Migrating to UVM: how and why!

Mike Bartley
Test and Verification Solutions
Agenda

• Motivation
  – Why do we need a common methodology?
  – What should a methodology provide?
• A short history of methodologies
• Synopsys support for OVM
  – Using some OVM VIP
• Migrating to UVM
The various types of verification

- Static
  - Reviews
  - Code Analysis
  - Formal
- Dynamic
  - Simulation
    - Dynamic Formal
  - Emulation etc.
Constrained random test bench

Design Under Test

Configuration

Driver BFM

Sequencer

Monitor BFM

Assertions

Code Coverage

Functional Coverage

Assertions

Monitor BFM

Sequencer

Driver BFM

Configuration

Mike Bartley
Why do we need a methodology?

• Building constrained random test benches is
  – Very complex
  – Very time consuming
• Most verification engineers will divide and conquer
  – This will allow various skills to be applied
  – And open up the scope for reuse
• And this is enabled through our methodology
What must a methodology provide?

- Standard to enable re-use - industry wide!
  - Abstraction re-use
  - Project re-use
  - Company re-use
  - Industry re-use
What must a methodology provide?

• A layered approach to enable a division of skills and labour
  – Developing verification IP
  – Test bench construction
    • Re-using and configuring existing verification IP
    • Writing of new code
  – Assertions and/or coverage points
  – Writing or re-using checkers
  – Writing or re-using sequences and test cases

• A word of caution
What must a methodology provide?

- A consistency of approach
  - Naming conventions (avoid name space clashes)
  - Verification IP configuration
    - Defining the number of agents
    - How they connect to the DUT
    - Whether an agent is ACTIVE or PASSIVE
  - Well-defined generation and simulation phases
    - Build, connect, pre-run, run, post-run
  - Standard hooks and events are provided
    - For example – “transfer_end”
What must a methodology provide?

- Ability to adapt the behaviour of the test bench
  - Macros
  - Callbacks
  - Factories
  - Virtual functions
  - Aspects

How much pre-planning?

Language Independence?
What must a methodology provide?

- The power to “find” bugs!

Stimulate

.....01010101
.....01001101
.....10011010
.....01001101

Propagate

01100101.....
11110101.....
00010101.....

Observe

Compare

Actual Results

Expected Results

Design Under Test

Mike Bartley
What must a methodology provide?

- Legacy
- Independence
  - Vendor independence
  - Language independence?
- Management
  - Planning, progress and completion
A brief history of methodologies

e Reuse Methodology
- First introduced in 2002
- Main aim was to enable re-use
  - Coexistence
  - Commonality
  - Cooperation
- Rules for building reusable, consistent, extensible, plug-and-play verification environments using e Verification components (eVCs)
A brief history of methodologies

Universal Reuse Methodology
• Introduced in 2006
• Delivered with supporting examples and documentation
• Based on proven techniques in production use with the e Reuse Methodology (eRM)
Advanced Verif. Methodology

- Provides libraries of base classes and modules in open-source form
- Uses TLM interfaces as the communication mechanism between verification components.
A brief history of methodologies

Open Verification Methodology

- Joint development initiative between Mentor Graphics and Cadence Design Systems first introduced in 2008
- TLM communication.
- Common user-extensible phases.
- Reuse and customization through a *class factory*.
- Common configuration interface
- Standard test writer interface for configuring a testbench and specifying constrained layered sequential stimulus
- Common message reporting and formatting interface
A brief history of methodologies

Reuse Verification Methodology
• Based on the Vera testbench language.

Verification Methodology Manual
• ARM and Synopsys began a collaboration in 2004 to develop a SystemVerilog verification methodology.
• The result – the VMM methodology – was defined in the book Verification Methodology Manual for SystemVerilog.
A brief history of methodologies

Unified Verif. Methodology

- Based on OVM and VMM
- Industry-wide verification methodology
- Requirements agreed by Accellera Verification Intellectual Property Technical Sub-Committee (VIP TSC)
- Specific modifications to OVM 2.1.1 to create an early adopter kit (UVM-EA)
  - “ovm” changed to “uvm”
  - review the “end-of-test” and “callback” services
  - add “message catching”
Simulating OVM 2.0.3 VIP with VCS

• OVM2.0.3 compiles on VCS 2009.12-3

  - Linux% setenv OVM_HOME <path>/ovm-2.0.3
  - Linux% cd examples/hello_world/ovm
  - Linux% vcs -sverilog +incdir+${OVM_HOME}/src hello_world.sv
Simulating OVM 2.0.3 VIP with VCS
Simulating OVM 2.0.3 VIP with VCS

- Only a few small code changes needed in VIP
  - Packed dimensions of arrays of enum types
  - Redefined enum types
  - Time units

vcs -sverilog -timescale=1ns/1ns\n    +define+TVS_SDC_ADDR_WIDTH=32 \n    +incdir+sv+rtl_tb +incdir+${OVM_HOME}/src \n    +incdir+examples/tb +incdir+examples/sve \n    +incdir+examples/test_lib \n    +incdir+examples/seq_lib +incdir+examples \n    ${OVM_HOME}/src/ovm_pkg.sv \n    ./examples/tb/tvs_sdc_tb_top.sv
About UVM

- UVM is being overseen by the Accellera “Verification Intellectual Property Technical Subcommittee” (VIP TSC)
  - http://www.accellera.org/activities/vip/
- Remit
  - To deliver a standard verification methodology and common base class library (CBCL) to enable users to deploy an efficient, reusable, and interoperable SystemVerilog verification environment
- API specification of the CBCL
- OVM version 2.0.3 the original starting point for UVM – now 2.1.1
- Add functionality from VMM, OVM later releases and/or other contributions from VIP TSC members
- The development model detailed in Intel’s foil posted on December 16, 2009
Adopting UVM?

- VCS users can already use OVM VIP
- If you want to mix VMM and OVM then read
- Migrating to UVM will provide you with a better long-term common, industry-wide platform
- Mentor UVM-EA kit based on OVM 2.1.1
  - Already out-of-date
  - Recommend you join Accellera VIP TSC
Summary

• Why do we need a common methodology?
• What should a methodology provide?
• A short history of methodologies
• Synopsys support for OVM
• Migrating to UVM

• Acknowledgements
  – Yassine Eben Amine and Douglas Fisher, Synopsys

• Any questions?