

Compact VHDL Testbenches and Verification with OSVVM

Online Live

Workshop

Applicable Technologies	Requirements	Contact
None	Advanced knowledge of VHDL and digital circuit design	Michael Schwarz P. +49 7664 91313-15 E. info@plc2.de
Fee (net per person)	Inclusive	Duration
OL € 1,900	Training material	3 days
WO € 2,300	Plus beverages during breaks Lunch	3 days

Workshop

Today's FPGA and ASIC designs have drastically increased in size and complexity since the very beginning of digital hardware design. These elaborate circuits are described as a hierarchy of subsystems in hardware description languages like VHDL. The subsystems are most likely connected by standardized bus infrastructures like AXI, PLB, Avalon or WishBone. In addition, these systems might add a soft CPU IP core or an embedded Arm® CPU core. Such a design is way too complex to verify with simple, assertion-based testbenches.

With Open Source VHDL Verification Methodology (OSVVM) a structured approach is given, that increases the reusability of testbench codes. OSVVM is a free and open source available VHDL library that offers packages, data types, subprograms, and algorithms that are needed in almost every testbench. There is no need to reinvent the wheel again and again. The latest feature of OSVVM is a predefined set of verification IPs, so a wide range of standard bus interfaces is covered.

OSVVM is offering a methodology that comprises the following topics: Transaction-Based Modeling (TBM), self-checking, scoreboards, memory modeling, functional coverage, directed, algorithmic and

constrained random as well as intelligent testbench test generation. A VHDL testbench environment based on OSVVM is as powerful as other competitive verification languages like SystemVerilog or >e<.

This course starts with simple testbenches and increases the level of abstraction progressively. Along the way the students learn about: subprogram usage, libraries, file reading and writing, modeling issues, transaction-based testbenches, bus functional models, transaction-based models, record types, resolution functions, abstractions for interface connectivity, model synchronization methods, protected types, access types (pointers), data structures (e.g. scoreboards), directed, algorithmic, constrained random, and coverage-driven random test generation, self-checking (result, timing, protocol checking and error injection), functional coverage, representation of analog values and periodic waveforms, timing and execution of code, test planning, and configurations.

Due to the short course duration, we are focussing on the theory, that is needed to implement efficient VHDL testbench environment.

Agenda

- | | |
|---|----------------------------|
| 01. From basics to subprograms | 07. Functional coverage |
| 02. Transaction-Based Models (TBM/BFM) | 08. Execution and timing |
| 03. Elements of a transaction-based model | 09. Advanced coverage |
| 04. Data structures for verification | 10. Advanced randomization |
| 05. Creating tests | 11. Test plans |
| 06. Constrained random testing | |