

VMM Deployment: Experience From the Field

November 9th, 2005

Quinn Canfield

Senior Verification Specialist



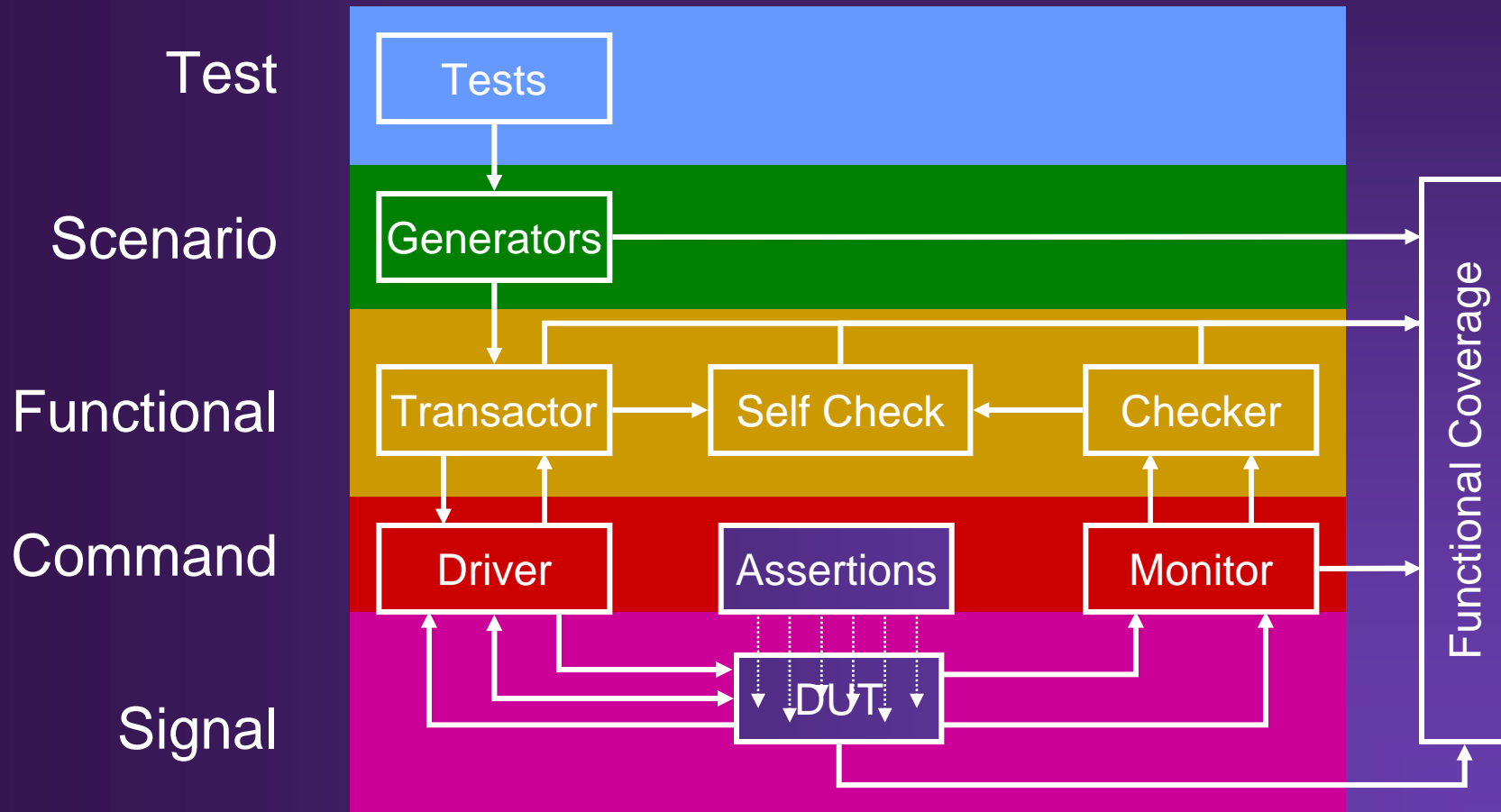
Background

- **Worked with a large number of customers**
- **Many different design types**
- **Wide range of backgrounds and training**
 - **Experienced TB people**
 - **RTL designers**
 - **Junior and senior level engineers**
 - **Different levels of OOP experience**

The Problem: Working with New Users

- **There are two distinct classes of users interested in VMM**
 - **Verilog/VHDL directed test users**
 - **HVL users interested in migrating to SystemVerilog**
- **Each group has different needs**
- **Each group will see value in different aspects of VMM**

Ideal Architecture



Users from the Directed Test World

- Majority of these users used Verilog or VHDL
- Typically not programmers
 - Little or no OOP experience
- They want to modernize their TB, but don't know how to do it
- VMM offers them:
 - Proven methodology
 - Base class library to get started
 - Building blocks
 - Interconnect

“Getting started is like drinking from a fire hose”

Users From the HVL World

- These users typically have used Vera, e, or C/C++
- Often programmers
- They know what has worked in the past
 - They also know what has not worked
- Higher expectations
 - Can't be a step backwards
 - Can't have big performance penalties
- Not looking to reinvent the wheel
- Looking for something “proven”
 - Battle hardened is even better

“We need more consistency between the various teams on the project”

Getting Started: Verilog/VHDL Users

- **Most of these users need basic OOP training**
 - Working with classes, inheritance
 - Polymorphism ? Sounds complicated..
 - What the heck is a “factory” pattern?
 - What is so special about callbacks?
- **Training is needed to help understand and apply the VMM methodology**
- **Users like the guidance provided by VMM**
 - Class reference tells them what is required
- **Normally start with:**
 - vmm_log
 - vmm_data
 - vmm_xactor
 - vmm_env

Getting Started: HVL Users

- **Most experienced customers are looking to improve:**
 - **Messaging**
 - vmm_log offers power and flexibility
 - **Building block connectivity**
 - Channels, callbacks, vmm_notify
 - Consistency is important
 - **Environment**
 - vmm_env defines a well structured environment blueprint
- **Consistent methodology and architecture from block to block *and* chip to chip**

Working with the Basics

- VMM methodology helps develop the block level TB diagram
- Design the data model (using vmm_data)
 - Composite or inherited data model?
 - What support methods do I need?
- Build the environment
 - How am I going to configure the TB and DUT?
 - How do I decide when the test is finished?
 - Do I need the ability to perform soft resets?

“We were able to get something up and running in just a few days”

Concerns about Overhead

- **Some HVL customers express concerns run time performance**
- **Some Verilog/VHDL customers wonder if this is “overkill” for block level testing**
- **These concerns are greatly out weighed by the benefits, but require a “leap of faith” for the first design**

Benefits Out Weigh Overhead Concerns

- **Ability to quickly find bugs and meet schedule deadlines should be the focus**
- **More automated testing**
 - **Focus on complicated areas of the design**
 - **Let the test bench find the “easy” stuff**
- **Code reuse will greatly improve ROI**
- **In the end, not a single RVM customer has gone back to their previous techniques**

Feedback from “Power Users”

- **VMM offers solutions for many of the difficult tasks users faced in the past**
 - **Messaging class**
 - **Interthread communication**
 - **Testbench restarts and resets**
 - **Error injection and other disturbances**
 - **Complex building blocks:**
 - **Broadcast, scheduler, scenario generator**

“Many of the guidelines that were presented sounded a lot like lessons we had learned the hard way “

Reuse: The Code. The People.

- **Customers like the idea of code reuse**
 - Reduces TB development on future projects
- **What is often overlooked is “human” reuse**
 - The ability for an engineer to understand other people’s code
 - Allows management to reallocate engineering resources which MUCH less ramp up time
- **VMM addresses both types of reuse with:**
 - Consistent, documented architecture
 - Well defined, documented base class

General Observations

- **Quick testbench development**
 - Participated in several code reviews
 - Impressed with how quickly verification teams put together GOOD test benches
- **Allows customers to focus on what is important**
 - One customer mentioned that they were now able to focus on finding bugs in the DESIGN
- **Fastest adoption of new technology I have seen!**

Conclusion

- Proven methodology
- Well documented
- Strong support from the user community
- Significant interest from the old school directed test community
- Decreases test bench development time
- Encourages reuse

“We would not have met our schedule without it”