
    - Can secure transactions access to all slaves ?
    - Are unsecured transactions getting errors from secured slaves ?
  - Power Management Use cases
    - Are we getting error from power off slaves ?
    - Are we able to wake up a slave ?
  - Cache Coherency
- **Performance Analysis**
  - Latency, Bandwidth
- **Interconnect integration**
  - Are the different CPU clusters and IPs well connected with the interconnect ?
- **Use cases**

Routes & Address Map



Protocol conversion issues

AXI transaction

Request transfer

AXI burst len=3
Size = WORD
Address = 0x5
Kind = WRAPED
LOAD

Response transfer

7	6	5	4	3	2	1	0
F	E	D	C	B	A	9	8
17	16	15	14	13	12	11	10

Possible Converted transaction in protocol X

On a 64 bit bus

Request transfer

LD16
Addr = 0

Request transfer

LD8
Addr = 0x10

Response transfer

7	6	5	4	3	2	1	0
F	E	D	C	B	A	9	8

Response transfer

17	16	15	14	13	12	11	10

On a 32 bit bus

Request transfer

LD16
Addr = 0

Request transfer

LD8
Addr = 0x10

3	2	1	0
7	6	5	4
B	A	9	8
F	E	D	C

13	12	11	10
17	16	15	14

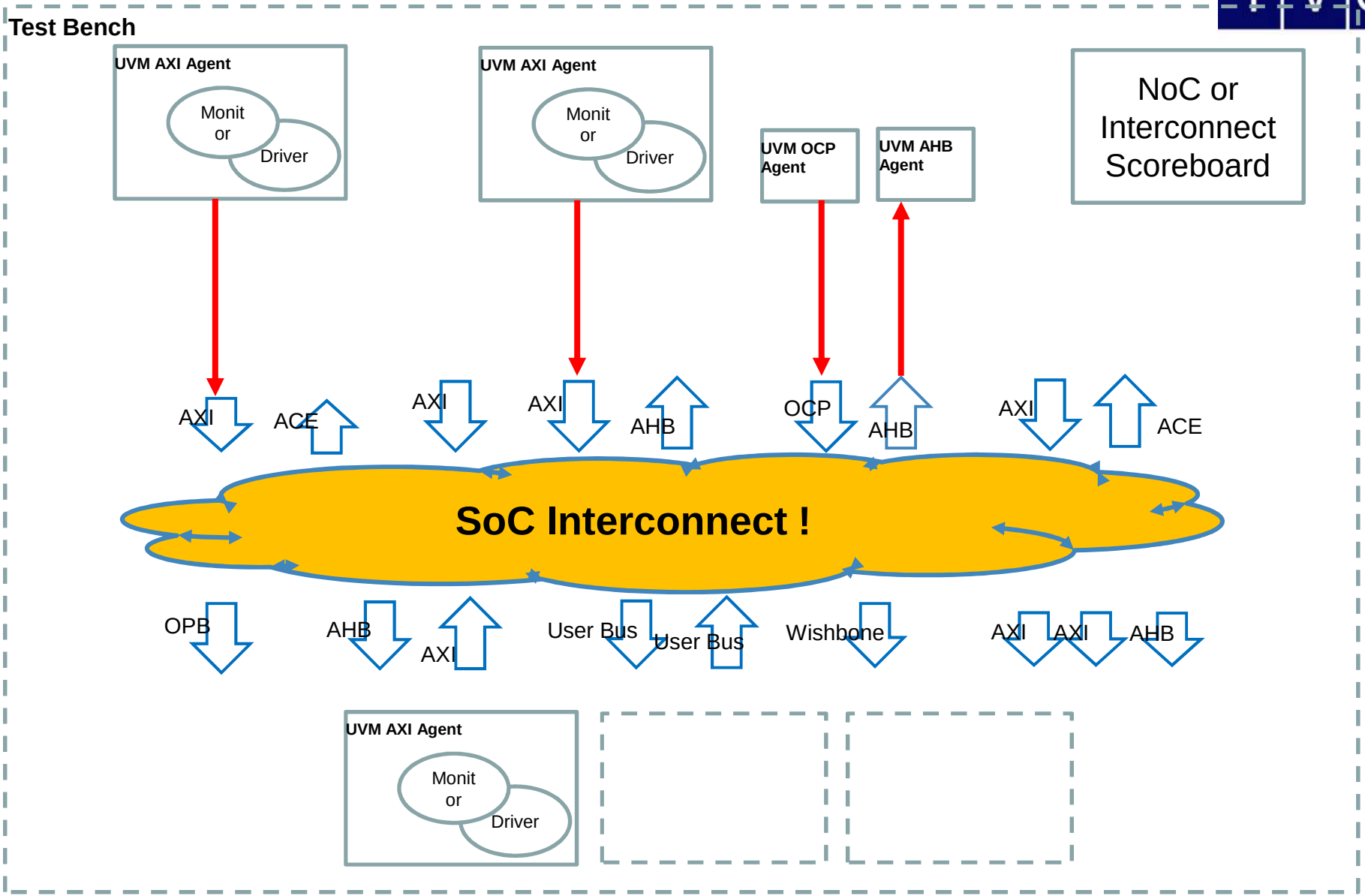
Testbench Requirements

- **VIPs**
  - For each protocol (master and slaves)
  - With good protocol chekers
  - With full transation monitoring sent to analysis port
- **Virtual Sequences**
  - Enable controlling all masters from one main sequence
  - Enable complex scenarios
- **NoC / Interconnect Scoreboard**
  - End to End Transaction Checking
- **Standalone or within SoC**

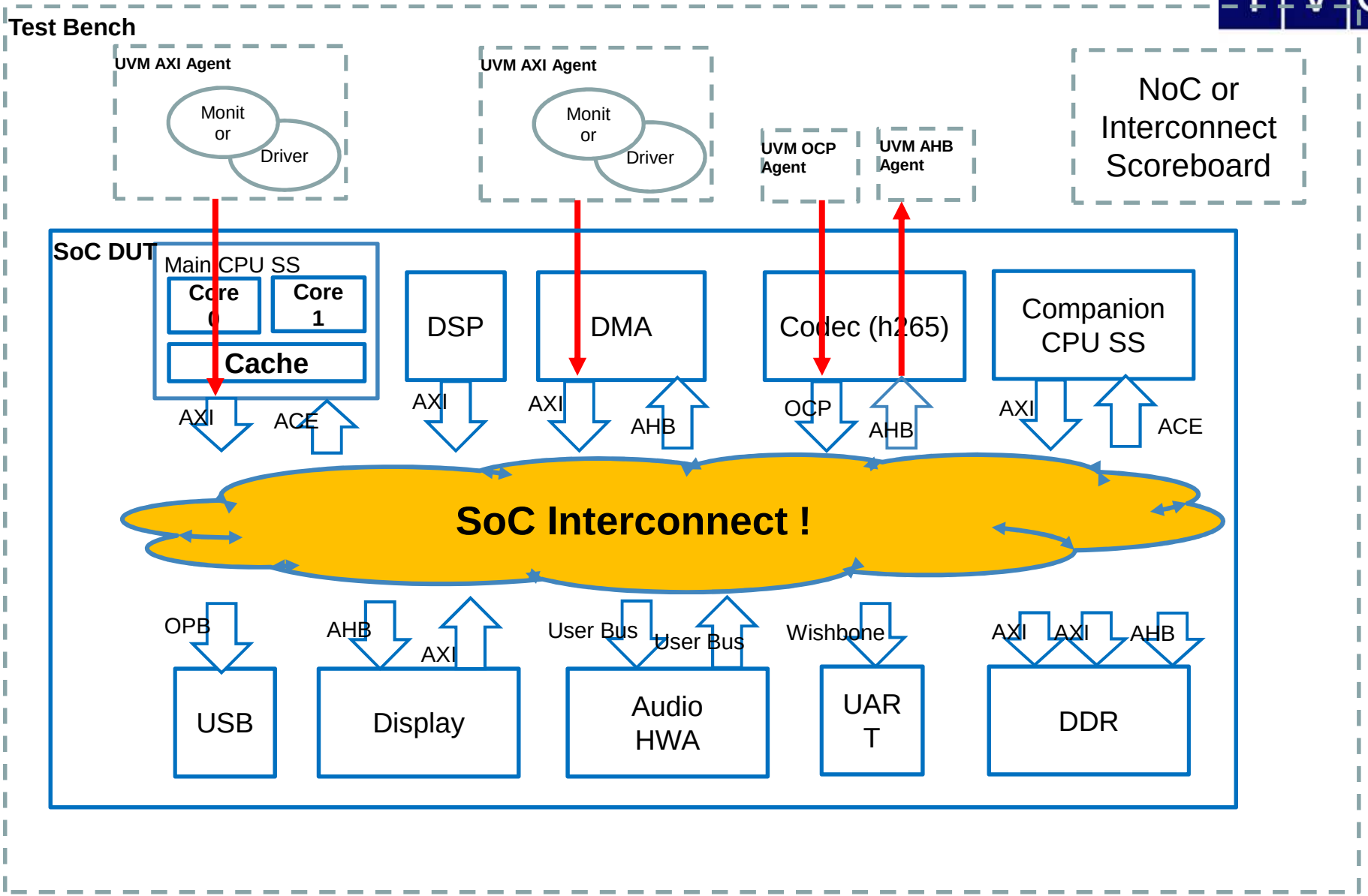
Standalone Testbench or with the SoC

- **Standalone**
  - Interconnect available before integration in the SoC
  - Only verify the interconnect out of context
- **Headless Within Integrated SoC**
  - Allows to find integration issues
  - Requires to black box all IPs

Interconnect TestBench - standalone



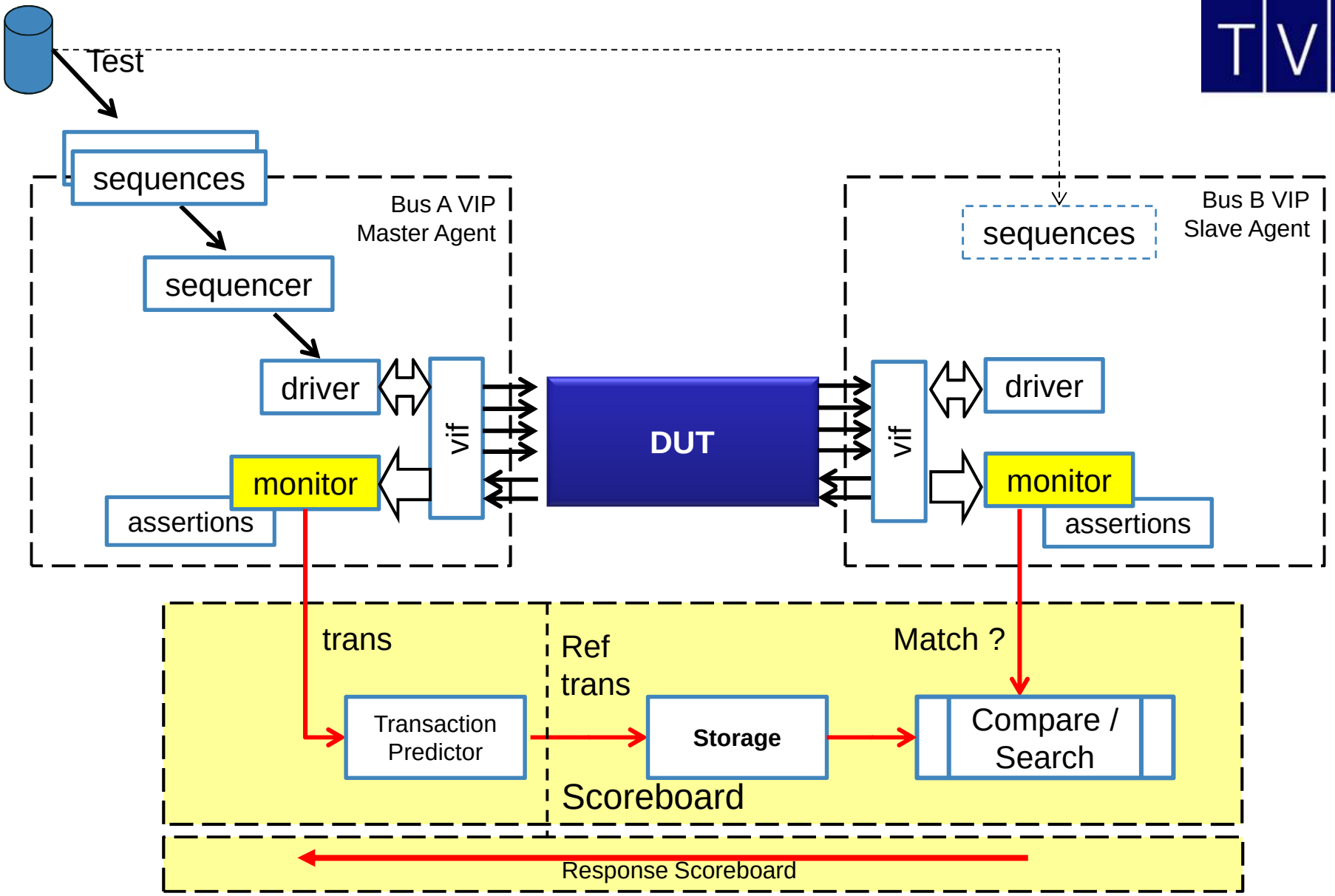
Interconnect TestBench in headless SoC context



Interconnect Scoreboard Requirements

- **Connect to any bus protocol VIP**
- **End to End transaction checking**
  - Data, direction, attributes, response, atomicity
- **Support for:**
  - Multiple address maps, Virtual address space
  - Address map reconfiguration, MMU
  - Security, Power management
  - Cache Coherency features (support for ACE)
  - User defined security/filtering (DRM, …)
- **Comparison policies**
  - Strict:
    - one to one transaction comparison
  - Permissive:
    - Allow transaction address realignment, dummy reads, nops
  - Per checker configuration
    - User switch on/off each checker (per path)

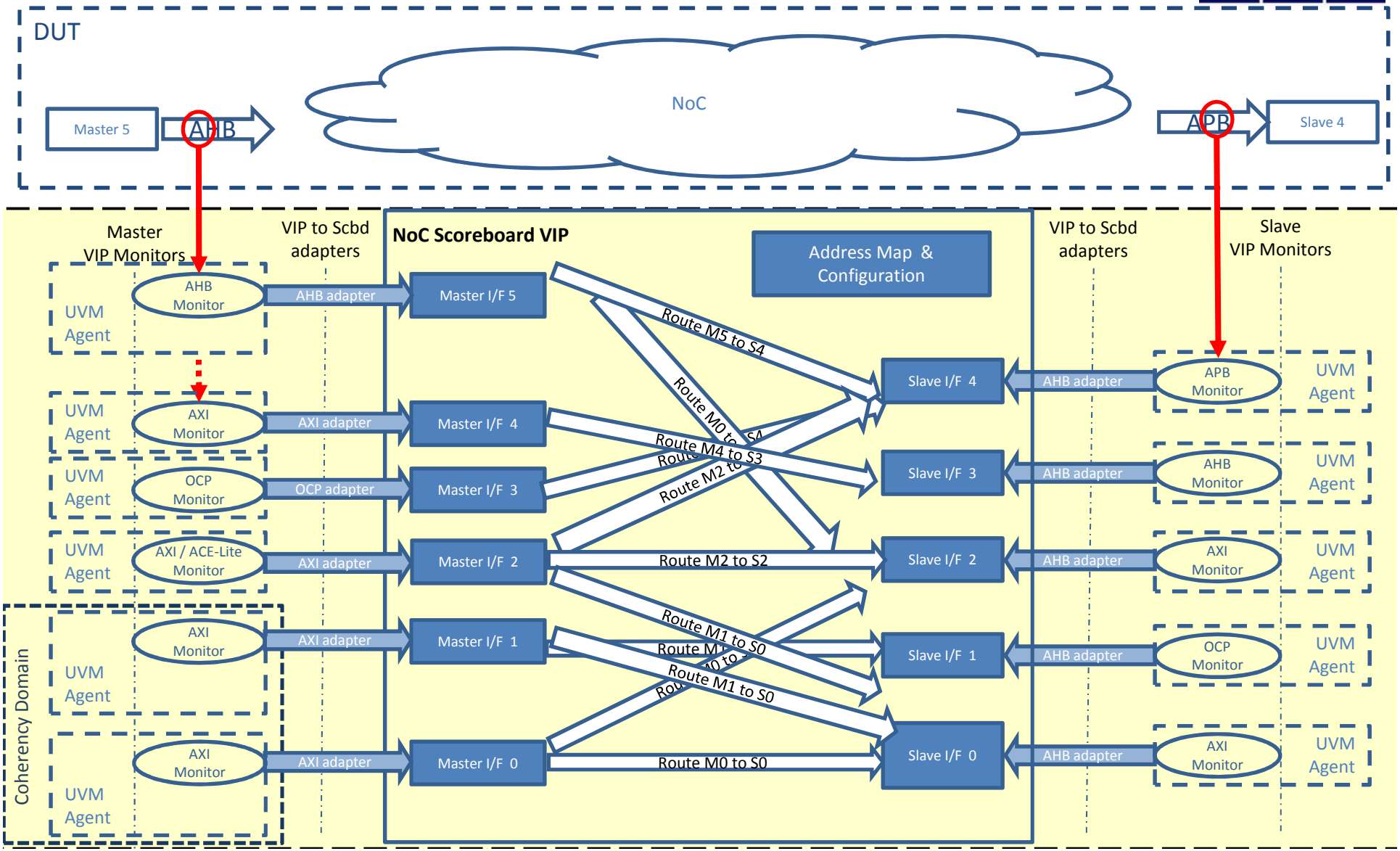
Simple Scoreboard principles – data flow design



Interconnect Scoreboard Structure

- **Internal Generic transaction**
  - Extendable with attributes using UVM factory
- **Adapter between VIPs and generic port**
- **One inner simpler scoreboard per route**
- **Reconfigurable Address Map table**

Interconnect Scoreboard Structure

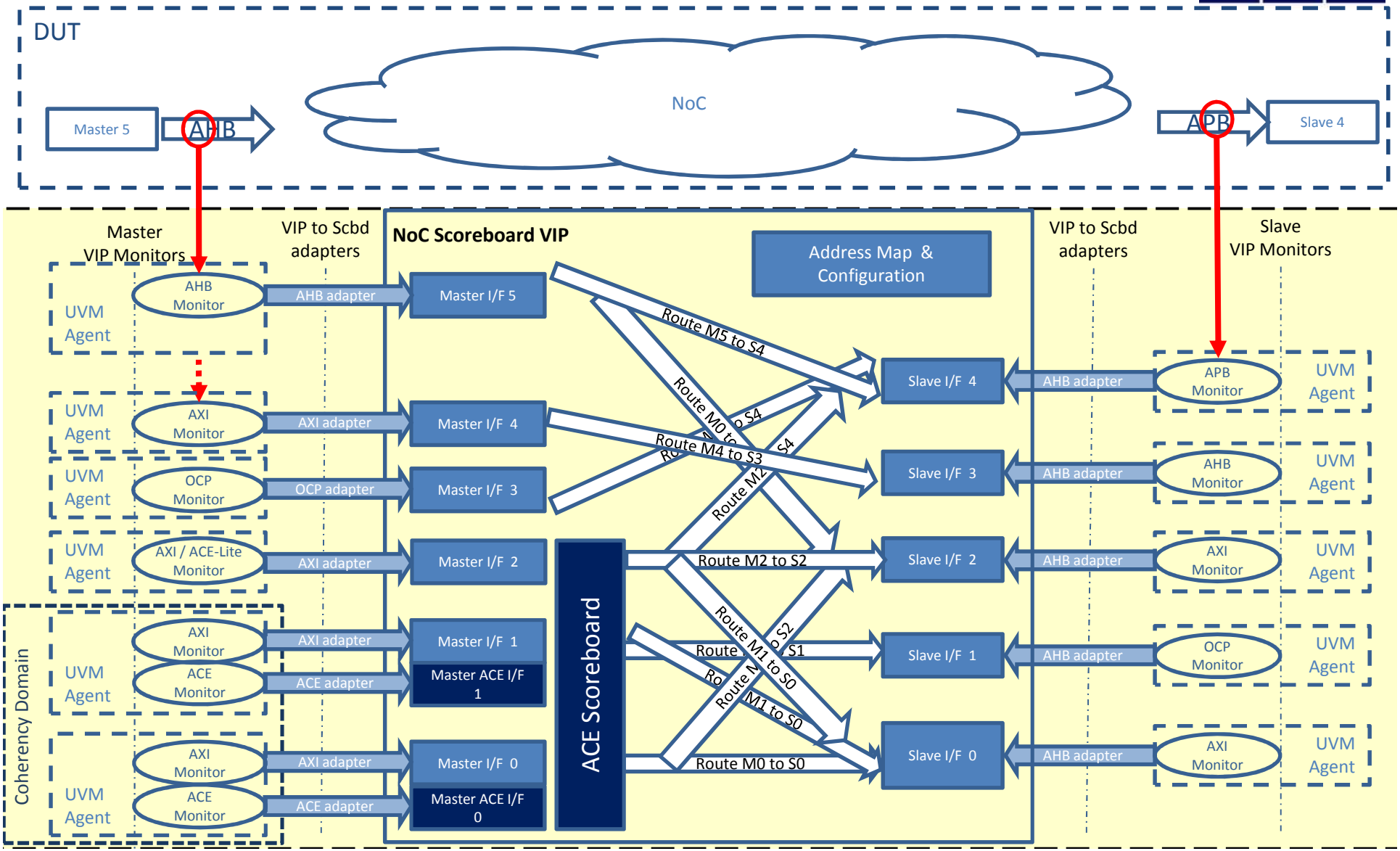


Adding Cache Coherency – ACE Support

- **ACE support**
  - Requests may not reach the targeted memory
  - Responses may come from another coherent master

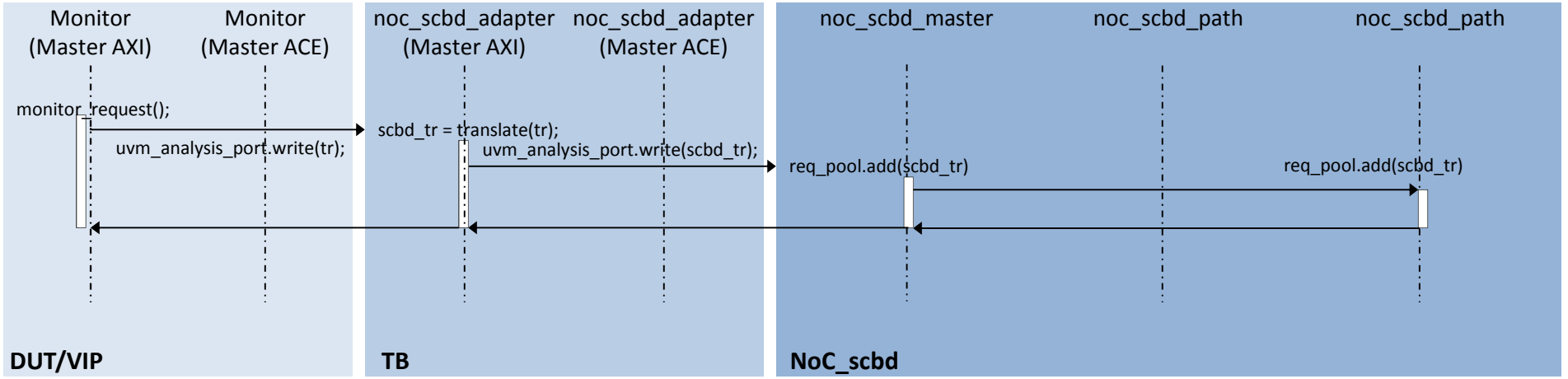
  ➔Breaks the general master/slave scoreboard principles

Interconnect Scoreboard Structure (with ACE)

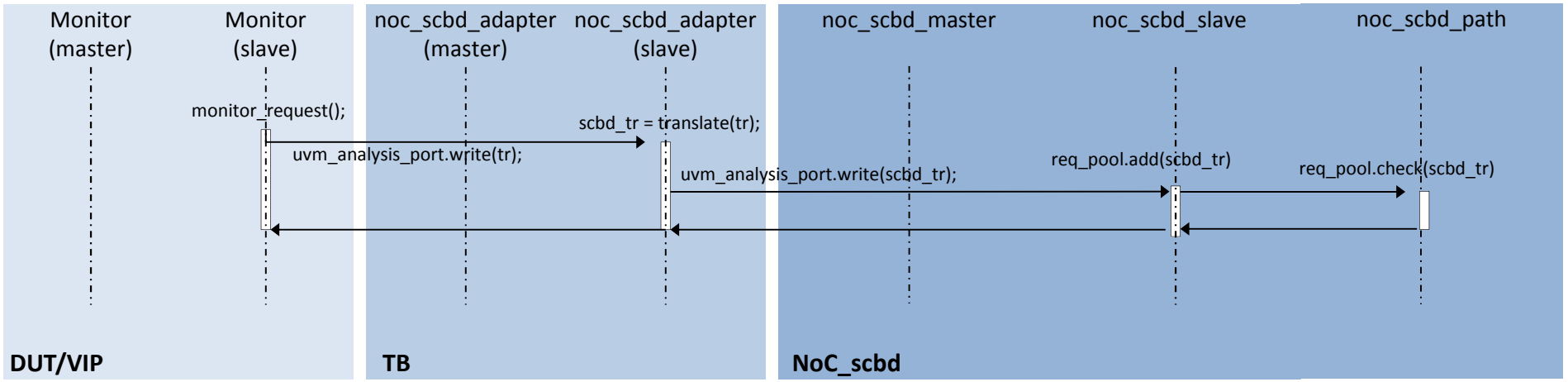


Scoreboard transaction flow – Request

Master



Slave



Scoreboard configuration example 1/2


```
// instantiate the scoreboard
scbd0 = noc_scbd::type_id::create("scbd0",this);

// create two address domains
scbd0.add_address_domain ("CPU0");
scbd0.add_address_domain ("CPU1");

// address domain 0 configuration
scbd0.add_address_segment("CPU0","","RAM",  'h0000_0000 , 'h2000000 , BUS);
scbd0.add_address_segment("CPU0","","UART", 'h9000_0000 , 'hFFFF , BUS);
scbd0.add_address_segment("CPU0","","ID0",  'hFFFF_FFF0 , 1 , BUS);

// address domain 1 configuration
scbd0.add_address_segment("CPU1","","RAM",  'h0000_0000 , 'h2000000 , BUS);
//...

//Masters declaration
scbd0.add_master("CPU0_instr", "CPU0");
scbd0.add_master("CPU0_data" , "CPU0");
//...

//Slaves declaration
scbd0.add_slave("ROM");
scbd0.add_slave("RAM");
scbd0.add_slave("UART");
//...
```

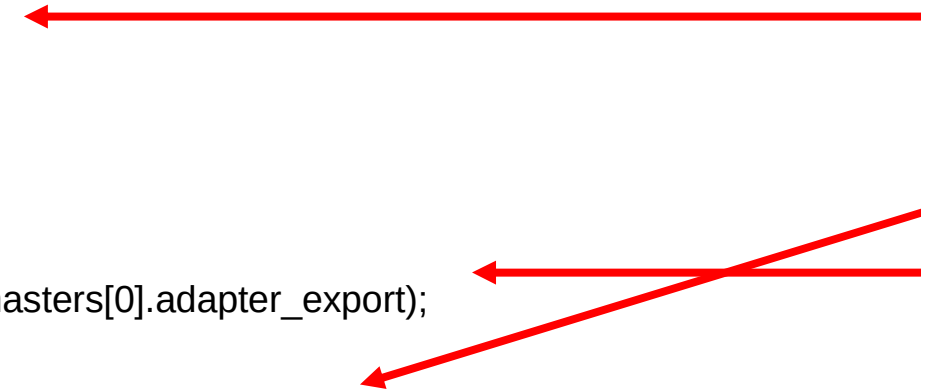
Scoreboard configuration example 2/2

```
//Paths declaration
scbd0.add_path("CPU0_data" , "ROM");
scbd0.add_path("CPU0_data" , "RAM");
scbd0.add_path("CPU0_data" , "UART");
//...

// Instantiate Adapaters
noc_scbd_axi_adapter CPU0_instr_adapter ;
CPU0_instr_adapter.noc_item.connect(scbd0.masters[0].adapter_export);

// connect monitors to scoreboard adapters
axi_env.agents[0].monitor.collected_request.connect(this.CPU0_instr_adapter.monitor_export_REQ);

//...
```



Advanced features

- **Protocol conversion to any VIP**
  - thru adaptors
- **Dynamic address map**
  - Thru configuration API
- **Attributes**
  - AXI attributes provided
  - Use the factory to extend the generic type
- **Security & Power Management**
  - Thru user defined methods
- **Functional Coverage**
  - path, address map, security, …
- **Performance**
  - Latency and bandwidth extraction available (requires proper use cases)
- **ACE & Cache Coherency**
  - Internally supported
- **Others**
  - Virtual method place holder can be user extended

User's experience

- **3 derivatives of a SoC Interconnect**
  - Approx 40 masters, 60 slaves with over 200 paths
  - 5 protocols, 3 different bus sizes
  - Security Management
  - Power Management features
  - Dynamic address translations
- **Scoreboard Developments**
  - Right architecture choice is key
  - Generic features / Generic Adapters
  - Search and comparison algorithms
- **Verification results**
  - Address map specification
  - Wrong protocol translations of AXI FIXED from 64 to 32 bit buses
  - Deadlock in some traffic congestions involving bursts
  - Deadlock in power management

Interconnect Verification Summary

- **SoC Interconnect needs to be verified from end to end**
- **Verification Environment should address**
  - Complex scenarios
  - Stress/congestion conditions
- **Interconnect SoC scoreboard should be generic & highly configurable**

Generic Scoreboard VIP summary

- **Allows quick testbench setup**
  - fully configured in minutes
- **Compatible with any VIPs**
  - Only requires proper monitor using uvm_analysis_port
- **Allow verification of security features, power management, ACE/Cache Coherency and more user defined features**

- **Questions?**