

VMM Adopter Class

Advanced Level – 2 days

The Verification Methodology Manual for SystemVerilog (VMM) specifies a functional verification methodology, and defines the VMM Standard Library implemented in SystemVerilog. VMM includes constrained random stimulus generation, functional coverage collection, assertions, and transaction-level modelling. VMM's layered structure and channel-based communication model make it suitable for building both very simple and very complex functional verification environments.

Delegates for this course must start with a working knowledge of SystemVerilog, including its object-oriented programming (class-based) features. This course takes delegates through to full VMM verification project readiness by focussing on the verification principles and the in-depth practical application of the VMM using Synopsys VCS™.

Workshops comprise approximately 50% of class time, and are based around carefully designed exercises to reinforce and challenge the extent of learning. In the hands-on workshops, delegates will progressively build a complete VMM verification environment for a small example system.

Who should attend?

- Verification engineers who need to develop, deploy or configure SystemVerilog verification environments based on the VMM
- Design engineers who wish to make use of SystemVerilog's verification capabilities for test bench development using the VMM

What will you learn?

- The principles of effective functional verification using SystemVerilog
- How to understand the VMM Standard Library classes, documentation and examples
- How to build complete, powerful, reusable VMM-compliant verification environments

Prerequisites

A sound working knowledge of SystemVerilog, including some experience with its object-oriented programming features, is essential. For engineers new to SystemVerilog the Doulos **Comprehensive SystemVerilog** or equivalent, is an essential prerequisite. For team-based courses, precursor training in SystemVerilog can be tailored to the team's specific profile using our **Modular SystemVerilog** portfolio. Contact Doulos to discuss options that suit your needs.

Course materials

Doulos course materials are renowned for being the most comprehensive and user friendly available. Their style, content and coverage is unique in the HDL training world, and has made them sought-after resources in their own right. The materials include:

- Fully indexed course notes creating a complete reference manual
- Lab files comprising the complete SystemVerilog source files and scripts

For further information contact your local Doulos [Sales Office](#).



VMM Adopter Class

Advanced Level – 2 days

Structure and content

Introduction

Course structure • motivation • principles • benefits • overview of VMM testbench architecture

Verification Methodology

Assertion based verification • functional coverage • structural coverage • constrained random • coverage-driven verification • the verification process • verification planning • coverage models

Classes – OOP Refresher (if required)

Object-Oriented Programming • class • object • method • constructor • static members • inheritance • overriding • virtual method • up-casting • down-casting using `$cast`

Modelling transaction data

The VMM-compliant transaction data model • implementing a VMM data factory • copy and compare methods • using the `vmmgen` script

Connecting a testbench to the DUT

Connecting classes to the DUT: signal drive from a program • clocking, interfaces and modports working together • VMM-compliant connection to the DUT using virtual interfaces • active and passive command-level transactors

Building a simple verification environment

Transaction data at higher levels of the testbench: VMM channels • Creating transaction stimulus: the VMM atomic generator • Putting it all together: constructing a complete VMM environment • The environment's execution phases • Controlling testbench execution

Customising an environment

Introducing directed tests • VMM callbacks for checking and error injection • Callback façade classes

Monitoring and checking

Monitoring and self-checking: using callbacks to connect a scoreboard • Coverage sampling and transaction coverage

Communication and synchronisation

The notification service • *ONE_SHOT*, *ON_OFF* and *BLAST* notifications • Creating your own notifications • Conventional standard notifications in transactions and transactors • The message logging service • Message severity, verbosity and customisation • Building a logger hierarchy

Advanced stimulus generation

The scenario generator • writing your own scenarios • adding custom scenarios to the testbench • hints for creating robust scenario constraints • debugging and keeping track of scenarios

Further opportunities

Overview of other aspects of VMM: assertions, XVCs • verification IP under VMM • integrating legacy code • the register abstraction layer (RAL) and hierarchical environment composition

For further information contact your local Doulos [Sales Office](#).



VMM Adopter Class

Advanced Level – 2 days

Related courses

- Comprehensive SystemVerilog
- SystemVerilog for Design Groups (**see Modular SystemVerilog**)
- SystemVerilog for Verification Specialists
- Modular SystemVerilog (**in-house delivery only**)
- Comprehensive SystemC
- Expert SystemC Modelling
- Expert SystemC Verification
- Modular SystemC (**in-house delivery only**)
- Comprehensive C++

Project services

Doulos Project Services enable clients to access our world-leading technical know-how and apply it directly to projects. **Expert-on-call**, **Expert-design** and **Expert-support** options can be flexibly packaged and delivered to provide valuable expertise and additional resource just when it is needed.

How to book a course

To make a provisional booking, or to obtain pricing information, please contact your local Doulos sales team. You will find contact details on our [website](#).

Doulos acknowledges all trademarks and registered trademarks as the property of their respective owners.

For further information contact your local Doulos [Sales Office](#).

