Advantages of SCE-MI Based BFM’s
Contents

- Introduction to SCE-MI
- Drawbacks in existing Emulation platform
- SCE-MI as a solution for effective Co-Emulation
- Generic Architecture of Co-Emulation Modeling
- Approach in building SCE-MI based environment
- Use-MODEL as Ethernet
Introduction

- To provide communication between the testbench which is in high level language like SystemC or UVM (s/w side) and the IP component placed in FPGA on an emulation board (H/W side) through SCE-MI infrastructure.
- Interfacing software models to emulators.

```
<table>
<thead>
<tr>
<th>s/w side</th>
<th>H/w side</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test in SystemC, C++</td>
<td>Transactor in VHDL, Verilog or SystemVerilog</td>
</tr>
<tr>
<td>SCEMI API</td>
<td>DUT in VHDL, Verilog or SystemVerilog</td>
</tr>
<tr>
<td>C++ Compiler</td>
<td>Synthesis and P&amp;R</td>
</tr>
<tr>
<td>Infrastructure SCEMI</td>
<td></td>
</tr>
</tbody>
</table>
```
Drawbacks in existing Emulation Platform

- Unable to reuse the testbench used for Simulation based verification in the Emulation platform. Thus requires more effort, cost and time to recreate the same for the emulation platform.

- All emulators uses proprietary API's. This makes very difficult for software products to port to the different emulators thus restricting the solutions available to the customers. This in turn leads to low productivity and low return on investment who build their own solutions.

- The existing API's are signal/pin level oriented and not transaction level.

- Customers are reluctant to invest in building applications on proprietary APIs.
SCE-MI as a solution for effective Co-Emulation

- Enables the reuse of existing testbench used for simulation based verification to be used for emulation platform. Thereby reducing the effort and time.
- Enables the portability of transactor models between emulation vendors and makes possible for IP developers to write only one model
- Enables to reuse the transaction level scenarios generated at verification stage in the emulation platform.
- Enables to develop user defined API’s and thus helps in more customization.
Generic Architecture of Co-Emulation Modeling

- Transaction level Abstraction <-> RTL level Abstraction.
- Generation of software models at high abstract level and executing on Hardware at RTL abstract level through SCE-MI API
Approach in building SCE-MI based Environment

**Virtual Interface**
(Handle to Interface)

SV TestBench

SV INF

Verilog/Vhdl DUT
(Design Under Test)

Conventional TestBench
TestBench for Functional/Emulation Platform

SV TestBench

SV layer for function calls

SCE-MI API
Linked to C/C++ libraries

DP I C or C++

SV xTOR

Verilog/Vhdl DUT (Design Under Test)

Synthesizable BFM
Same Approach even for UVM TestBench

SV layer for function calls

SCE-MI API
Linked to C/C++
libraries

DP
or
C++

Synthesizable BFM

Verilog/Vhdl DUT
(Design Under Test)

UVM SEQR
UVM DRIVER
UVM TRANS
UVM SB
UVM MON

UVM SB
Using SystemC TestBench

SCE-MI API
Linked to C/C++ libraries

SystemC TLM TB

Proxy s/w ports

DP I C or C++

SV xTOR

Verilog/Vhdl DUT (Design Under Test)

Synthesizable BFM
Synchronization between Transactor and TB

- The most crucial and significant part in building the environment is synchronization between the Transactor and the TB

- The approach involves using
  - Semaphores
  - Handshake variables
  - FIFO’s
  - Callback functions
  - Tasks
USE MODEL – FUNCTION BASED MODEL EHTERNET

UVM Layer

Test Layer

Eth_agent

Eth_seqr

Eth_driver

Eth_sb

Eth_mon

Ethnet_Transactor.svh

With Functions, classes

DP

C or C++

RTL

Ethernet_Transactor.sv
# DPI Datatype Mapping

<table>
<thead>
<tr>
<th>DPI Arguments</th>
<th>C- MAPPED DATATYPES</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit</td>
<td>Unsigned char</td>
</tr>
<tr>
<td>byte</td>
<td>char</td>
</tr>
<tr>
<td>Int unsigned</td>
<td>Unsigned int</td>
</tr>
<tr>
<td>Int</td>
<td>int</td>
</tr>
<tr>
<td>1D Array – bit</td>
<td>svBitVecVal</td>
</tr>
<tr>
<td>1D Array - logic</td>
<td>svLogicVecVal</td>
</tr>
</tbody>
</table>
Write Operation

• DPI-C Import & Export Functions

  o import tx_wr_strt()
    ✓ Transactor provides start of packet indication

  o export tx_wr_pkt()
    ✓ The generated packet from user is send to C

  o import tx_wr_data()
    ✓ Transactor calls this after writing each data where in the written data is stored in FIFO in C.

  o import tx_wr_done()
    ✓ Transcator after driving one complete packet

  o export tx_wr_data_mon()
    ✓ The data from FIFO in C is given to SV TB through call back functions.
Read Operation

- DPI-C import and export Functions:
  - **import rx_rd_data()**
    - ✓ The RX data received from DUT is stored in FIFO created in C.
  - **import rx_rd_data_mon()**
    - ✓ When complete packet is received by SV TB through callback functions, indicate to C
  - **export rx_rd_callback()**
    - ✓ This callback Function is called to get the read packet form FIFO in C to SV
QUESTIONS …???