Adopting a Methodology: Experiences of OVM Deployment at Dialog

Steve Holloway – Senior Verification Engineer
Dialog Semiconductor
Why Choose OVM?

• OVM is a standard methodology
  • Built on SystemVerilog (IEEE 1800)
  • Long heritage of best practise (AVM + URM)

• Cross tool compatibility
  • Switch tools & vendors more easily

• Availability of 3rd party VIP
  • Higher quality verification of (standard) interfaces
  • Rapid development of complex testbenches

• Reduced team ramp-up time
  • Rapid familiarity with company verification environment
  • Easier to find resources with appropriate skills

• Sensible choice for new adopter of Metric Driven verification
What are we trying to achieve?

- Constrained-random stimulus vectors
- Automatically checked DUT response
- Functional Coverage = verification completeness
- Modular, *reusable* coding strategy
• The OVM package contains three main types of class:

**ovm_component**
- A structural component used in an OVM testbench
- It can be “built” in a hierarchy
- It can have configurable items which are “set” as it gets built
- Is synchronised with other ovm_components with a well-defined flow of simulation phases
- Facilitates a “modular” reuse strategy

**ovm_object**
- A generalised data structure
- Can hold testbench configuration information

**ovm_transaction**
- A data structure for stimulus generation, monitoring and analysis
The OVM Factory

- Based on the concept of the OOP “Factory Pattern”
- It can create objects at runtime, whose type are dynamically selected by overrides

```cpp
set_type_override_by_type(i2c_trans::get_type(), i2c_error_trans::get_type())
```
The Importance of a Plan

- Project Manager: Tracking status
- Architect: Ensure intent is realised in design
- Verification Engineer: Common status document & buy-in
- Design Engineer: Ensure implementation is in line with spec

Verification Plan

- Verification Environment Development
- Execute Sessions
- Plan completion
- Coverage Metrics
- Debug
Guaranteed Success?

- OVM = Kit of standard parts & user guide
- SystemVerilog = Feature-rich HVL
- There's More Than One Way To Do It
  - Inconsistent “look & feel”
  - Non – reusable code
  - Using the wrong level of abstraction
- “Boilerplate” getting in the way of the real work
  - Coverage closure & debug
- Recreating in-house class libraries & macros
Rollout Steps

• External training courses & workshops

• Internal seminars & knowledge sharing
  • Best practise guidelines
  • Wiki knowledge base
  • Code examples

• External OVM resources
  • OVM Forum
  • Verification Academy
  • External consultants

• Introduction on live projects
  • Code review sessions

• Library OVC components
Problems in Execution

• RTL-centric engineers learning OOP concepts

• Stimulus not constrained appropriately

• Checking at the wrong level of abstraction
  • Reference model in module-based “helper” code + assertions

• Dangerous use of configuration settings
  • `set_config_int("*", "num_agents", ...);`
  • No encapsulation of configuration

• Slippage between Vplan & coverage model

• Derivative projects could not reuse agents easily
  • Tightly coupled to interface

• Module – chip reuse is non-trivial
Solutions

- Encapsulate VIP settings in configuration objects
- Encapsulation of BFM tasks in interface
  - Better reuse model for derivative DUTs with changing i/f
- Structure of Scoreboard for reuse & decoupled checks
  - E.g. MVC pattern
- Leverage common sequence API (e.g. register-based)
- Review process essential to ensure consistent verification approach
- Multi-layered approach to verification
  - Infrastructure & VIP development
  - Project specific stimulus, checks and coverage
Reviews throughout

Vplan Creation

Vplan Review

Environment Creation

Verification team

Code Review

Coverage & Checks Creation

Design team

Metrics Review
Leveraging Reuse

- Derivative project
- Initial rollout project

VPlanning & Environment Coding

Debug & Closure
Conclusions

• OVM constrained-random approach resulted in:
  • High rates of bug discovery
  • Easier tracking of real progress
  • Managed verification closure

• OVM won’t initially reduce your verification effort
  • Until reuse is leveraged

• Legacy directed tests can still add value
  • OVM checking in passive mode

• Engineers were able to get running quickly
  • Application-specific examples & knowledge sharing

• UVM roll-out in progress!
Energy Management Excellence