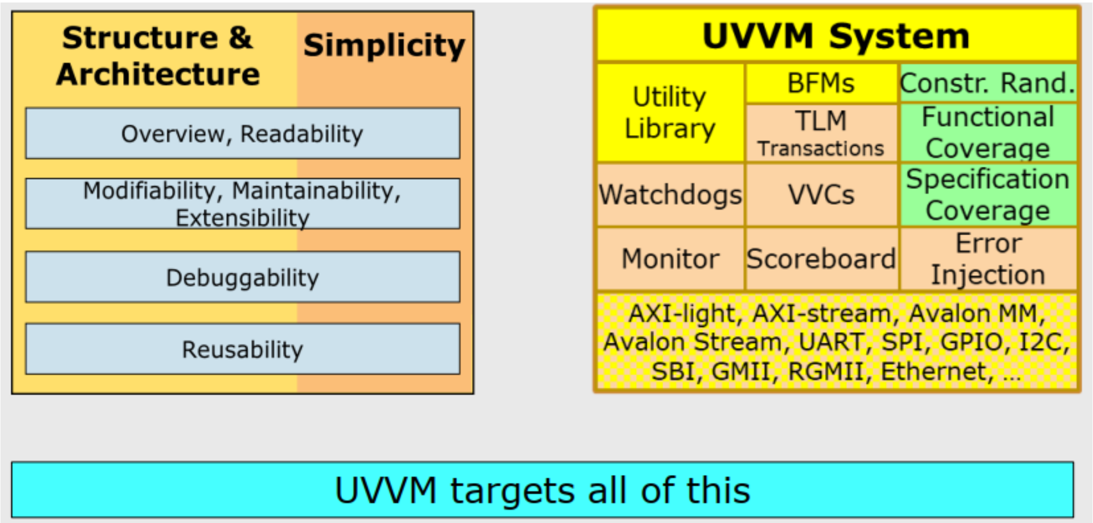
UVVM

UVVM (Universal VHDL Verification Methodology) is a free and open source methodology with the libraries. The purpose of that is improving the efficiency of verification and decreasing the verification time without no quality loss because almost half of the project time was spent on verification. The structured FPGA architecture serves for it with the features that are shown in below figure.

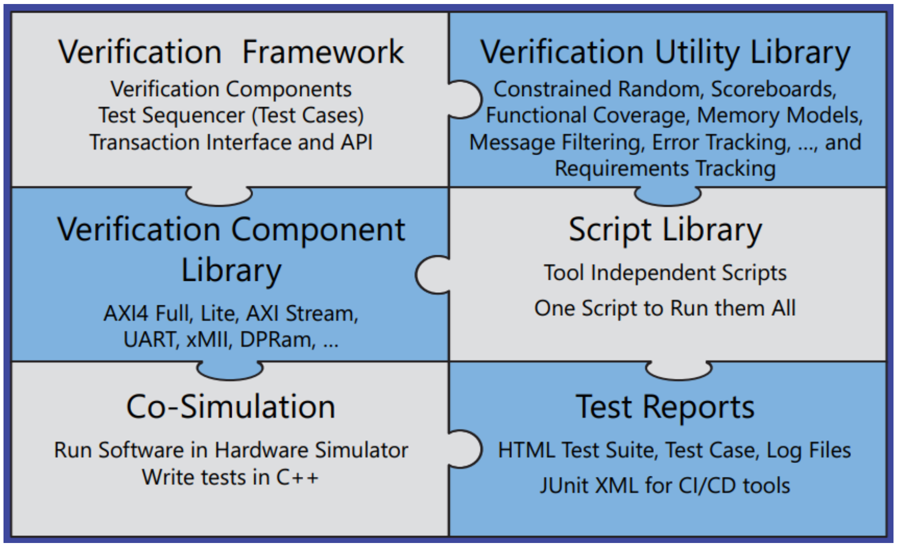


UVVM includes structures (functions, procedures etc.) for ease and reusability like clock generator, log, check value, Gen pulse, random, replace etc. in UVVM Utility Library. It has BFMs (Bus Functional Models) that make easier data transaction like SPI, UART, I2C, AXI etc. BFMs are simple and great for basic testbenches. BFMs uses procedures but a process can do only one thing at a time. For more thing, VC (Verification Component) is needed. It can be said that the VCs are uses BFMs (BFMs are sub-module of VCs) but do more things like cases on what to do, bit/frame rate or gap checker etc. VVC (VHDL VC) has some advantages like simultaneously running on multiple interfaces, synchronization of interfaces, reusability, queue, sequencing etc. UVVM also has scoreboard that checks values that come from DUT (Desing Under Test) and golden value (that is generated from a trusted model) and reports the results. The other capability of UVVM is specification coverage for the requirements of the design. The other coverage of UVVM is functional coverage. Functional coverage verifies the DUT's function according to given scenarios. Coverage reports can be shown with desired details. Randomization with constrained in a simple way also possible in UVVM to extend the verification. UVVM supports learning with well documentation, free webinars, testbench examples etc.

With all that, UVVM provides fundamental blocks to easy, reusability and improving the verification.

OSVVM

VHDL is the most widely used HDL (Hardware Description Language) worldwide and half of the VHDL FPGA users are use OSVVM for verification methodology according to https://blogs.sw.siemens.com/verificationhorizons/2022/11/21/part-6-the-2022-wilson-research-group-functional-verification-study/. OSVVM (Open Source VHDL Verification Methodology) is free, open source verification methodology with framework (verification components, test cases etc.), utility library (constrained random, scoreboards, functional coverage, etc.), script library (tool independent) and extensive test reports.



Verification Framework
Verification Components
Test Sequencer (Test Cases)
Transaction Interface and API

Verification Utility Library
Constrained Random, Scoreboards, Functional Coverage, Memory Models, Message Filtering, Error Tracking, ..., and Requirements Tracking

Verification Component Library
AXI4 Full, Lite, AXI Stream, UART, xMII, DPRam, ...

Script Library
Tool Independent Scripts
One Script to Run them All

Co-Simulation
Run Software in Hardware Simulator
Write tests in C++

Test Reports
HTML Test Suite, Test Case, Log Files
JUnit XML for CI/CD tools

Verification components implement interfacing signaling, test sequencer calls transaction (test cases). Transaction interfaces equal records, transaction API (Application Programming Interface) equals procedures.

Some protocols do same things, MIT (Model Independent Transaction) exists for these. MIT speed up VC development, test case development and documentation.

VCs comprise of interface (DUT, transaction) and process that make easier to verification and any VHDL engineer is capable of coding VC. Test sequencer controls the test structure with concurrent process per interface (just like design). Also directed test, constrained random, scoreboards, functional coverages are easy for use and reports are supported. Verification with requirements, alerting and logging are also supported. Randomization is more realistic than directed test and constrained randomization is more valuable. This is also supported in OSVVM. Scoreboard structures make easier to checking values of expected and actual. Functional coverage with code coverage determines that testing is done. Scripting without tool dependent is supported in OSVVM. OSVVM has a wide and detailed reporting of verification results include coverages, test cases, requirements, alerts, scoreboards etc. Reports consist of the main summary, and it can go deeper with details. HTML or XML file type is also supported for reports. OSSVM also has many resources to support and learning.

Note: OSVVM is developed by the same VHDL experts who have helped develop VHDL standards.