Advanced UVM Tutorial - Taking Reuse to the Next Level

This tutorial provides intermediate and advanced users of the Universal Verification Methodology (UVM) with comprehensive in-depth material on all aspects of achieving vertical and horizontal verification reuse using UVM. The content is derived from Verilab's extensive experience in solving reuse issues for many different clients, projects and applications using a variety of verification languages and includes pragmatic guidelines as well as real-world examples. It is aimed at engineers with a good understanding of SystemVerilog and practical experience in either OVM or UVM. UVM beginners may also find the content interesting although the details could be somewhat overwhelming, at any rate it should raise awareness and provide some good reference material for the future.

After a very brief introduction to UVM in order set the scene and put the other topics into context, the tutorial takes a detailed look at reuse concerns such as structural architecture, user-focussed encapsulation, automatic adaptation and tuning of stimulus, checks and coverage based on static and non-static configuration fields as well as required roles of the corresponding agents in the enclosing verification environments. The material is grouped into the following topics:

- ***Vertical & Horizontal Reuse of UVM Environments***
- ***Self-Tuning Functional Coverage - Strategy & Implementation***
- ***Adaptive Protocol Checks - Configuration Aware Assertions***
- ***Configuration Object Encapsulation & Appropriate config_db Usage***
- ***Parameterized Classes, Interfaces & Registers***

A deeper understanding of how these key reuse requirements are implemented in UVM provides the verification engineer with the expertise necessary to go beyond the beginner level and excel at applying UVM reuse concepts to a wide variety of applications. The focus is very much on pragmatic real-life working solutions, but we go behind the scenes of UVM where required in order to highlight the mechanics and demonstrate how it all fits together.

Duration – 3 hours.

Organizer:
- Mark Litterick - Verilab GmbH, mark.litterick@verilab.com

Speakers:
- Mark Litterick
- Jason Sprott
- Jonathan Bromley

Vertical & Horizontal Reuse of UVM Environments

Many of the architectural defects in UVM environments come from a failure to understand the impact of reuse requirements that can be both invasive and demanding. This section discusses vertical reuse in a generic manner and then provides specific UVM-based examples of how to achieve these reuse goals with a minimum of effort ensuring consistent repeatable results. We also point out where reuse is necessarily limited and perhaps not worth the effort to support in the base code for a UVC (and explain how the user can layer it on top). We conclude by describing how the supplier can validate vertical reuse compliance in a pragmatic and stand-alone manner with no access to the user's environments.

Self-Tuning Functional Coverage - Strategy & Implementation

It is surprisingly straightforward to implement a functional coverage model that provides an overwhelming amount of irrelevant information, misleading results and incomplete metrics - a problem that is not unique to UVM. This section highlights the issues and provides pragmatic guidance on what to look out for, how to detect problems, and how to repair the defects. In addition we also demonstrate some cool implementation solutions for UVM components, including advanced use of functions and automatic tuning of coverage classes to run-time configuration (working around the SystemVerilog restriction that forces covergroups to be constructed with new() prior to build-phase conclusion).

Adaptive Protocol Checks - Configuration Aware Assertions

Functional checks are by necessity distributed throughout UVM components; these can often be poorly encapsulated, isolated from other aspects of UVM operation, difficult to control, and have missing coverage. This section discusses the responsibilities of the various components for transaction content checks in monitors, transaction relationship checks in scoreboards and signal protocol checks in interfaces - with a focus on error isolation, controllability, easy coverage collection and operation in a passive role for vertical reuse scenarios. The adaption of SVA checks located in the interface to dynamic configuration field changes in the class-based world is also presented.

Configuration Object Encapsulation & Appropriate config_db Usage

Intermediate user's of the UVM are often confused about how to structure and encapsulate configuration fields into object classes, or discrete entries in the config_db. This section provides detailed pragmatic guidelines derived from large-scale client projects, including how to correctly encapsulate hierarchical configuration objects to ensure accuracy and enable effective distribution of values throughout the environment, while providing a consistent user API to the test layer. The build-order for these objects is discussed in relation to UVM phases, and the propagation of object handles and appropriate configuration fields using the config_db is demonstrated.

Parameterized Classes, Interfaces & Registers

The use of the same environment for several different DUT variations is often achieved using parameterization, but in UVM this can lead to complications. This section provides a detailed explanation of how to setup and use parameterized classes and interfaces in UVM environments. The structural challenges that come with parameterized type propagation to all user classes, correct use of appropriate macros, and common debugging problems are highlighted. In addition we also demonstrate a pragmatic solution to working with parameterized register models, which are not components, but rather objects, in the UVM world and therefore need a slightly modified approach.