TrueConnect
Enriching Collaboration for Innovative Excellence
BENGALURU 2018

Effective Verification Techniques For System on Chip

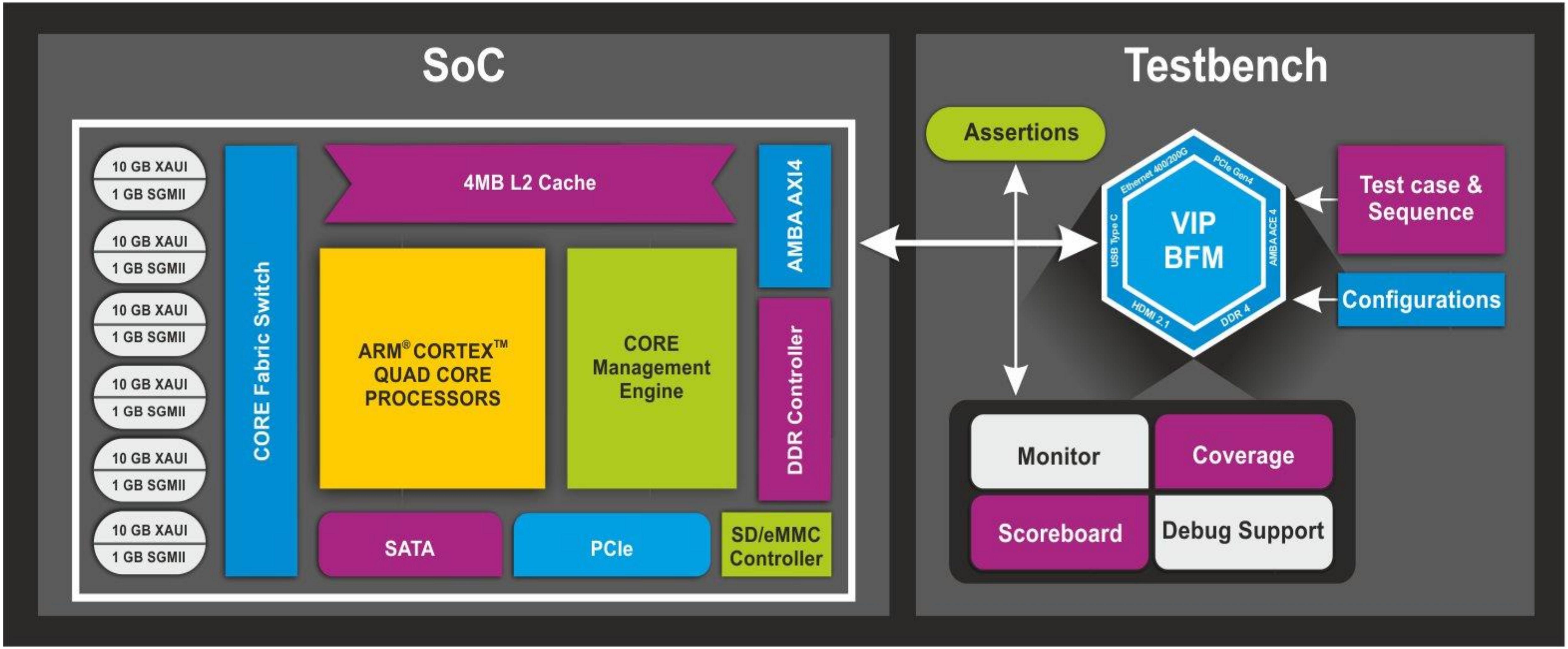
Agenda

- Introduction
- System On chip
- SoC Quality Verification
- Effective Techniques for Verification
  - Fully Random Environment
  - Using Stub Models for IP
  - Master AXI BFM replacing CPU
  - Initializing IP`s using BFM Driver at SOC level
  - Access to single/multi bus channels through single sequence model
- Q/A

Introduction

- SoC can be seen as integrating a Microcontroller or Microprocessor with other peripherals like USB, Ethernet, Memory etc. on to a single platform (chip).
- Around 60-70% of the time and money is spent on verification.
- Release product in shorten span of time with effective verification and zero bug.
- With advancement in technology, the complexity of the chip is increasing.
- Adding more advanced peripherals increase the complexity of the chip which directly affects the verification effort and time.

System On Chip (SOC) Verification Block



SoC

10 GB XAUI
1 GB SGMII
10 GB XAUI
1 GB SGMII
10 GB XAUI
1 GB SGMII
10 GB XAUI
1 GB SGMII
10 GB XAUI
1 GB SGMII
10 GB XAUI
1 GB SGMII

CORE Fabric Switch

4MB L2 Cache

AMBA AXI4

ARM® CORTEX™ QUAD CORE PROCESSORS

CORE Management Engine

DDR Controller

SATA

PCIe

SD/eMMC Controller

Testbench

Assertions

VIP BFM

Ethernet 400/200G
PCIe Gen4
USB Type C
AMBA ACE 4
HDMI 2.1
DDR 4

Test case & Sequence

Configurations

Monitor

Coverage

Scoreboard

Debug Support

SoC Quality Verification Assurance

- All interrupts tested at FC level.
- All register accessed at FC level.
- Performance analysis / Bandwidth utilization.
- Back pressure testing.
- Check All masters can access any slaves.
- Check all end points can be accessed at FC level.
- Random reset testing.
- Boot FSM coverage.
- All configuration interfaces must be tested.
- Must ensure no glitch in the design.

SoC Quality Verification Assurance

- For Multicore processor in design one core can wake up other cores.
- Every interface must be toggled at least once.
- Check X propagation in the design.
- All reset should be tested Hard / cold /software.
- Maximum delay of units (watchdog timer test).

SoC Blocks Quality verification Assurance

- ❖Test plan Maturity
- ❖Ease of Integration
- ❖Ease of Customization
- ❖Scalability
- ❖Deliverables
- ❖Result Analysis
- ❖Sign off criteria
- ❖Performance analysis
- ❖Formal verification

Effective Techniques for Verification

- Fully Random Environment
- Using Stub Models for IP
- Master AXI BFM replacing CPU
- Initializing IP`s using BFM Driver at SOC level
- Access to single/multi bus channels through single sequence model

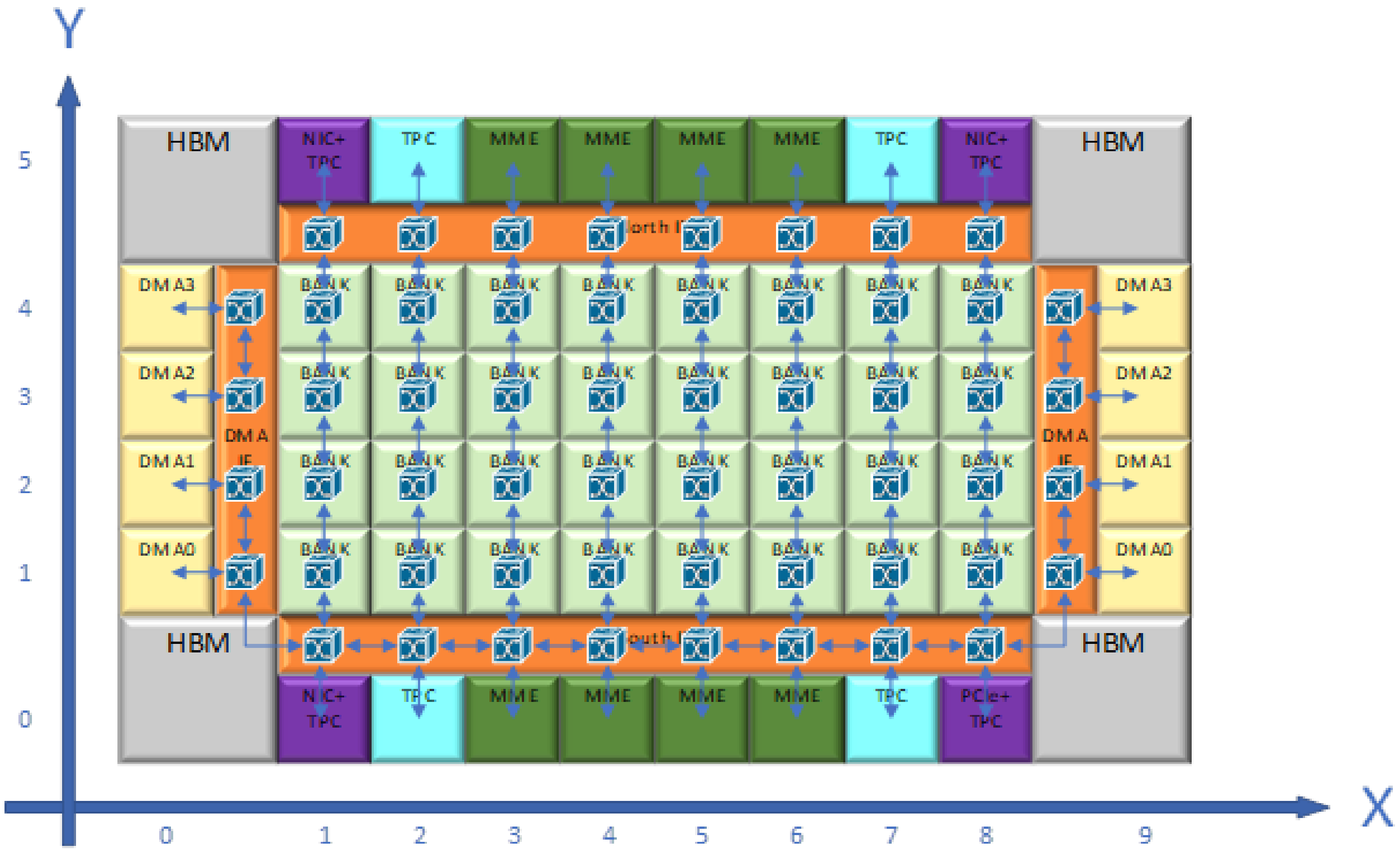
Fully Random Environment

- Connectivity test

- Performance testing

- Bandwidth Calculation

Block Diagram



Random ENV

- Replace All IPs by Master and Slave BFM’s at FC level. Depending upon the Randomized configurations.
- Configuration of Master and Slave are set according to the IP for which it is replaced.
- Master Initiates transaction towards multiple slaves at same time on random basis.
- Converters (AXI – AHB / AXI – APB Bridge) are used for address translation.
- In a single Env we can verify different interfaces (Low bandwidth / High Bandwith / Debug / trace etc.)

Conclusions/Observations

❖Helps in checking connectivity at FC level without the involvement of CPU.

❖Helps is performance calculation and Bandwidth Utilization.

❖Ease debugging and with less efforts we can catch corner cases.

❖Helps in Regress testing of full chip connectivity between the blocks.

Using Stub Models for IP

- Stub models can be used for the Blocks at Full Chip level.
- Blocks that are not been functional during the testing scenario can be replaced by there stub models.
- A Feed through has been provided which ensures that none of the signals are driven unknown.
- Example , while verifying read and write access to HBM from CPU we can stub DMA, NIC, PCIe etc.
- This helps in reducing the compilation and simulation time.

Master AXI BFM replacing CPU With C-Sequence

- ❖Performing read and write access from CPU consumes a lot of clock cycles.
- ❖Replacing CPU (Processor) with AXI BFM Driver can ease testing CPU interface at FC level.
- ❖A Perl script, that parse the C code to (Read and Write) AXI sequence will reduce the efforts of recoding the same sequence for SV.
- ❖This helps in regress testing of CPU interface, With less efforts.

Initializing IP`s using BFM Driver at SOC level

- ❖Normally Initializing an IP requires read and write to nearly 1000 of registers.
- ❖Performing transaction from CPU requires large number of clock cycles that increases the simulation time.
- ❖As CPU will first fetch the program code (From SRAM / SPI etc) Decode it than executes , this process consumes a lot of clock cycles.
- ❖Connecting the IP configuration interface by force at zero simulation time to BFM Driver (Interface).

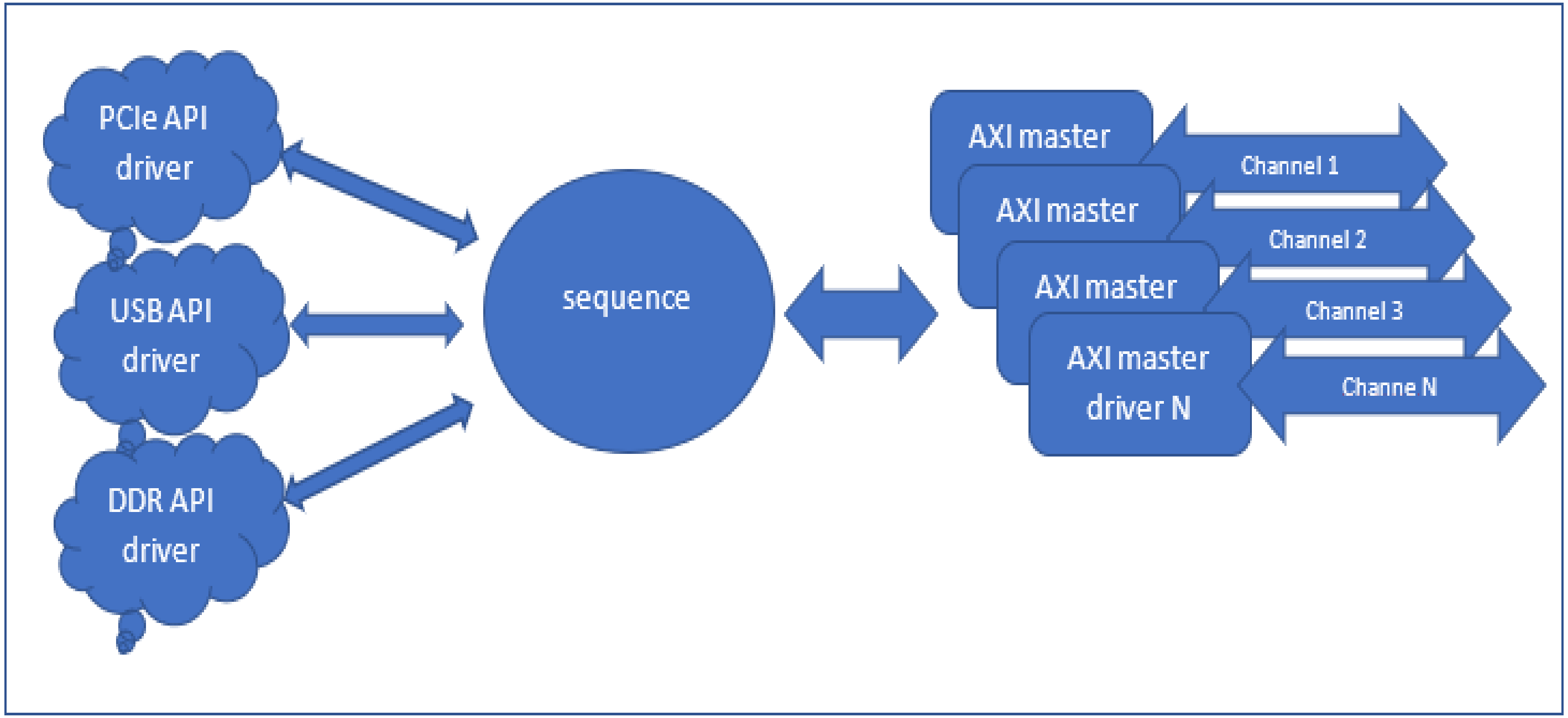
Initializing IP`s using BFM Driver at SOC level

- ❖Running the IP initialization sequence on BFM Driver.
- ❖Releasing the force when initialization is done will active the interface for SOC and other blocks can reconfigure it and perform read and write.
- ❖Many IP`s can be initialized at the same time, that will save a lot of simulation time.
- ❖ Helps to Decrease the simulation time for an IP having PHY and other Trainings.

Access to single/multi bus channels through single sequence model

- Multi channel system controllability using single sequence.
- Control the sequence using existing API’s/firmware
- Access to multi masters using ID
- Transaction priority decision within masters.
- Waited and non-waited write transactions
- Reduces the sequence/test complexity

Access to single/multi bus channels through single sequence model



Q/A

THANKS