Be a Sequence Pro to Avoid Bad Con Sequences

Speaker & Organizer: **Mark Litterick** - *Verilab, GmbH*

Contributers: **Jeff Vance** - *Verilab, Inc.,* **Jeff Montesano** - *Verilab, Inc.*

Tuturial Motivation

The Universal Verification Methodology (UVM) is widely accepted as the industry standard for verification of chip designs. While UVM has many features to help with the challenges of verifying complex designs, UVM sequences for stimulus arguably have the most impact on verification quality. Despite the maturity and widespread success of UVM across many projects, evidence has shown that UVM sequences are typically not applied to their fullest potential. This is largely due to a lack of published resources with guidance on how to define large-scale sequence libraries to solve more complex verification challenges. The consequences are seen in typical projects where large collections of ad-hoc sequences create additional problems for engineering teams instead of helping them tape out bug-free designs on schedule. These challenges include additional unnecessary complexity, insufficient control of stimulus for tests, and limited reuse.

In addition to the general problems of ad-hoc sequence libraries, UVM sequences are vital to success in several ways that are often overlooked. There is little guidance on how to apply sequences for controlling continuous streams of data over time. This includes digital streaming data, analog stimulus, and clock generation. Sequences also serve as an API to engineering teams who must create verification plans, manage test regressions, triage debug tasks, report project status, and manage risks to meet schedules. An inadequate sequence API to support these tasks frequently creates workflow bottlenecks as teams struggle to transfer knowledge, isolate problems, and transparently report status to project stakeholders.

Finally, UVM sequences will continue to be used by teams that are considering adopting the newer Portable Stimulus methodology. In order to successfully integrate high-level portable stimulus with a simulation-based testbench, an adequate UVM sequence API is vital.

This tutorial addresses all these problems by providing guidelines based on extensive project experience applying UVM sequences in a variety of situations, from small block-level testbenches to complex integrations with challenging reuse and usability demands.

Tuturial Content

This tutorial provides a comprehensive overview of sequences and how we use them to orchestrate effective constrained-random stimulus in UVM testbench environments. The operation of these sequences on virtual and physical sequencers is presented in the context of proactive masters, reactive slaves and autonomous data stream generators. Guidelines for encapsulating sequences, architecting sequence libraries, managing complexity and enabling reuse are also provided. Techniques to maximize project productivty and improve progress tracking by leveraging the sequence API are discussed as well as the relationship between the sequence architecture and portable stimulus extensions using PSS.

- **Sequence Library Guidelines**

We outline a sequence API layering strategy that provides better encapsulation for UVCs, reduces complexity for test writers, and allows better reuse of sequences within a sequence library (and between derivative projects). We then give implementation guidelines that most impact the problems of complexity, control and reuse in a sequence library. These guidelines include a comparison of different constraint model techniques and recommendations for which are most ideal for certain testbench goals.

- **Sequences for Streaming Data**

We give guidelines on controlling continuous streams of values over periods of time. Examples include clock generation with jitter and skew, steaming data to external digital interfaces, and analog stimulus. In these cases, we need fine grained control over the input patterns. This allows us to validate interesting combinations of stimulus that might otherwise be hard to do from a UVM sequence.

- **Improving Verification Productivity**

In addition to managing the technical challenges of large sequence libraries, we show how to use sequences to improve verification productivity. We show how to use sequences to isolate design features for easier debug, regression management, and better coverage control. These techniques directly improve the visibility of project status to stakeholders and allow flexibility to adapt to changing schedules.

- **Portable Stimulus Considerations**

We show how a well-designed sequence API is vital for adoption of a Portable Stimulus workflow. Teams who are currently using portable stimulus, or are considering using it in the future, can save time and effort by ensuring upfront that their sequence library directly supports both UVM test suites and portable stimulus.