Enhancing TBA Performance Using Efficient SCE-MI Modelling

Ponnambalam Lakshmanan

ANALOG DEVICES

accellera SYSTEMS INITIATIVE™

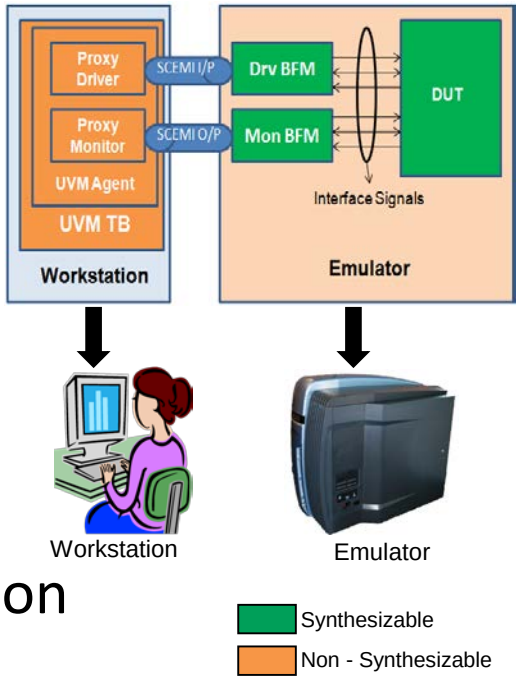
2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Agenda

- Introduction
- Factors Impacting Acceleration
- SCE-MI
- Testbench Architecture and Existing Challenges
- SCE-MI Direct Memory Interface
- SCE-MI Function Based Interface
- SCE-MI Pipes – Challenges and Solutions
- Results
- Conclusion and Future Work

Introduction

- Acceleration Challenges – Addressed with Hardware assisted testbench acceleration
  - Hardware accelerators to our rescue
  - Various acceleration modes



- Transaction Based Acceleration (TBA)
  - Testbench (synthesizable part) + DUT runs on emulator
  - Testbench (non-synthesizable part) runs on simulator.

Factors Impacting Acceleration

- Various factors directly affect the emulator performance
  - Inefficient usage of SCE-MI
  - Usage of Behavioral constructs (Not purely-synthesizable)
  - Memory Handling
  - Amount of code running on simulator and emulator
  - Simulator – Emulator synchronizations

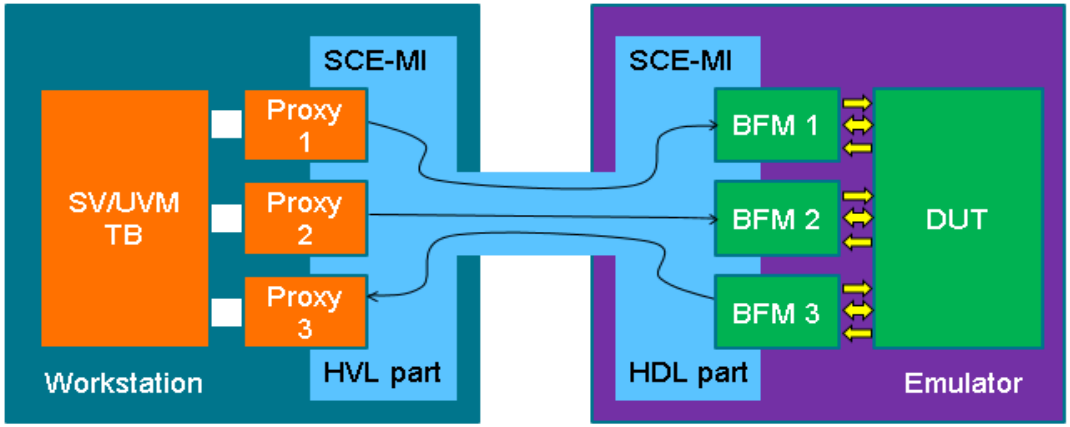
SCE-MI

- Standard Co-Emulation Modeling Interface (SCE-MI) is an Accellera standard

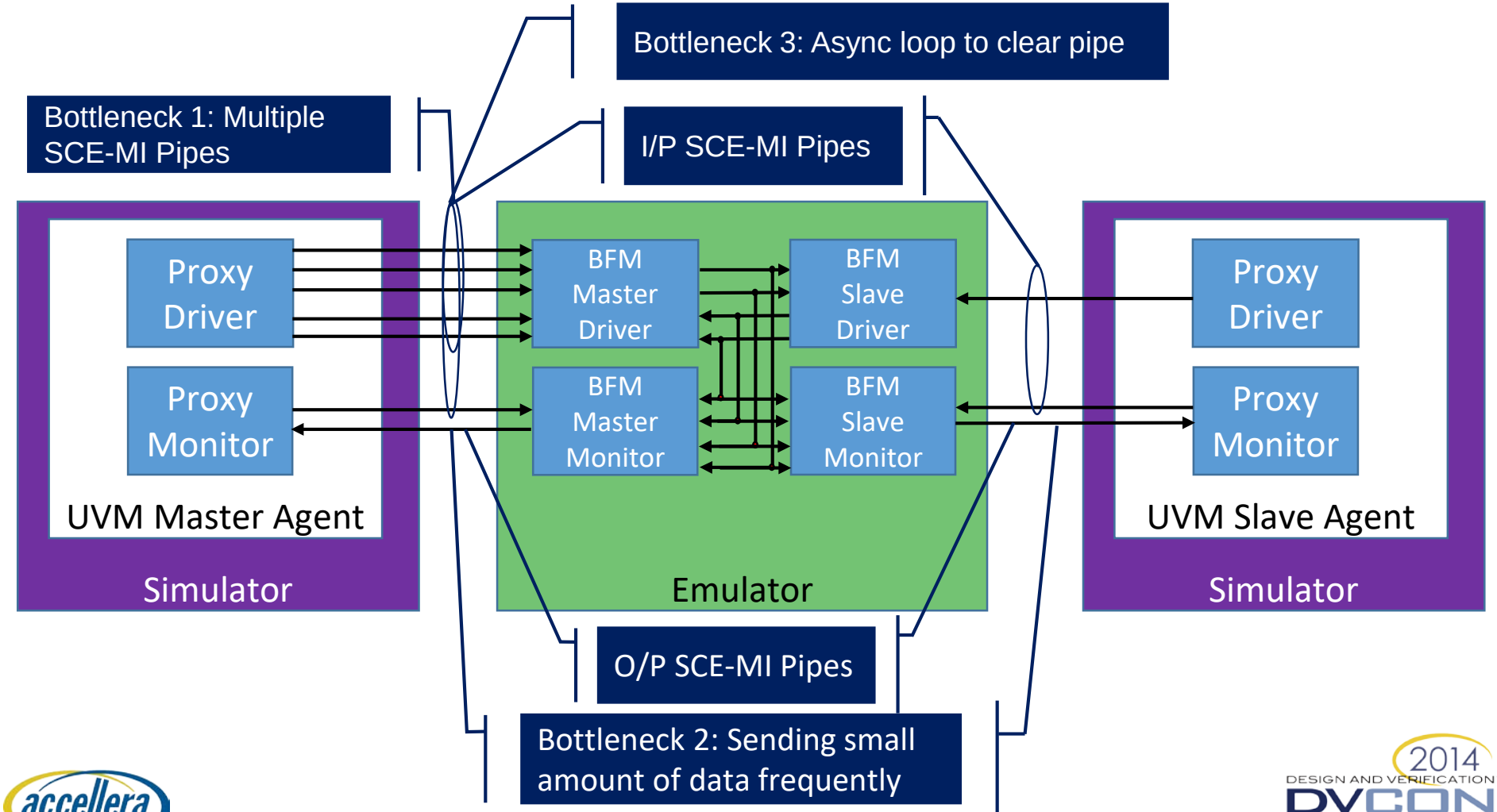
- Different flavors of SCE-MI
  - Pipe based Transaction
  - Function based Interface
  - Direct Memory Interface (DMI)

SCE-MI Pipes

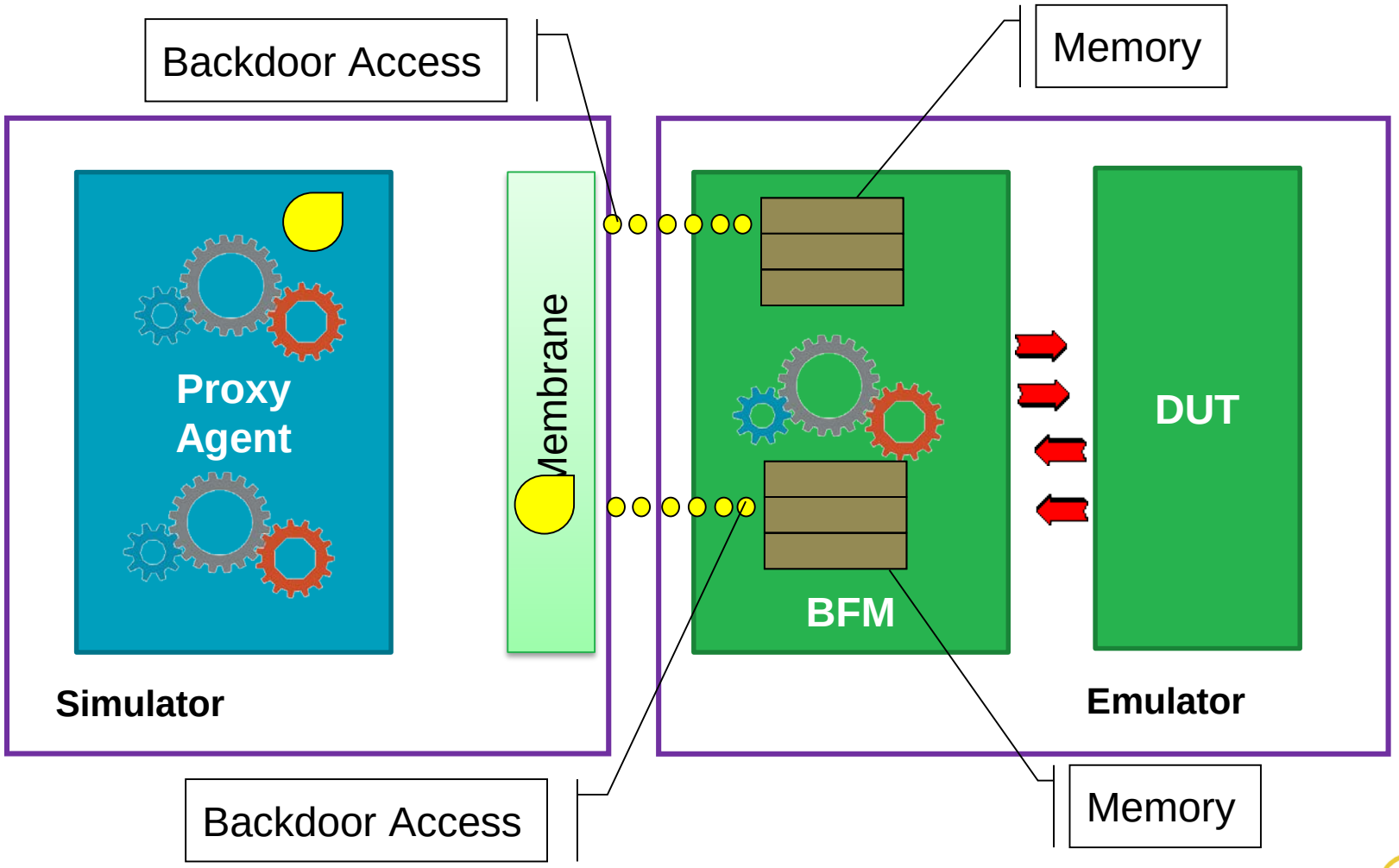
- Salient Features:
  - Unidirectional
  - Batching
  - Buffering
  - Flushing
  - Data shaping
  - Blocking and Non-Blocking constructs



Testbench Architecture and Existing Challenges



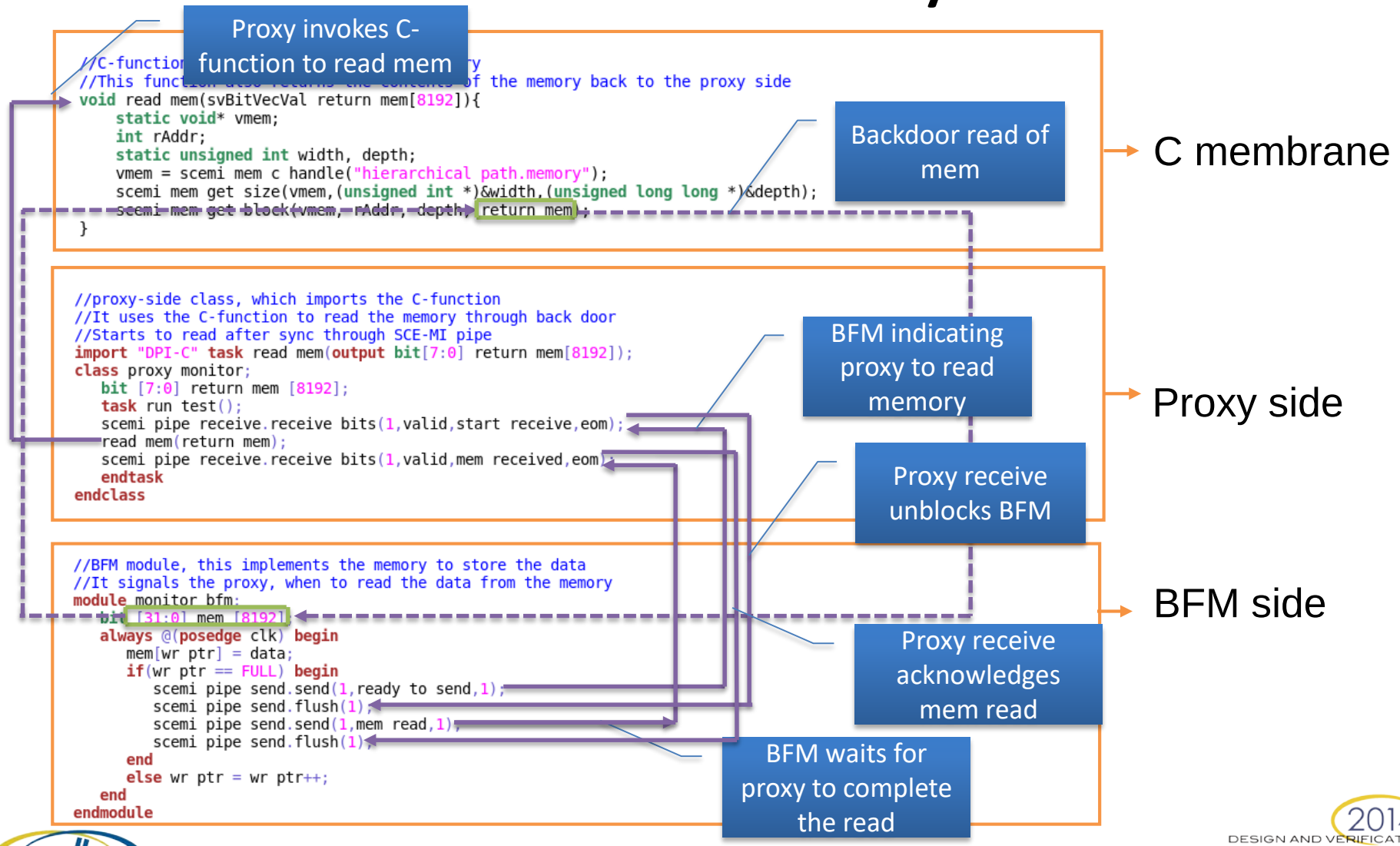
SCE-MI Direct Memory Interface



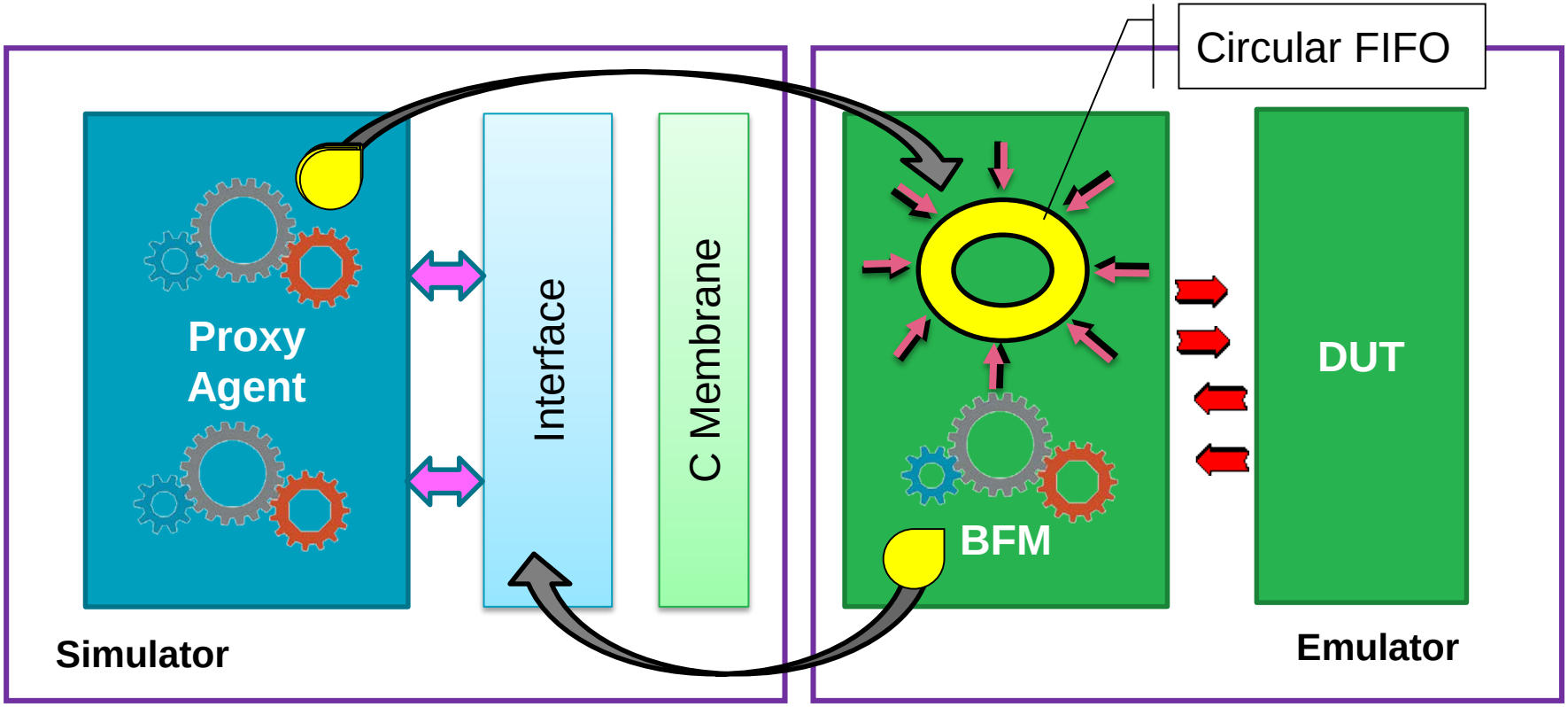
SCE-MI Direct Memory Interface

- Used to access DUT/BFM memory from the proxy domain
- Transfer Bulk data (Proxy <-> BFM)
  - Block interface
  - Word interface
- Performance Improvement:
  - HW-SW synchronizations reduced by ~50%
  - TBA run time decreased by >25%
- DUT/BFM registers cannot be accessed

SCE-MI Direct Memory Interface



SCE-MI Function Based Interface

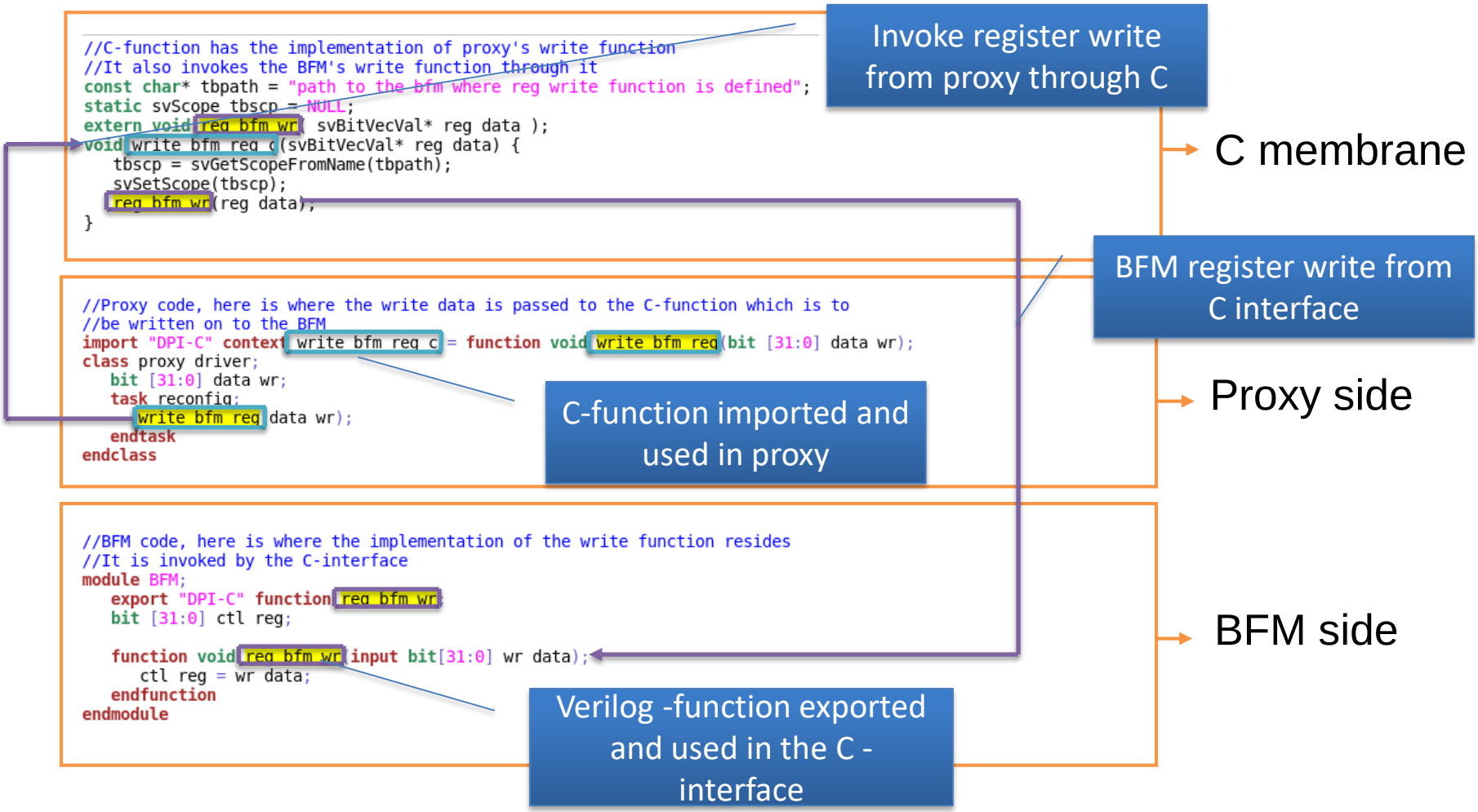


- Used to write into BFM registers
- Memory access through DMI has better performance

Write Pointer

Read Pointer

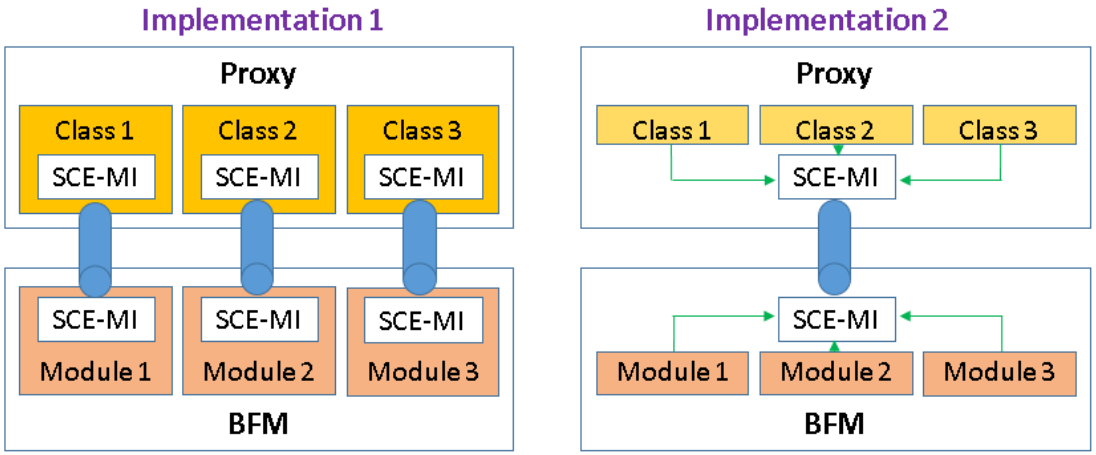
SCE-MI Function Based Interface



SCE-MI Pipes – Challenges and Solutions

- Multiple instances of SCE-MI pipe

Challenge	Solution
Behavioral evals with every SCE-MI access	• Send data via a single SCE-MI pipe. • Static SCE-MI pipe - Bit vectors • Dynamic SCE-MI pipe - Array of data



SCE-MI Pipes – Challenges and Solutions

- Behavioral evals with SCE-MI pipe

Challenge	Solution
• Asynchronous mode creates significant behavioral evals	Synchronous mode of SCE-MI pipe IS_CLOCKED_INTF = 1

Code Snippet

```
scemi_input_pipe #(
.BYTES_PER_ELEMENT(20),
.PAYLOAD_MAX_ELEMENTS(1),
.VISIBILITY_MODE(2),
.IS_CLOCKED_INTF(0) )
inbox (clk);
```

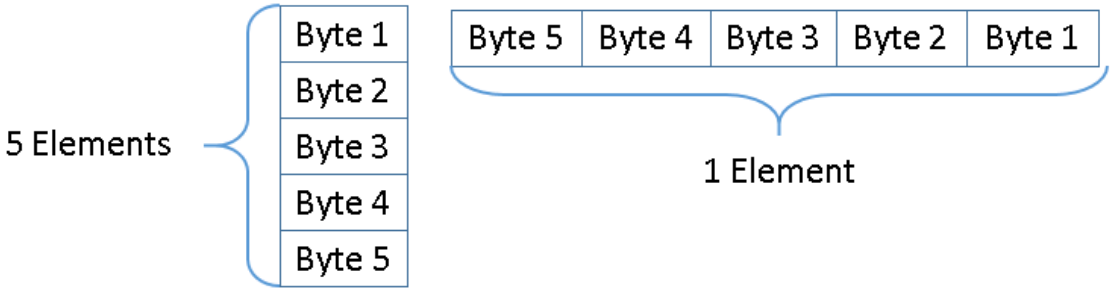
Code Snippet

```
scemi_input_pipe #(
.BYTES_PER_ELEMENT(20),
.PAYLOAD_MAX_ELEMENTS(1),
.VISIBILITY_MODE(2),
.IS_CLOCKED_INTF(1) )
inbox (clk);
```

SCE-MI Pipes – Challenges and Solutions

- Amount of data accessed

Challenge	Solution
Accessing more than one element per access results in behavioral evals • PAYLOAD_MAX_ELEMENTS = 5; • BYTES_PER_ELEMENT = 1;	If the intention is to access 5 Bytes per call then set • PAYLOAD_MAX_ELEMENTS = 1; • BYTES_PER_ELEMENT = 5;



SCE-MI Pipes – Challenges and Solutions

- Synchronization between Simulator and Emulator

Challenge	Solution
• Frequent data transfer causes the emulator to halt frequently • Results in degraded acceleration	• Accumulate multiple bytes of data and transfer at once • Try to buffer the elements • Avoid unnecessary flush() usage

Code Snippet

```
scemi_input_pipe #(
.BYTES_PER_ELEMENT(20),
.PAYLOAD_MAX_ELEMENTS(1),
.BUFFER_MAX_ELEMENTS(10),
.VISIBILITY_MODE(2),
.IS_CLOCKED_INTF(1) ) inbox (clk);
```

SCE-MI Pipes – Challenges and Solutions

- Clearing the contents of SCE-MI pipe

Challenge	Solution
• Discarding buffer contents on reset • No inbuilt functions • Increase in step-count	• Use the fastest clock available to synchronously fetch data from pipe

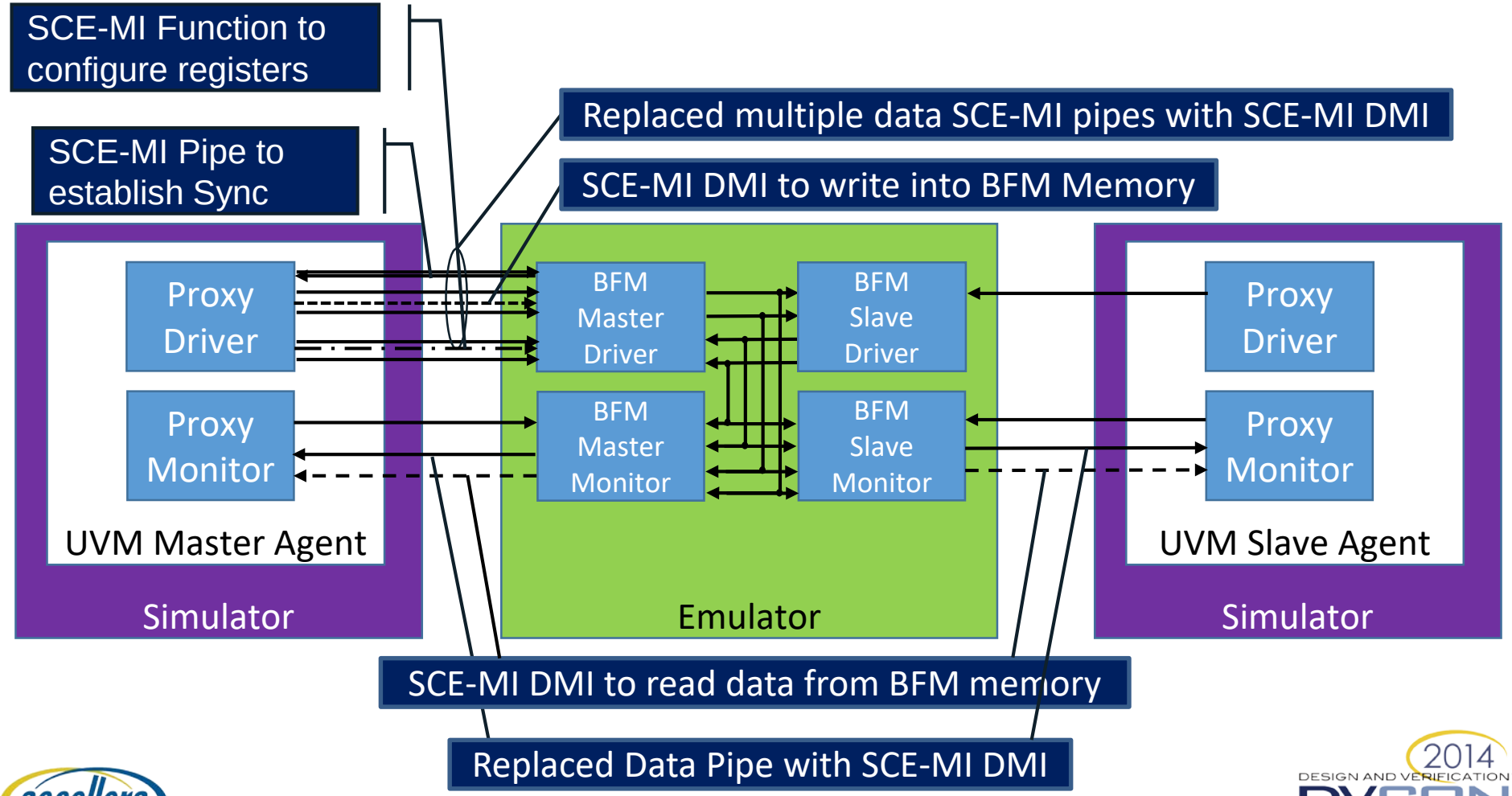
Code Snippet : Before

```
always@(posedge clk, posedge rst) begin
  if(rst) begin
      // Statements
      for(int i=0, i<20), i++)
         inbox.receive(1,ve,data,eom);
  end
  else begin
      // statements
  end
end
```

Code Snippet : After

```
always@(posedge fst_clk, posedge rst) begin
  if(rst) begin
      rst_buff = 1;
      // statements
  end
  else begin
      if(rst_buff && !eom)
         inbox.receive(1,ve,data,eom);
      else
         rst_buff = 0;
   end
end
```

Testbench Architecture: After Optimization



Results

- TBA performance improvement

	Gate Count	Bevals	HW-SW Sync	TBA Time
Before Optimization	~2 M	18,299,173	4,728,998	~60 min
After Optimization	~2 M	218	18,439	5 min

- Simulation v/s TBA run-time comparison

	Simulation Time	TBA Time
Simulation v/s TBA	~360 min	~9 min

Conclusion

- SCE-MI has multiple interfaces for different use cases
- HW-SW sync and Bevals could greatly deteriorate the performance of emulator.
- Efficient usage of SCE-MI helps in leveraging maximum performance from emulator resulting in handsome **SPEED-UP**.

Future Work

- To integrate the agents with the actual DUT (IP) and test on emulator

- Porting to SOC environment and analyze the performance

Acknowledgements

- David Brownell, ADI

Questions