Taking UVM to wider user base – the open-source way

Universal Verification Methodology (UVM), as defined by the Accellera standard and soon becoming IEEE 1800.2, is getting adopted widely across ASIC design teams. UVM is also getting its flavour in SystemC so that the Electronic System Level (ESL) community can move to UVM-based approach for verifying models. Given the standard way of information flow, test sequencing (in terms of phasing for instance) being well defined, UVM is attractive to almost all electronics design teams.

However, the full-fledged UVM can be perceived as an overkill for some specific tasks, certain classes of designs. And then there are smaller teams with no formal training on advanced, software centric approach being promoted by UVM. The other group of electronic designs includes FPGA designs with majority of them using simpler, linear and procedural testbench styles with access to less powerful EDA tools. Also FPGA teams often look for template creation tools as add-ons to their EDA tools to speed up their testbench creation process. There are also cases in ASIC design flow, where-in the test inputs come as stream of structured data (such as DFT patterns) where-in teams are finding it difficult to leverage on well-defined UVM approach. Last but not the least, various university students across the globe are looking for a quick start into industry standard UVM and they do not have access to all the sophisticated training and hand-holding need to get started. Given that the future DV engineers emerge from these universities, it is imperative for the industry to get the students started UVM as early as possible.

Given the technical commonalities across all these various design verification tasks, one could imagine that some portions of UVM are still applicable to all of these tasks, while some advanced UVM features may not be so appealing to them. In this paper we present our experience in working with customers coming from many of the previously mentioned backgrounds and being able to deploy UVM to them via a convenience layer around the standard UVM named ***Go2UVM***. To make it clear ***Go2UVM*** is NOT a script to generate a set of files that users can fill-in, rather it is an OOP layer around standard UVM hiding all the glory details of UVM and providing the first-time UVM users an easy to use procedural interface to UVM. ***Go2UVM*** is open-sourced, sits on top of standard UVM and hence is 100% in-line with UVM philosophy of test creation.

What's *Go2UVM*?

In simple words, ***Go2UVM*** is an open-source, SystemVerilog package around Accellera UVM base class library. It provides a test layer around standard UVM to hide the common complexities such as:

- Phasing (reset_phase, main_phase etc.)
- Objection mechanism (A must in UVM to get even a simple stimulus through to the DUT)
- Multiple layers of components that at times smaller designs may not require
- Hierarchical component hook-ups via UVM's preferred ***uvm_component::new (string name, uvm_component parent)*** pattern
- Unclear macros that may not add value to a given task at hand

Motivation behind *Go2UVM*

Taking a step back, let's see how the mobile phone evolution has been over many decades. **Martin Cooper**, inventor of mobile phones, who was the lead engineer of the Motorola team that developed the first mobile phone recently said:

> Well, we knew that someday everybody would have a [cell] phone. Phones have gotten so complicated, so hard to use, that you wonder if this is designed for real people or for engineers.

Some customers (from the category of design teams as mentioned earlier) felt UVM is on a similar route. To summarize their feeling on UVM:

> While the "**U**" in UVM reads as "**Universal**", we wonder if it is only for software savvy, OOP fanatic engineers or can it be used by many others who are more of hardware design engineers trying to get simple verification done.

This is what got us started on the simple, yet powerful open-source package ***Go2UVM*** with the sole intention of making UVM easy to use for the first-timers.

Inside *Go2UVM*

As mentioned earlier, ***Go2UVM*** is a SystemVerilog package around standard UVM. It extends the ***uvm_test*** base class and adds 2 methods: ***reset()*** and ***main()*** – both of them are user extendable methods. The ***go2uvm_base_test*** invokes these methods inside standard UVM phasing with proper objection raising and dropping under-the-hood. It also uses ***pure virtual*** property to avoid common errors of incorrect naming of these methods. It takes care of the standard UVM requirement of component hierarchy hook-up via ***name, parent*** under the hood so that first time users do not need to bother about it. More details with the full source code can be found at: http://www.go2uvm.org/download/VW_Go2UVM_Pkg_2016.04.tar.gz

The idea is to provide the fastest way for an engineer to get started with UVM. Since it is 100% IEEE 1800 (SystemVerilog) and IEEEP1800.2 (UVM) compatible, users can easily start with ***Go2UVM*** and move to a full-fledged UVM environment, as they get mature with the technology. ***Go2UVM*** is well tested on all major simulators from popular EDA vendors.

Results/Applications

DFT patterns in simulation

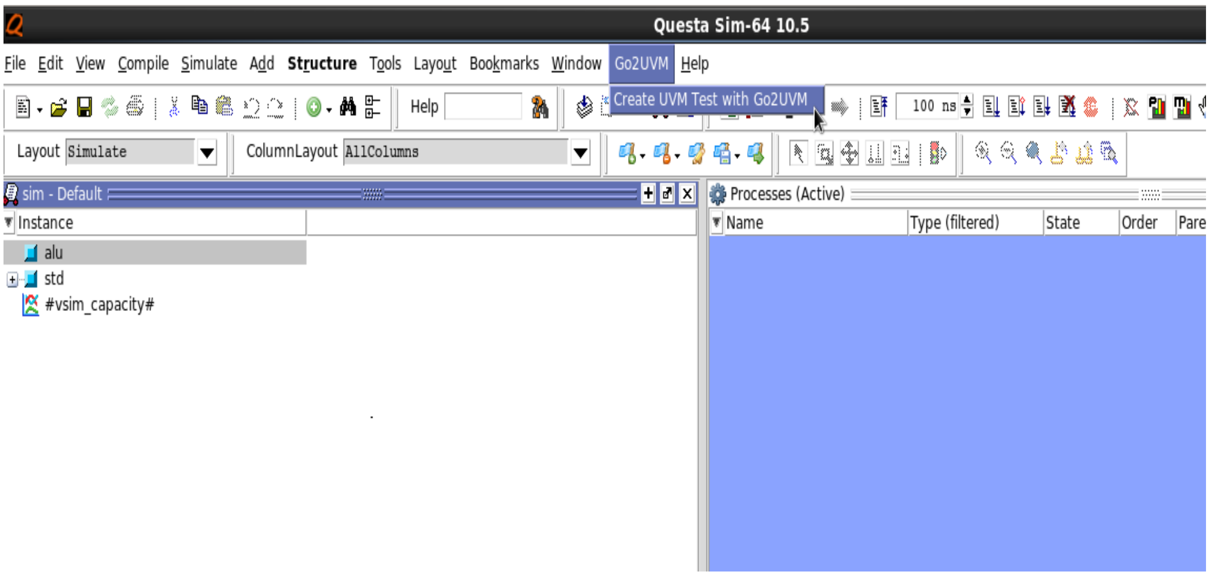
Design For Testability (DFT) is a critical step in standard ASIC design flow. As part of verification of the DFT structures engineers often run a set of scan patterns on the design annotated with scan cells and ensure nothing is broken. Given the exorbitant cost of failed silicon, it is a standard practice to run a sub-set of the scan patterns in simulation itself during the Pre-Silicon stage of the ASIC design. Except that the source of the stimulus is from DFT pattern generator, this task is quite similar to the functional verification. Since standard verification is all done with UVM, it is very lucrative for the DFT teams to run their design patterns in a UVM framework and be able to report failures, analyse coverage etc. just like how the functional verification teams do. ***Go2UVM*** has been successfully deployed in such situations to provide a convenience layer around the UVM and feeding the DFT patterns to DUT without having to go through layers of standard UVM components.

DO-254 &FPGA designs

One of our observations of FPGA design industry (and the associated DO-254 kind of designs) is that engineers doing verification of these designs have a great affinity to use Graphical User Interface (GUI) very early – as soon as the design is ready. This is quite contrary to typical ASIC design verification teams who bring up the GUI once a test (say in UVM) is ran, shows a failure and a debug database (such as DUMP file) is created. Even teams using interactive debug in ASIC design verification flow tend to use the GUI for "debug" than "test authoring".

Given this behaviour of DO-254 and FPGA teams, ***Go2UVM*** is well integrated into popular EDA tools' Graphical User Interface such as Mentor Graphics' Questa simulator. Similar integration is being done for other popular EDA vendors in the FPGA field.

The goal is to enable RTL designers to compile their designs, bring up the GUI and with a click-of-a-button generate a UVM test with necessary hooks in place. A sample integration is shown below:



Summary:

Go2UVM is a package derived from UVM library. For first-time users who may not (yet) have the prior knowledge of UVM and OOP's concepts, ***Go2UVM*** provides a simpler way to verify smaller designs. With free tools providing automation around this ***Go2UVM*** package, it is really quick for FPGA designers, students and others to get started with UVM the fastest way.

References

Accellera UVM: http://accellera.org/downloads/standards/uvm

UVM SystemC http://accellera.org/activities/working-groups/systemc-verification

Go2UVM: http://www.go2uvm.org