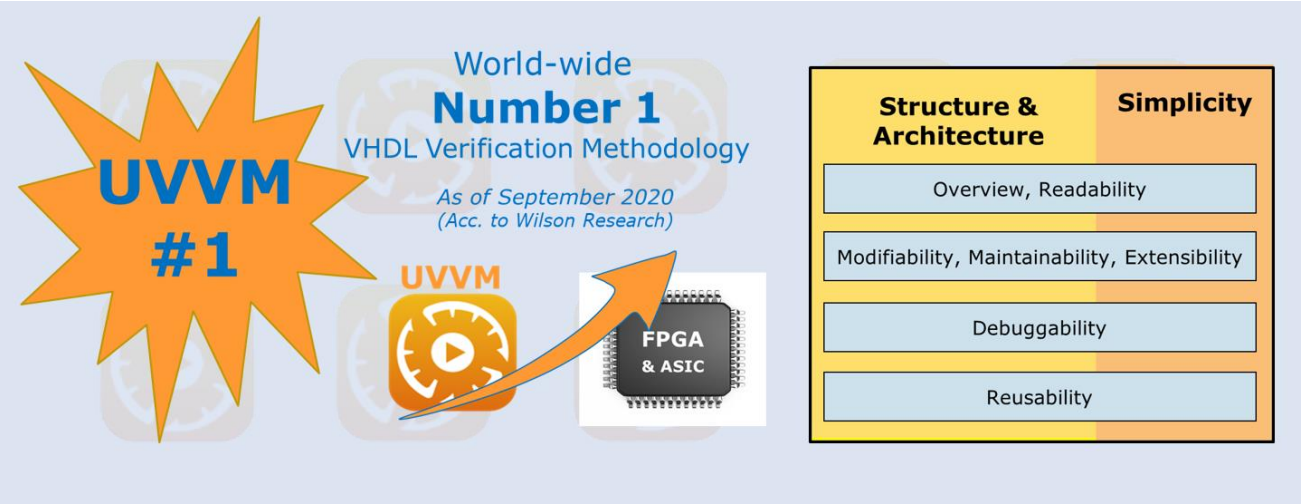
EmLogic

Embedded Software
Hardware
FPGA

# Advanced VHDL Verification – Made simple

## Learn the world-wide #1 VHDL Verification Methodology from the creator and main architect of UVVM

*Efficiency and quality are both a question of overview, readability, extensibility, maintainability, and reuse, - and a good architecture is the answer.*
*This applies for both Design **and** Verification*




## Description

On average half the development time for an FPGA is spent on verification. It is possible to significantly reduce this time, and major reductions can be accomplished with minor adjustments. This is an intensive 3-day course on how to reduce development time and at the same time improve the quality.

The main differentiators between this and other similar courses are the focus on simplicity and the very structured approach to reuse - also inside a single project. We have seen and heard of many complex testbenches by various designers. A major problem with most of these testbenches seems to be that it gets too complex for everybody apart from the VHDL expert who designed it, – sometimes a person with a far more than average interest in the language or system details.

This course is based on the principles of 'maximum cohesion & minimum coupling' and 'Divide and Conquer', where the test case writer doesn't have to know anything about the testbench implementation details, and the testbench implementer has a structured architecture all the way down. This approach to VHDL testbenches typically leads to man-hour savings of 20-60% and more, and is unique for this course.

esa

*Development of new UVVM features and functionality is done in cooperation with ESA. The ESA-UVMM-projects are initiated in order to improve efficiency and quality of FPGA VHDL verification.*

# Main Benefits

The main goal of this course is to show how you can achieve a far better testbench overview, readability, extensibility, maintainability and reuse – resulting in better quality and faster development.

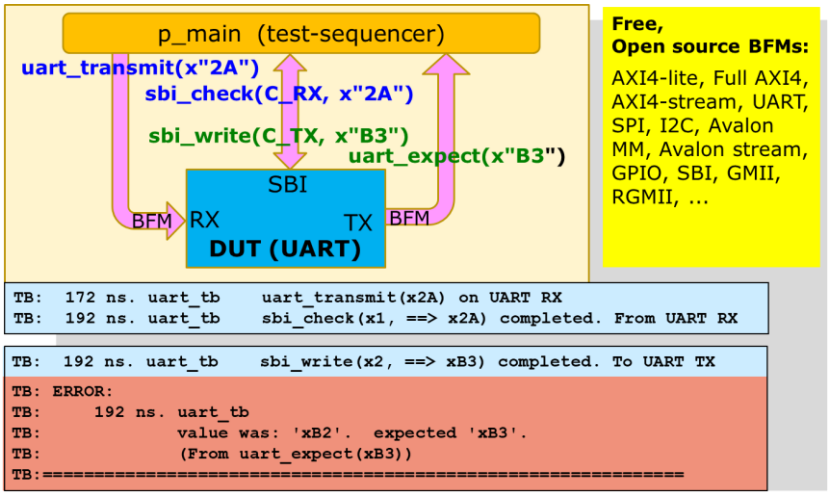
- **Overview:** So that anyone can easily understand the overall structure – just like a simple block diagram
- **Readability:** So that anyone can read the test driver and from that understand exactly what is going on
- **Extensibility:** Allowing fast and simple testbench changes to check new design modules and functionality
- **Maintainability:** Allowing fast and simple testbench changes to adapt to design changes
- **Reuse:** Between modules in a project; From modules to top-level in a project; From one project to another

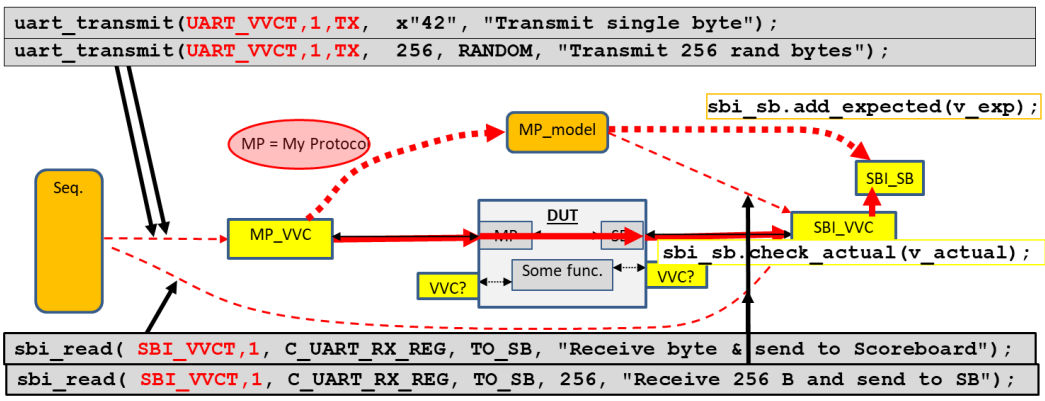
# What you will learn

It is actually possible for almost all companies to speed up the FPGA verification and at the same time improve the FPGA quality and fault coverage. Learn how to build your testbenches in a structured way, which is the key to overview, readability, extensibility, maintainability and reuse. Theory is mixed with practical examples and hands-on tutorials. The course will also cover important general verification issues like:

- Using sub-programs and various important VHDL constructs for verification
- Handling simple verification in a simple manner
- Making self-checking testbenches
- Using logging and alert handling
- Applying standard checkers for value and stability, and for waiting with a timeout for events
- Using simple procedure based transactions like uart_transmit() and avalon_read() for simple verification scenarios
- Making your own Bus Functional Model (BFM) – and adding features to speed up verification and debugging
- Getting a kick start on BFMs with UVVM's open source BFMs for Avalon, AXI4-full(lite/stream, UART, SPI, I2C, etc
- Making directed or constrained random tests – and knowing where to use what - or a mix
- Applying specification coverage (aka requirements coverage)
- Using verification components and advanced transactions (TLM) for complex scenarios
- Target data and cycle related corner cases and verifying them
- Learning to use UVVM to speed up testbench writing and the verification process
- Getting a kick start on your testbench by using available UVVM Verification components for AXI4 full/lite/stream, Avalon MM, SBI, SPI, I2C, UART; - and use these as templates for your own VVCs
- Making a new UVVM Verification component in 30 minutes
- Understanding and using Scoreboards and models
- Making an easily understandable and modifiable testbench

even for really complex verification – and do this in a way that even SW and HW developers can understand them




See further down for a day-to-day agenda.

## Guided Labs

The course will be approximately 50% theory and 50% hands-on. You will start by building a rather simple self-checking testbench, and then add procedures to simplify it, add Bus Functional Models to access your interfaces and add important functionality to these to speed up the verification process. Then you will write a test sequencer to control already available VHDL Verification Components (VVC) and experience how very easy that is. The next step is to add commands to an existing VVC and control constrained random stimuli and specification coverage, and finally you will generate and adapt your own VVC from scratch. Using a scoreboard and a model is also included in the labs.
All participants must bring their own PC with their own preferred simulator (Questa, ModelSim, Riviera-PRO or Active-HDL). This allows an easy continuation after the course is finished.

## UVM for VHDL - only much simpler

To be fair - UVM is significantly more advanced than UVVM in some areas, but for the huge majority of testbenches that extra functionality does not yield higher efficiency or quality. In fact, that extra complexity most often results in slower development and a verification system that is too complex for most of the FPGA development team. UVVM on the other hand is just a logical extension on VHDL, and thus a logical extension for VHDL designers, who can implement this step by step - when they see the need for more functionality.

UVVM also allows VHDL developers to continue using fairly inexpensive VHDL tools rather than the much more expensive SystemVerilog and UVM related tools, and includes the following:

a) the most important UVM functionality, but extremely simplified for the user,
b) the modular approach of a good FPGA design, with a hierarchical testbench structure that mirrors the design structure,
c) a standardised VHDL testbench architecture and a standardised VHDL Verification Component architecture,
d) a pure VHDL approach
e) the lowest possible user threshold for the functionality you need
f) no lock-in, but the user can pick any one or more procedures, functions, BFSs, VVCs, and support functionality
g) randomisation and specification coverage

UVVM will be used as example throughout the presentations and labs, but the principles taught and shown are general state of the art VHDL verification methodology. Hence you basically get 2 courses in one:

- General VHDL verification course with best practices for making good testbenches
- UVVM course – enabling the best possible VHDL testbench infrastructure and architecture
-

# Target participants

The course is aimed at FPGA designers and verification engineers with a good knowledge of VHDL and some experience with VHDL testbenches and verification. You should also have working knowledge with Questa, ModelSim, Riviera-PRO, Active-HDL or other simulator.
(The course if equally relevant for ASIC designers using VHDL for verification)

# Agenda

*(Note: This agenda is for the 3-day classroom course. The 5-day online courses will be slightly different but covering exactly the same material. The course may be altered slightly from case to case)*

Day 1:

Making a Simple, Structured and Efficient Testbench, Step-by-step
- Testbench basics and basic testbenches, Building a simple TB from scratch, Simple TB architecture
- Testbench infrastructure, Logging, Alert handling, Checking values, Waiting for events with timeouts
- Procedures and functions, When to make subprograms, Using subprograms in TBs
- Simple randomisation, Transactions and Bus Functional Models

Advanced TB Infrastructure aspects
- Advanced logging and verbosity control, Advanced alert handling
- Pulse and clock generation, Advanced waiting-procedures
- Process synchronization between two or many processes

Making good BFMs (Bus Functional Models, Transactions)
- Good BFMs, TB and BFM synchronization, Making flexible subprograms,
- Normalisation and sanity check of subprogram inputs, Configuration of BFM behaviour
- Subprogram parameter simplifications

Introduction to more advanced TB architectures
- Regular and irregular data streams
- Parallel activity, Corner cases, Challenge of verifying a UART with corners

Labs: TB infrastructure and BFMs
- 1: Getting started, 2: Simple BFM, 3: Advanced BFM
- Procedures, logging, alerts, checkers, test harness, simple data communication using BFMs

Day 2:

Advanced TB architectures
- Limitations of typical TBs, Corner case categories, A structured TB architecture,
- Transaction level models, VVCs (VHDL Verification Components)
- Explaining a VVC, BFMs vs VVCs, VVC architecture, VVC reuse
- Structured TBs, Overview, Readability, Maintainability, Extensibility and Reuse
- VVC structure and extensibility, Handling multi-thread interfaces
- Making simple high level test case commands, Debugging, Hitting the corner cases
- Value and cycle related corner cases, Randomisation, Local sequencer
- Understanding and using Scoreboards

UVVM details (and general good verification methodology)
- Transaction commands – step-by-step, Command distribution
- Delta cycles, Non time consuming commands
- Commands queuing, VVC commands, Execution of VVC commands
- Controlling a VVC based TB, Procedure naming and parameters
- VVC configuration and status, Multicast and Broadcast
- Large data transfers, Handling split transaction interfaces (and out-of-order)
- BFM code walkthrough, VVC code walkthrough

Labs: UVVM VVC Framework
- 4: Getting started, 5: TB sequencer, 6: Making new commands
- Initiation, Logging in a VVC framework, Transaction commands
- Register access and UART examples, Simple synchronization, Simultaneous commands
- Adding VVCs to TB, Adding new commands to VVC, Controlling randomisation, Scoreboards

Day 3:

Randomisation, Functional Coverage, Specification Coverage – and Code coverage
- Code coverage, Functional coverage, Directed vs Random
- Advanced randomization explained and exemplified (using 3rd party open source plug-in)
- Functional coverage explained and exemplified (using 3rd party open source plug-in)
- Using randomization and functional coverage inside UVVM
- Specification Coverage / Requirement coverage

Various verification issues
- Documentation, Verification strategies, Verification approaches
- BFM and VVC vs SW for CPU access,
- Assertions, Checkers, Monitors
- Simple and advanced scoreboards
- Specification Coverage

Reuse, Roadmap and Summary
- Testbench reuse at module and FPGA level, Reuse efficiency
- Debugging a VVC based TB, Roadmap for UVVM
- Efficient vs inefficient verification

Labs: UVVM VVC Framework
- 7: Making built-in checkers, Randomisation and Functional coverage, Making a VVC from scratch
- Make checker, Control checker
- Constrained randomisation, Randomisation inside a VVC, Intelligent random stimuli (using 3rd party plug-in) Automatic coverage, Controlling randomisation and coverage
- Generate a new VVC template, Adapt VVC to your interface, Add commands
- Make a model of the DUT and use this with scoreboards

## Doulos recommends UVVM for VHDL testbench architecture

In a recent webinar Doulos presented several really good reasons for using UVVM as a basis for making structured and reusable VHDL testbenches. The quotes below are from John Aynsley, CTO in Doulos.

**'the standardized testbench architecture that you find in UVVM is really useful for reuse'** as you can in fact reuse a major part of your testbench from one project to another and reuse the verification components exactly as is.

**'within this framework you've got as much flexibility as you need'** because you can define the test harness as you like, use any part of UVVM that you like (for instance just one or five specific VVCs) and mix and match with any legacy code or 3rd party verification IP.

**'each VVC has a standardized structure'** and this common architecture allows all VVCs to be easily understood, integrated, controlled and extended, - and of course makes it very simple to make new VVCs.

**'It's relatively straight forward to write your own VVC', --- 'the first time it took me a little longer than 30 minutes to make my own VVC from scratch, .. but not much longer (given that the BFMs exist)'**, --- **'(using the provided Python script...) this really is straight forward'**
**'Adding that monitor was a pretty straight forward thing to do'** - thus showing that extending the VVCs with new functionality and new commands is quite easy.

**'The UVVM structure is quite simple'** - stating and showing that detailed knowledge of UVVM is not at all necessary to make new VVCs, make new structured testbenches or write good test sequences using the high level VVC commands. UVVM really makes it much easier and faster to make good VHDL testbenches. The UVVM based testbenches are the best possible with respect to structure, understanding, maintainability, extensibility and reuse.

# Presenter

Presenter and lab instructor at this course is Espen Tallaksen, the main architect behind UVVM.

Espen is the CEO and founder of the newly established EmLogic and previously also Bitvis, both independent design centres for embedded software and FPGA, - with Bitvis as a leading Nordic company within its field and EmLogic soon to be. He graduated from the University of Glasgow (Scotland) in 1987 and has 30 years' experience with FPGA and ASIC development from Philips Semiconductors in Switzerland and various companies in Norway. During twenty years Espen has had a special interest for methodology cultivation and pragmatic efficiency and quality improvement.
One result of this interest is the UVVM verification platform that is the #1 VHDL verification methodology and library world-wide, and in fact the fastest growing FPGA verification methodology independent of HDL.

He has given many presentations and keynotes internationally on various technical aspects of FPGA development, including lots of hands-on tutorials and presentations at FPGA-Kongress every year since 2016; - all with a crowded audience and great feedback. He is also giving courses world-wide on how to design and verify FPGAs more efficiently and with a better quality.
The fundamental message is always the same: Overview, Readability, Extensibility, Maintainability and Reuse are the key elements to Quality and Efficiency. Overly complicated design or verification systems should be avoided – even when they are structured. Simplicity – to the extent possible - should always be the target for any challenge.

# More info:

Invited Paper DATE2019 / OSDA in Firenze 2019: As an introduction
UVVM — The Fastest Growing FPGA Verification Methodology Worldwide!

See also the following Webinars via Mentor and Aldec:
https://www.linkedin.com/pulse/introduction-efficient-vhdl-verification-espen-tallaksen/
https://www.linkedin.com/pulse/uvvm-steps-up-gear-review-some-new-features-vhdl-espen-tallaksen/
https://www.linkedin.com/pulse/advanced-vhdl-verification-made-simple-free-webinar-espen-tallaksen/

# Registration:

If you received an invitation to this course from one of our course partners, please register via them (unless you have been told otherwise).

Otherwise please register to info@emlogic.no and give the following information: Name, Company name, Company Address, Email, Mobile, – and if required by your company a Purchase order reference.