# Advanced VHDL Verification – Made simple

*Efficiency and quality is all a question of overview, readability, extensibility, maintainability and reuse, - and a good architecture is the answer.*

## 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, – often 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.

## 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 or Active-HDL.

## Main Benefits

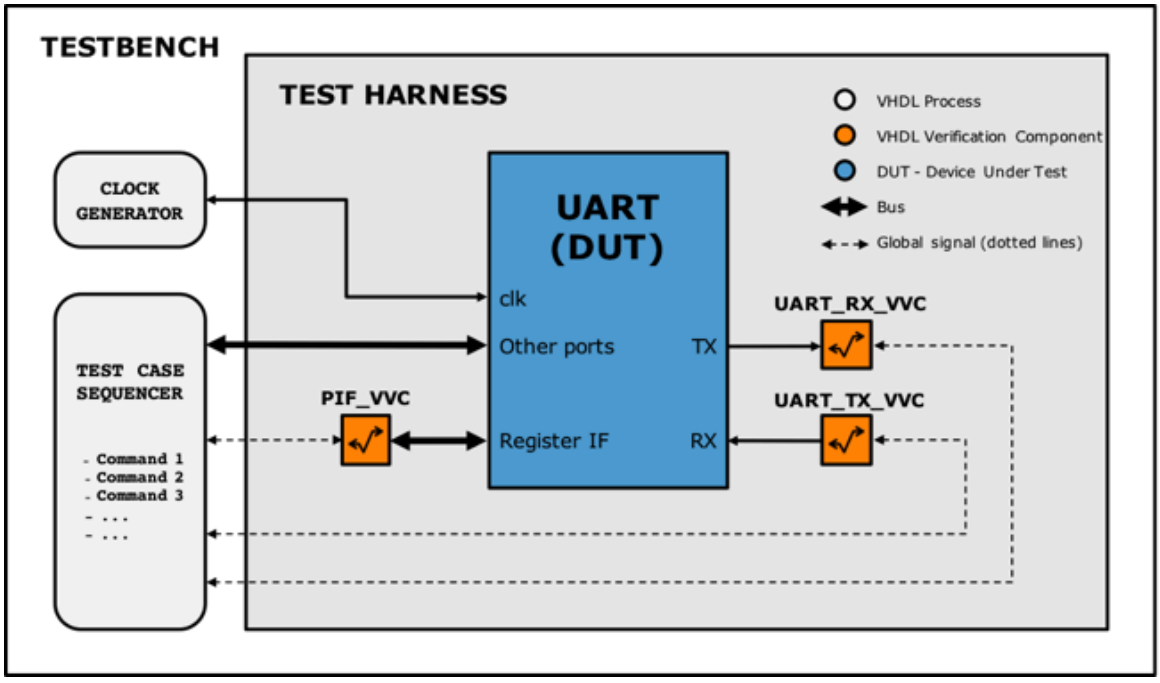
The main goal of this course is to show how you can achieve a far better testbench overview, readability, extensibility, maintainability and reuse – resulting in better quality and faster development.

- **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

## You will learn

This course focuses on FPGA- verification and how you can build your testbenches in a structured way. 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 value and stability checkers and 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-lite/stream, UART, SPI, I2C, etc.
- Making directed or constrained random tests – and knowing where to use what - or a mix
- Learning to use the most important features from OSVVM for randomization and functional coverage
- Applying coverage driven tests in a controlled manner
- 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-lite, AXI4-Stream, Avalon MM, SBI, SPI, I2C, UART; - and use these as templates for your own VVCs
- Making a UVVM Verification component in 30 minutes.
- 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.




## Guided Labs

The course will be approximately 50% theory and 50% hands-on. You will start by building a rather simple self-checking testbench, add procedures to simplify it, add Bus Functional Models to access your interfaces and add 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 extremely easy that is. The next step is to add commands to an existing VVC and control constrained random stimuli and functional coverage, and finally you will generate and adapt your own VVC from scratch.

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.

# UVVM (and OSVVM)

UVVM (Universal VHDL Verification Methodology) also called UVM for VHDL is an open source VHDL Verification Methodology and Library. UVVM in combination with constrained random and functional coverage from OSVVM combines the best of two worlds:

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 pure VHDL approach, where the user can just pick the functionality needed from well documented VHDL libraries, and
d) the lowest possible user threshold for the functionality you need

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. OSVVM will be used as example for constrained random and functional coverage. Examples will also show how OSVVM can be used **as is** inside and in parallel with UVVM.

# Presenter

*Espen Tallaksen*

Presenter and lab instructor at this course is Espen Tallaksen, the main architect behind UVVM. Espen is also the director and founder of Bitvis, the leading design centre for Embedded SW and FPGA in Norway.

He has 30 years' experience from FPGA/ASIC development in various companies in Norway as well as NXP in Switzerland. He has given lots of courses on both Design and Verification, and technical presentation on various conferences around Europe. 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:

- Part 1: The testbench architecture
- Part 2: The testbench sequencer
- Part 3: The VHDL Verification Component (VVC)

# Registration:

See www.bitvis.no : Events
For course in Oslo (Asker) March 7-9, 2017: Se here