

Successful Adoption of OVM and What We Learned Along the Way

by Mike Bartley, Founder, TVS and Steve Holloway, Senior Verification Engineer, Dialog Semiconductor

ABSTRACT

OVM has gained a lot of momentum in the market to become the dominant verification “methodology” and the indications are that UVM will gain even greater adoption. OVM is built around SystemVerilog and provide libraries that allow the user to build advanced verification test benches more quickly. There is extensive documentation, training and support for how to best develop such test benches and to encourage test bench re-use. However, there is less advice on how to adapt your verification processes on your project to best use OVM and even less advice on how to do this for company wide adoption.

In this article we discuss the experiences of the authors of a company-wide adoption of OVM. We consider the original motivations for that adoption and the expected benefits, and the actual results achieved and problems that have been overcome. The aim of the article is to give advice to other companies considering adopting OVM.

WHY ADOPT A METHODOLOGY? AND WHY OVM?

Figure 1 shows a brief of industry test bench methodologies and goes beyond OVM to UVM.

Dialog had already decided that they wanted to improve their verification process by moving it from a directed testing approach to use pseudo-random verification. The main

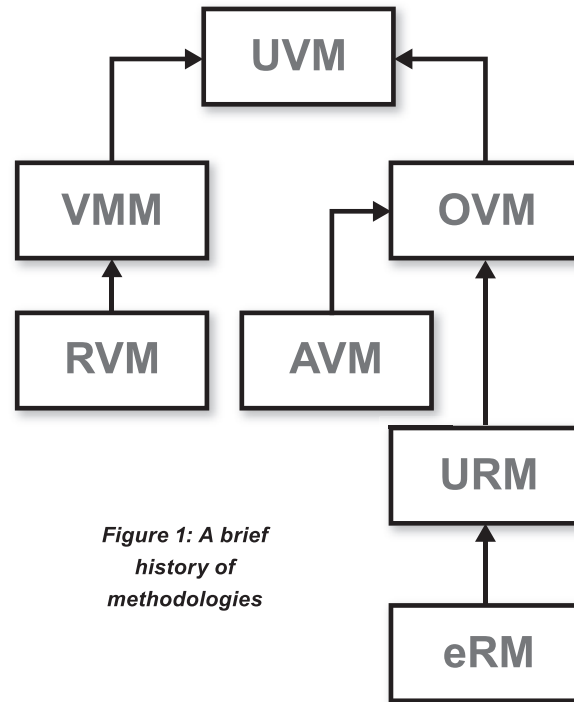


Figure 1: A brief history of methodologies

question was the most effective way of making such a transition and it was clear that adopting an existing methodology that supported a pseudo-random verification was the best way forward.

OVM is currently the leading industry pseudo-random verification methodology and has a well planned and supported roadmap to UVM (the Universal Verification Methodology) which is supported by Cadence, Mentor Graphics and Synopsys. Thus the decision by Dialog to adopt OVM was reasonably obvious.

Figure 2 (on the following page) “OVM Verification Hierarchy” shows how OVM is made up of a class library and methodology. The OVM class library is built on SystemVerilog which has been standardised (IEEE 1800). The class library is freely distributed and can be run across multiple industry simulators from different vendors. The OVM methodology is then supported by the OVM class libraries. The methodology is also supported by other languages: SystemC® (IEEE 1666), and e (IEEE 1647) languages.



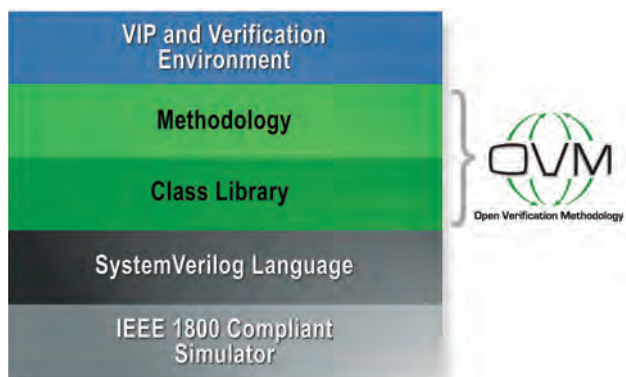


Figure 2: OVM Verification Hierarchy

The fact that the language, library and methodology have been standardised and are supported by multiple companies and vendors brings significant additional advantages to Dialog:

- **Availability of training & support:** As the adoption of a common methodology gains widespread industry acceptance then the market for services and products based on the standard increases. Thus vendors are able to start providing those services and products.
- **Compatibility & availability of 3rd party IP:** Similar to training and support, the availability of 3rd party IP increases with growing adoption of a methodology. In particular, vendors are able to produce and sell their verification IP (VIP) to a larger market. It should also be easier for the user to incorporate that VIP into their verification environment if that environment and the VIP are compatible through the standard – a “Write once, run everywhere?” concept. Note also that the OVM standard supports the use of plug-and-play verification IP (VIP) written in SystemVerilog (IEEE 1800), SystemC® (IEEE 1666), and e (IEEE 1647) languages.
- **Cross-tool compatibility:** The standardisation process should ensure that tools from different vendors support both the methodology and the underlying class libraries making it easier for users to switch tools and thus vendors more easily.

We can all point to differences in the ways that simulators interpret the semantics of a particular language. Indeed, in many places the language standard does not completely cover the implementation and the tool vendor is left to make their own decision

– race conditions being an obvious example. However, these issues are usually minor enough not to block a switch to an alternative supplier.

The ability to switch supplier is vitally important commercially when negotiating with tool vendors. It also means that the user has an alternative should their current supplier cut support for their simulator.

- **Reduced ramp up time for subcontractors & new starters:** Skilled verification resources remain in high demand and short supply. Use of an industry standard methodology both increases the probability of finding resource with the appropriate skills and knowledge, and reduces the time to ramp those people in your particular verification environment.

ADVANTAGES OF ADOPTING EXTERNAL VIP

In addition to the above reasons from Dialog, TVS has noticed additional advantages that other clients have seen from the adoption of OVM. The first of these is in the use of 3rd party OVM VIP:

- The ability to more rapidly develop complex test benches
- Independent interpretation (of protocols) leading to higher quality verification of interfaces
- Ensure higher coverage of protocols through the cover points and assertions built into the VIP and mapped to the protocol specification (assuming your VIP supplier does this)
- Easier re-use of VIP between OVM-compliant projects and test benches, and across hierarchies in the same project (e.g. from block to SoC – the VIP is used more for traffic generation in the latter given the interface is already proven)
- Leverage knowledge and experience from others
- Huge potential ROI (return on investment) from the VIP

ROLLING OUT OVM ACROSS

Figure 3 “Dialog OVM test bench structure” shows a typical OVM-style test bench and this is the style of test bench that Dialog chose to adopt. The agents shown could be either internally developed or external VIP (such as the VIP from TVS).

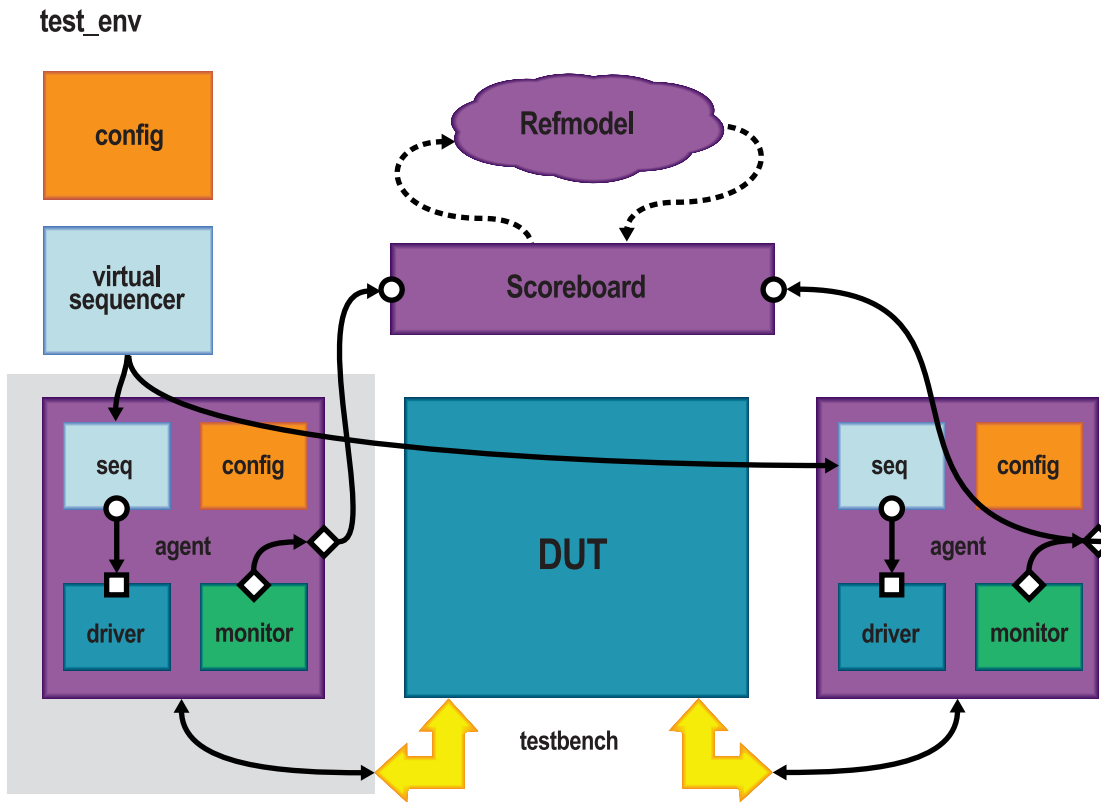


Figure 3: Dialog OVM test bench structure

In the rest of this section we look specifically at the methods that Dialog used to roll out adoption of OVM across their organisation. It should be noted that this was a multisite roll out. [1]

The following is the list of steps deployed by Dialog.

1. External training courses & workshops: As discussed above the fact that Dialog were adopting an industry standard methodology meant that there were external providers of training and workshops.
2. Internal seminars to consolidate knowledge & discuss best practices: Dialog were of course not starting from scratch. They had engineers with experiences from: previous jobs; attending industry events; reading online, press and papers, etc; external training and workshops. Dialog found it useful to consolidate that knowledge via internal seminars where they also discussed best practise.
3. Set of golden rules for implementing OVM environments: Distilled from the internal seminars and discussions, and from externally available materials.
4. Internal Wiki for knowledge sharing: This allowed Dialog to have a “living” knowledge base where engineers could add their growing knowledge and practical experiences of using OVM.
5. Using external resources on OVM: For example, the Mentor Graphics Verification Academy and TVS white papers on deploying OVM VIP.
6. Set of “template” components: These templates were a good way to encapsulate knowledge in a practical format and were distributed as an example environment
 - a) Demonstrate best practise
 - b) Form the basis of module environments
 - c) Scripts automated generation of environments from template code
7. Code review sessions: These were used to ensure that engineers started on the right track and give them feedback on their coding style. These were used to continuously improve the code being written and the environments being developed
8. Library OVC components created: An internal library was created containing components for potential re-use.
9. Standard VIP procured: For example, Dialog procured OVM compliant VIP from TVS.

PROBLEMS IN EXECUTION

Dialog overcame the following problems in their execution of OVM adoption.

1. Problems with implementing checking at the appropriate level of abstraction
 - a) Data & transformation checking in scoreboard
 - b) Timing and protocol checking with SVA
2. Problems with expending effort on coverage closure metrics
 - a) Appropriate priority setting on coverage goals
 - b) Keeping vplan “live” – linking coverage model and reviewing coverage model
 - c) If functional coverage is high, but code coverage is low then suspect the completeness of the model...
 - d) Achievable coverage goals (e.g. defining appropriate coverage buckets for a 32-bit real-time counters)
3. Problems with appropriate constraints on stimulus. For example,
 - a) Don't randomly toggle test mode signals in functional tests
 - b) Use highly-directed sequences required to arrive at system state of interest and then use CRV
Domain specific knowledge is usually required in these circumstances.
4. RTL-centric engineers found it hard to grasp ideas from the software world
 - a) Encapsulation
 - b) Inheritance
 - c) Polymorphism
 - d) The OVM factory & overrides
TVS have encountered this situation in numerous organisations. OVM often requires a software mindset which RTL engineers often find it hard to adopt.
5. Initially the OVM reuse paradigm was not fully leveraged. TVS have found that other companies overcome this through the use of a central VIP library and “librarian” who is able to actively market the VIP across the company and help engineers in the reuse of the VIP.
6. Reuse from module level to chip-level is non-trivial
 - a) Many issues were found when vertical reuse was done (requiring rewrite of module environments)
Dialog sees this as a learning process and is now achieving significant vertical reuse.
7. SV is a “rich” language, so there are many ways to code functionality. For example, it is not always obvious how to...

1. **Reusable Verification Components encapsulate a number of:**
 - **Agents**
 - **Scoreboards**
 - **Checks**
 - **Coverage**
2. **Verification Components shall be scalable for module, subsystem or chip level**
3. **Verification Components shall be located in SystemVerilog packages**
4. **Agents shall follow OVM guidelines**
5. **DUT interaction should not be done outside the agent**
6. **Synchronization should be handled with ovm_event_pool**
7. **Assertion-based checks should be placed in interfaces**
8. **TLM should be used wherever possible to communicate data**
9. **Configuration should be stored in a config component**
10. **Class and Module based code should only communicate via interfaces**

Figure 4: Basic Dialog OVM Guidelines

- a) make appropriate choice of language constructs
- b) write the interface between module-based code and class-based code
8. OVM is merely a library of parts and a user guide so a strong adherence to the methodology is needed. Dialog resisted the temptation to create Dialog-specific class libraries, macros or an OVM wrapper layer. This may ease the introduction of OVM but ultimately this approach is potentially restrictive and requires more ramp-up and longer-term maintenance. Dialog thus had a strong drive to stick with “vanilla” OVM and the approach suggested in the reference guide examples.
9. Dialog had an expectation that OVM would reduce their verification effort whereas the reality was that the effort increases (at least until reuse is leveraged and engineers become more familiar and experienced in its usage). The initial perception of the verification engineers was “why do I need to write so much code”? Dialog thus sees the main initial advantage of OVM as improved quality in verification. This will be of high

importance to Dialog going forward as the complexity of their designs increases. Dialog also expects to improve efficiency within a relatively short time period.

THE MAIN RESULTS ACHIEVED

In this section we first consider the main results achieved by Dialog in their rollout:

- The engineers were able to get running quickly with random stimulus generation and transaction coverage more quickly when they were supplied with reusable templates.
- Dialog did indeed reap all of the benefits normally associated with a Constrained Random Verification (CRV) methodology: high rates of bug discovery; easier tracking of real progress; managed closure driven by a metric-driven approach.
- Dialog was able to add coverage collected on directed tests (in passive mode) to that achieved via CRV. This was of high importance as it meant that Dialog were able to exploit the value contained in their legacy tests within their OVM environment

Dialog now has a well-defined approach to exploiting OVM and a roadmap for its deployment on future projects.

RECOMMENDATIONS

From their experience, TVS and Dialog would recommend the adoption of the vanilla OVM with a planned roll out taking most of the points from the Dialog roll out plan. Realistic expectations need to be set: expect improved verification but that improved efficiency will follow in time after the initial ramp up.

From experience through working with other adopters of OVM, TVS also recommends a multi-layered building approach to test benches and verification environments:

- Have OVM experts develop the main infrastructure and test benches.
- Use verification engineers to develop project sequences.
- Consider using non-verification engineers to develop the specific scenarios.

This allows engineers to work at different levels of the test bench according to their need to understand OVM and their project specific knowledge. For example, at one TVS client a software engineer used the TVS SDCard VIP to set up various scenarios to allow them to test corner cases in their driver code.

BACKGROUND

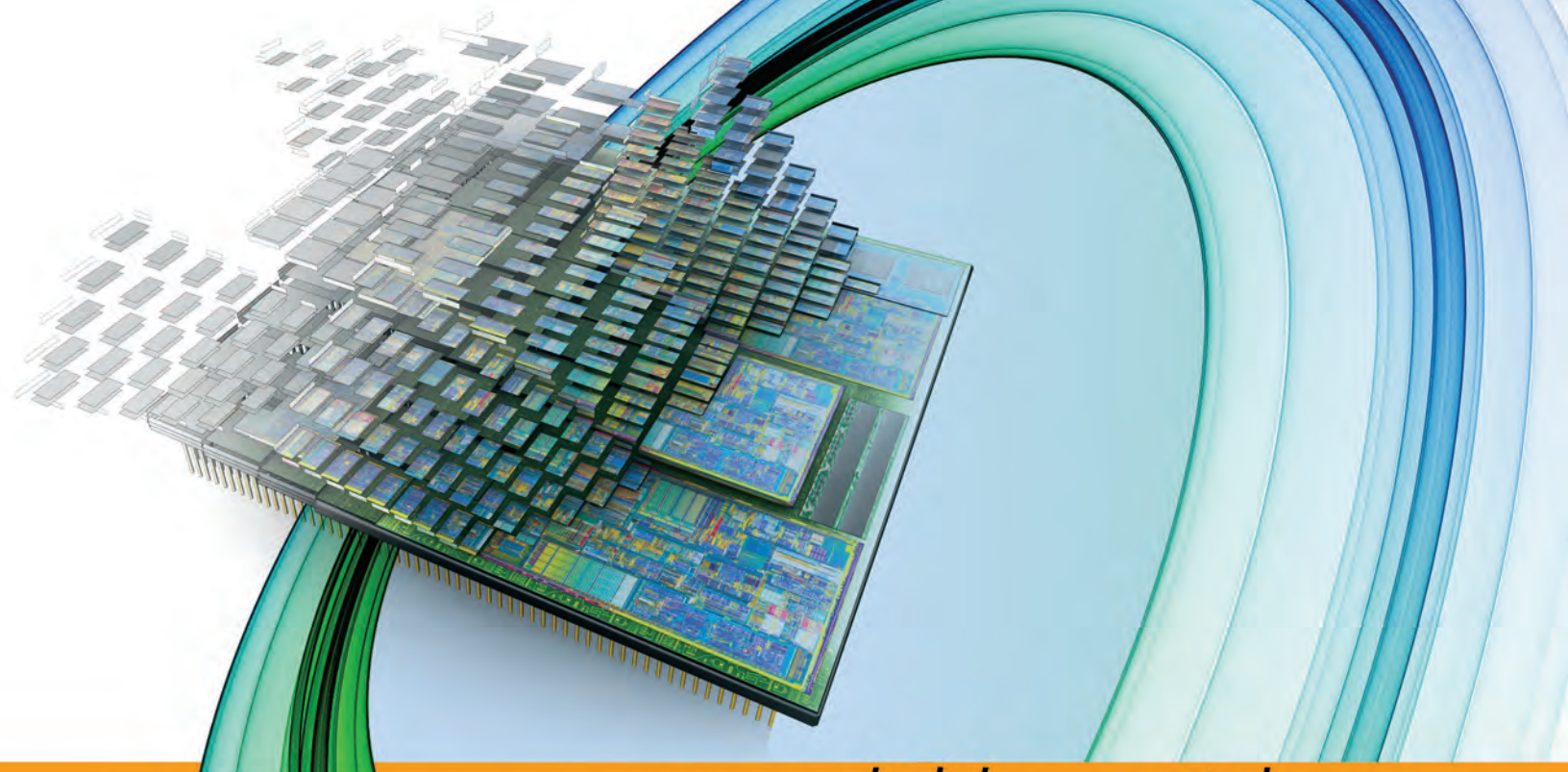
Test and Verification Solutions (TVS, Bristol, UK) delivers an independent testing service that not only reduces costs and time-to-market, but also improves product quality. TVS combines skills and experience in software testing, hardware verification and outsourcing to provide customers with an efficient, well-managed, quality assurance service. The company provides both consultancy and execution services using experienced engineering resources in several locations around the world.

Dialog Semiconductor creates energy-efficient, highly integrated, mixed-signal circuits optimised for personal portable, short-range wireless, lighting, display and automotive applications. TVS have been helping Dialog with their successful adoption of OVM.

For more information about TVS, please visit www.testandverification.com

END NOTE

[1] Dialog has development sites around the world including Asia and the US, as well as numerous countries in Europe.



verification HORIZONS

Editor: Tom Fitzpatrick
Program Manager: Rebecca Granquist

Wilsonville Worldwide Headquarters
8005 SW Boeckman Rd.
Wilsonville, OR 97070-7777
Phone: 503-685-7000

To subscribe visit:
www.mentor.com/horizons

To view our blog visit:
VERIFICATIONHORIZONSBLOG.COM

Mentor
Graphics[®]

www.mentor.com