DVCon India 2016
Abstract submission template

TITLE OF PAPER	Partitioning for easier reuse of UVM benches – learnings from the trenches
AUTHOR 1	Name: Sumeet Gulati Organization: NXP Semiconductors, Bangalore Job Title: Senior Technical Leader Email ID: sumeet.gulati@nxp.com Mobile no: 9741277005 Alternate contact no: 080 4024 0000
AUTHOR 2	Name: Ketki Gosavi Organization: NXP Semiconductors Job Title: Trainee Email ID: *ketki.gosavi@nxp.com* Mobile no: 080 4024 0000 Alternate contact no: 080 4024 0000
AUTHOR 3	Name: Srinivasan Venkataramanan Organization: VerifWorks, a venture of CVC Pvt Ltd Job Title: Verification Technologist Email ID: srini@cvcblr.com Mobile no: 9620209225 Alternate contact no: 080-42134156
AUTHOR 4	Name: Azhar Ahammad Organization: VerifWorks, a venture of CVC Pvt Ltd Job Title: ASIC Design Verification Engineer Email ID: *azhar@cvcblr.com* Mobile no: 8123320660 Alternate contact no: 080-42134156

ABSTRACT

Mobile audio chips are critical for modern day electronic devices including cell-phones, smart phones, tablet PCs etc. Given the very fast turn-around time of these systems, a lot of design IP gets reused in these systems. While the IP reuse on the design (RTL) domain is made possible through various initiatives such as CoReUse (Ref: 1), RMM etc. As far as verification of these designs at IP level is concerned, we have a lot of legacy testbenches written in TCL and other languages from many years. New blocks are being verified using modern SystemVerilog based benches with UVM framework. As our team adopted SystemVerilog & UVM slowly into the design cycle we stumbled upon few interesting use cases that prevented reuse from IP level to SoC. While UVM has good coding guidelines and features for enabling reuse throughout this project we realized that “reuse” is much more than just base classes and

features like factory, register models in UVM. In this paper we present some of our deep learnings earned during working in the trenches deploying UVM. We also share our deep analysis on the register model randomization features, the ability to control randomness from top level etc.

Relevance of the paper

This paper would be very relevant to:

- Teams doing IP level design, verification looking to ship their verification code along with design
- Teams trying to clear the big wig marketing around SystemVerilog & UVM that makes it feel like the word UVM means reuse – with clear anecdotes we demystify that.
- Teams using UVM RAL models heavily in their systems
- Teams looking to migrate from legacy testbenches and looking for convincing reasons to do so!
- Secure designs need special tweaks with UVM RAL – we will elaborate on that in final paper

SoC architecture

Our chip is a mobile audio SoC and is part of our family of products that reuse several IPs over many years. Figure-1 below shows a higher level view of the SoC DUT (Design Under Test).

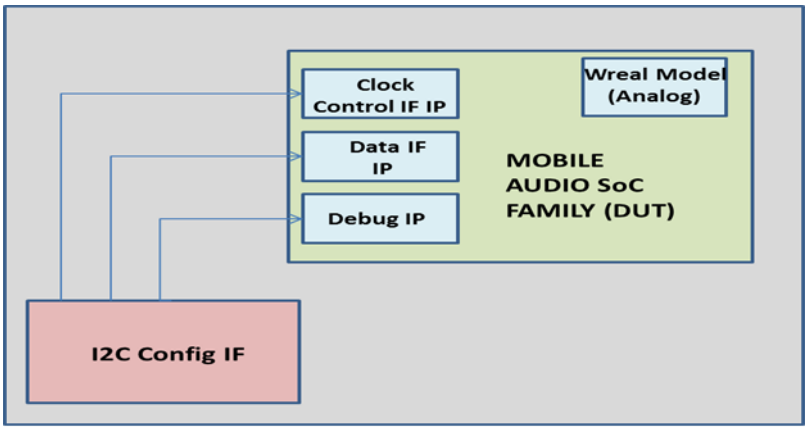

Figure 1 – SOC architecture higher level view

The actual SoC has many more interfaces and more than one I2C (Inter-Integrated Circuit) interface, but for the sake of this paper we will use the simplified view as above. Also the design is heavy Analog & Mixed Signal (AMS) in nature with WREAL (Wire-REAL) models being used to describe the analog portions. In this work we will focus on the digital part though.

Optimal Design Partitioning

One of the significant limitations of legacy environment was the inability to reuse IP level scenarios directly at SoC level. This led to duplication of all register programming sequences per IP at SoC level leading to:

- Redundant work
- Wasted debug cycles due to wrong configurations at SoC level (All SoC integrators are not familiar with every IP sequence, for instance)

So our team realized that mere use of UVM & SystemVerilog alone doesn't always enable reuse. After some brainstorming we decided to re-partition the design at IP level and include I2C control block with relevant registers at every IP level DUT as well.

Authors will share ion detail how the old partitioning hindered IP sequence reuse at SoC level and also the solution we arrived at in the final paper.

Randomization issues

The IEEE 1800-2012 LRM has well defined rules for randomization and controlling (turning ON & OFF) the same. This becomes significant in the reuse scenarios of sequences leveraging on UVM Register models.

Our team spent quality time in brainstorming various use cases that could emerge at the SoC level while reusing the IP level sequences using UVM register model. Since the register model is fixed in the structure and uses several private properties as recommended by standard OOP guidelines, end users need to be aware of means of controlling randomization from the model level and the features available to control the granularity of randomization via rand_mode (0/1). The following table captures popular use cases that our team came up with while reusing IP level UVM register sequences. (Some of the legends and use cases will require more detailed explanation, will provide the same later if the paper gets shortlisted).

API		**Model**	**Register**	**Field**
.rand_mode(0)	**.Randomize()**			
Field	Model	Random values	Random values	**No RANDOM**
Field	Register	NA	Random values	**No RANDOM**
Field	Field	NA	NA	Random values
Register	Model	Random values	**No RANDOM**	**No RANDOM**
Register	Register	NA	Random values	Random values
Register	Field	NA	NA	Random values
Model	Model	Random values	Random values	Random values
Model	Register	NA	Random values	Random values
Model	Field	NA	NA	Random values

Tweaking UVM RAL model predictors

There is some amount of hype around UVM that tends to imply "UVM knows how to verify your system" – however the reality is that UVM only provides a reusable framework and users need to write good amount of code on top to make it usable. A classical case is that of secure/protected registers in modern designs. With our design containing several such blocks the access to certain portions of the design can be controlled via dedicated protection registers. Modelling these behaviours in UVM RAL was challenging and the team used built-in UVM callbacks in the RAL model to achieve the same. We will further expand this in full paper.

Summary/Results

With proper up-front thinking of several reuse scenarios in this new chip design cycle we are able to completely reuse IP level sequences to SoC level. Though UVM had good guidelines for reuse in this project we have realized a good design partitioning and verification architecture is key to reuse. We have also stumbled upon corner case behaviours of SystemVerilog language with respect to controlling randomization from SoC level while reusing IP level sequences involving UVM Register models.

References

(1) CoReUse – NXP's framework https://www.ip-extreme.com/coreuse.shtml