Connecting a Company’s Verification Methodology to Standard Concepts of UVM

Frank Poppen, OFFIS
Marco Trunzer, Robert Bosch GmbH
Jan-Hendrik Oetjens, Robert Bosch GmbH
The Challenge

• electronics in heterogeneous systems
• ambient and safety relevant
• increasing complexity
• design and verification
• lining up for the task
  – tailored solutions
  – standards
  – languages
  – tools
No „One Size Fits All“

• verification engineers choose and combine what …
  – fits best for the company
  – the design-team
  – the application domain
  – (budget, roadmap, …)
• deep roots in the design process
• changes endanger productivity
• change carefully and incrementally
Outline

• Motivation
• Verification Islands
• Matching Concepts of UVM and IFS
• Bus Arbiter Test Case
• UVMC can do / cannot do
• Architecture of the Experiment
• Conclusions
Bringing it All Together

- lining up for the task
  - tailored solutions
  - standards
  - languages
  - tools

VHDL-AMS
VHDL
IFS
UVM
SystemVerilog
SystemC
MATLAB SIMULINK
Verification Islands

Methodologies & Concepts

(Hardware Design) Languages

Frameworks & Tools

UVM
- SystemVerilog TestBench
- DUT

IFS
- SystemC TestBench
- DUT

MBD
- Matlab TestBench
- DUT
Outline

• Motivation
• Verification Islands
• Matching Concepts of UVM and IFS
• Bus Arbiter Test Case
• UVMC can do / cannot do
• Architecture of the Experiment
• Conclusions
IFS long before SystemVerilog

- enhanced from VHDL with ...
  - VHDL-AMS
  - SystemC
  - Matlab/Simulink
  - (and now SystemVerilog and UVM)
- SystemC based library simulates with any simulator (IEEE 1666)
- tailored to relevant use scenarios in special contexts
- simple IFS command language for (self-checking) test cases
  - digital designer
  - analog designer
  - verification engineer
  - system engineer
  - software engineer
Matching Concepts of **UVM** and IFS

- **UVM Environment**
- **UVM Agent**
- **UVM Sequence**
- **UVM Sequence Item**

Diagram:
- Test Environment
- Coverage Monitor
- Agent
- Monitor
- Driver
- Sequencer
- (Virtual) Sequencer
- Scoreboard
- Sequence Item
- Configuration
- DUV
Matching Concepts of UVM and IFS

- CF (Command File)
- IFS Command File
- IFS TBM Command
- IFS Controller
- sc_main
- UVM Environment
- UVM Agent
- UVM Sequence
- UVM Sequence Item

TBM (Test Bench Module)
TBM (Test Bench Module)
TBM (Test Bench Module)

© Accellera Systems Initiative
Matching Concepts of UVM and IFS

• TBM directly correlates to UVM agent
• transactor between test bench and DUV
• translate messages.Commands to bit wiggles
• Objective: reuse of UVM agents delivered with UVM test environment / verification IP
• UVM agent can be used inside SystemC

• connects any SystemC test bench to SV UVM ...
  ... does not rely on the IFS library!
Matching Concepts of UVM and IFS

- CF (Command File)
  - TBM Command
- IFS Controller
- IFS Test Bench Module
- IFS Command File
- IFS TBM Command
- UVM Environment
- UVM Agent
- UVM Sequence
- UVM Sequence Item

sc_main

Controller

TBM (Test Bench Module)

DUV

UVM Agent
Outline

• Motivation
• Verification Islands
• Matching Concepts of UVM and IFS
• Bus Arbiter Test Case
• UVMC can do / cannot do
• Architecture of the Experiment
• Conclusions
Simple Switched Bus Arbiter Test Case

• configurable number of bus masters and slaves
• suited for experiment – simple to understand – easy to verify
• used for several mixed-language simulations evaluations
• VHDL, Verilog and SystemC
Simple Switched Bus Arbiter Test Case

CLK PERIOD 10 ns
CLK RESET 0 12

ALL SYNC ALL

SL1 print "Config slave 1!"
SL1 Set_Offset 300
SL1 Set_I_Wait 100
SL2 print "Config slave 2!"
SL2 Set_Offset 200
SL2 Set_I_Wait 99
SL3 print "Config slave 3!"
SL3 Set_Offset 100
SL3 Set_I_Wait 47
SL4 print "Config slave 4!"
SL4 Set_Offset 0
SL4 Set_I_Wait 69

ALL SYNC ALL

#loop 4
ALL SYNC ALL
MS1 Write $(100+#i) $(100+#i)
MS1 Read $(100+#i) $(100+#i)
eol

ALL SYNC ALL

SL1 print "End Of Test Script"

ALL QUIT
Simple Switched Bus Arbiter Test Case

SystemC 2.3.0-ASI --- Nov 29 2013 14:57:17
Copyright (c) 1996-2012 by all Contributors,
ALL RIGHTS RESERVED
INFO (0 s) Loading script: 'control.cmd'
INFO (0 s) Exiting simulation.
Info: /OSCI/SystemC: Simulation stopped by user.
INFO (0 s) Report:
INFO (0 s) Encountered errors: 0
INFO (0 s) Encountered warnings: 0
Outline

• Motivation
• Verification Islands
• Matching Concepts of UVM and IFS
• Bus Arbiter Test Case
• UVMC can do / cannot do
• Architecture of the Experiment
• Conclusions
UVM Connect (Verification Academy)

- interconnect UVM and SystemC
- UVMC is an open-source UVM/OVM-based library
- uses System Verilog Direct Programming Interface
- good for TLM messages and control commands
- less good for RT level signal interfacing
UVM Connect (Verification Academy)

- interconnect UVM and SystemC
- UVMC is an open-source UVM/OVM-based library
- uses System Verilog Direct Programming Interface
- good for TLM messages and control commands
- less good for RT level signal interfacing
  => instantiation through foreign language module
Outline

• Motivation
• Verification Islands
• Matching Concepts of UVM and IFS
• Bus Arbiter Test Case
• UVMC can do / cannot do
• Architecture of the Experiment
• Conclusions
Architecture of the Experiment

- **Controller**: Manage communication between different components.

- **Wrapper**: Connects SystemVerilog module to UVM.
  - `init()`: Object creation.
  - `finalise()`: Object destruction.

- **SystemVerilog module**: Functions like `execNextCmd` and `seqItmX()`.

- **UVM Agent**: Components like driver, sequencer, and monitor.

- **UVMC/TLM**: Interface between SystemVerilog and UVM.

- **Bus Arbiters**: slave0, slave1, slave2, slave3, master1, master2, master3.

- **Sc_port**: Interface between components.
  - `write()`, `read()`, `seqItmX()`.

- **Wrapper UVM Connector**: Bridge between SystemVerilog and UVM.

- **VIF**: Verification Interface Framework.
Running the Experiment

• elaboration phase: instantiating foreign module
• SystemC part of UVMC opens port
• IFS registers test bench modules
• loading command file
Running the Experiment

- running simulation
- UVM phasing starts after SystemC elaboration
- UVMC starts
- System Verilog part of UVMC opens port
Running the Experiment

- raising run phase objection
- both ports SC and SV get connected
- SC sending payload / SV receiving payload
- UVM agent (driver) receiving sequence item
Running the Experiment

- script reached ‘quit’ command
- SystemC wrapper drops UVM objection
- simulation ends
Outline

• Motivation
• Verification Islands
• Matching Concepts of UVM and IFS
• Bus Arbiter Test Case
• UVMC can do / cannot do
• Architecture of the Experiment
• Conclusions
Conclusion

• evolution of tailored verification methods to UVM
• UVM agents can be controlled in SystemC test environments
• approach is generic: IFS library is not mandatory
• standards for SV/DPI and SC are mature to support UVMC
  – worked for both simulators Questa and Incisive
  – no real doubt would work with VCS
• complex interferences between SystemVerilog, UVM, UVMC, SystemC and simulator
• future work
  – reverse approach and use TBM in UVM test environment
Questions

Acknowledgements: This work has been funded by the German Federal Ministry for Education and Research (Bundesministerium für Bildung und Forschung, BMBF) under the grant 01IS13022 (project EffektiV). The content of this publication lies within the responsibility of the authors.