UVVM vs OSVVM: The main benefits – and some history

Ever since the release of UVVM in 2015, UVVM and OSVVM have learned from each other and improved. UVVM is in fact now the fastest growing Verification Methodology in the world, independent of HDL. See below for more info. As given by the name 'Universal VHDL Verification Methodology', UVVM is providing the VHDL version of UVM, only a) far simpler to use and understand, b) a logical step-by-step evolution on VHDL, c) a gentle introduction to modern verification, and d) allowing low-cost simulators to be used. UVM is more advanced, but UVVM is sufficient and more efficient for almost all testbenches.

As we (the UVVM team) have been asked several times about the differences between UVVM and OSVVM, and as we have seen lots of misleading "information" on this subject, we finally decided that it was about time to give an overview of the many advantages of UVVM over OSVVM.

The Game changer

First of all, UVVM is a VHDL verification methodology and library that allows a significant verification efficiency improvement and at the same time allows a major design quality improvement. As UVVM is quite new, it was targeted at solving the following issues:

- Overview and Readability
- Maintainability, Modifiability and Extensibility
- Efficient and reduced Debugging
- Efficient Reuse at all levels

Additionally, UVVM had a very strong focus on simplicity where that matters the most for the users.

This made UVVM a real game changer and resulted in UVVM adoption rocketing in the VHDL community - becoming the world-wide fastest growing verification methodology of all (*1).

Some of the UVVM benefits

The beneficial value of various UVVM advantages do of course differ from one application to another and from one complexity level to another. This is a list of what we consider to be the most important benefits of UVVM for an average complexity DUT (module, FPGA, CPLD or ASIC). For more complex DUTs, the benefits increase almost exponentially, whereas for really simple DUTs, there are still significant benefits, and although they no longer amount to hundreds of hours, they could still be critical for the DUT quality.

Unique features in UVMM

- May control simultaneous activity on multiple interfaces from consecutive commands in one single test sequencer (process).
  In OSVVM this is not possible, and the recommended approach is to control separate interfaces from separate processes and then use semaphores or barriers to control the alignment between accesses on various interfaces. This looks trivial for a shown scenario of 3 interfaces and 3-5 statements for each interface but will result in complete chaos for a practical **far** higher number of statements per interface and in most cases also a higher number of interfaces. Then imagine the difficulty in keeping the overview when process A

line 30 is waiting for line 17 to be executed in process B and line 23 in process C, and before that process C line 12 is waiting for line 19 in process F, and so on…

- Specification coverage (aka Requirements Tracking) allows simple reporting of Requirements Traceability Matrix, which is required for safety and mission-critical projects, and in fact any project where quality is important.
- May command interface models to insert a given delay before a transaction, and thus skew one interface with respect to another. This is a critical feature to reach very error prone cycle related corner cases
- Very simple handling of split transactions and out-of-order protocols, as for instance the read-request executor process automatically passes the command on to the read-response executor process.
- The test sequencer can be set to automatically wait for either a given interface transaction to complete – or for all transactions on a given interface.
- There is a set of common commands that may be applied to any verification component
- Broadcast and Multicast may be used to control multiple verification components at the same time
- The command queues allow multiple commands to be sent to the same verification component at the same time
- May easily extend verification components with additional functionality like protocol checkers to increase the reuse value of the verification components
- Simple reuse
- Simplicity where it matters the most (when writing the test cases)
- May be used for FPGAs with lots of interfaces without losing the overview

Other major benefits of UVVM (over OSVVM)

- The largest collection of open source VHDL interface models available (BFMs and VVCs) (AXI4-lite, AXI4 Full, AXI-Stream Transmit and Receive, UART Transmit and Receive, SBI, SPI Transmit and Receive, I2C Transmit and Receive, GPIO, Avalon MM, Avalon Stream Transmit and Receive, RGMII Transmit and Receive, GMII Transmit and Receive, Ethernet Transmit and Receive, Wishbone)
- The readability and thus overview and understanding of the advanced randomisation in UVVM is at a completely new level – using brief explanatory keywords, making the meaning of complex constraints obvious to anyone
- The same readability and overview advantages for Functional coverage, including reports that are actually easy to understand.
- Functional coverage for transitions, which is really useful to check for a required sequence of values
- Lots of randomisations have been added to some of the interface models to allow far better error detection. An example is the randomisation flexibility and programmability for turning on and off the flow control in an AXI-stream interface. Lots of users have found bugs in their design with the help of this functionality.
- Developed in cooperation with ESA (the European Space Agency)
- Webinars on methodology in cooperation with both Siemens EDA (Mentor) and Aldec.
- Standardised VVCs and Test harness (in the sense that VVCs from different users will work together, and users will know how VVCs behave and how any test harness will work
    - Standard VVC interface, protocol, common commands, multicast and broadcast

    - Standard Configuration and Status interface
    - Standard VVC interface alignments and synchronisation
    - Standard VVC internal architecture. including queuing and control of checkers
    - Standard Debug support

General UVVM characteristics

- Free and open source – and available on both GitHub and IEEE Standards Association Open source project
- Tool agnostic. Runs on any VHDL 2008 compatible simulator
- Has BFMs, TLMs, VVCs, Scoreboards, Transaction Info, Free running watchdog, Activity watchdog, Error injector, …
- UVVM's user threshold is a fraction of the UVM threshold – for VHDL users
- UVVM allows a gentle introduction to modern verification

*1: According to Wilson research