Increasing Regression Efficiency with Portable Stimulus

Niyaz. K. Zubair[1]
QUALCOMM India Private Limited,
Outer Ring Road, Doddanakundi, Bangalore
nzubair@qti.qualcomm.com
(Contact: +91 - 9902029185)

Subba Kota Rao Sajja[2]
QUALCOMM India Private Limited,
Outer Ring Road, Doddanakundi, Bangalore
kotas@qti.qualcomm.com
(Contact: +91 - 9448467129)

***Abstract*: Functional coverage closure using random testing of modern multimedia designs is a tough task. Closure of last 1% to 2% of the coverage bins require manual effort, which is the most difficult part of the entire process. This paper explains the experience and results of deploying a portable stimulus-based random verification methodology to boost regression efficiency for an existing System Verilog testbench, by reducing manual interventions for coverage closure and enable functional coverage closure in a short and predictable time.**

I. INTRODUCTION

Video codec IP is an integral part of today's mobile phones, tablets etc. Modern Video cores support multiple codecs (MP4, H264, H265, VP8/9 etc.) and up to 8K resolutions, with various profiles and features. The video core has multiple sub blocks supporting various codec tools (estimation, compensation, filtering, (de)coding, data management etc. [1], which makes the verification a complex task. Random verification is used at the block and sub block level.

Due to the huge number of codec tools, functionalities and scenarios to be supported across those multiple codecs, the functional coverage closure using random testing is a challenging task. It takes a huge set of regressions to reach the target and the last 5% usually necessitate manual interventions to generate directed scenarios. Fundamentally the number of cover bins and cross cover points that can be hit, which can be added in coverage plan, are also limited by this effort involved and time consumed. Moreover, there is no clear estimate possible on the number of random seeds or runs needed to achieve the 100% coverage target. There is a high possibility for various unalignments between stimuli (valid values, maximas, dependencies between random variable etc.) and corresponding coverage definitions, which usually get reviewed and corrected only during the last stages of coverage closure activity. This also affects the efficiency of overall coverage closure activity using regression. This paper covers the experience and benefits of adopting a portable stimulus-based methodology, which solved all above issues associated with functional coverage closure in a random verification environment.

The Accellera Portable Test and Stimulus Standard (PSS) language defines a specification to create a single representation of stimulus and test scenarios usable by a variety of users across many levels of integration under different configurations. This representation facilitates the generation of diverse implementations of a scenario that run on a variety of execution platforms [2]. It enables a formal model of both test intent (stimulus) and coverage goals to be captured. In addition to enabling the same model of test intent to be reused across a range of verification environments, portable stimulus methodology enables the process of efficient stimulus generation to be approached in a different way. Instead of generating stimulus open-loop, the declarative portable stimulus models of test intent and coverage goals can be used to generate efficient stimulus that is focused on the achieving the coverage goals, as captured by the coverage model.

The formal and declarative nature of a portable stimulus description also supports analysis to detect mismatches between test-intent and coverage models. This capability was used to identify mistakes in both the constraints and the coverage models, and to ensure that the reachable size of coverage goals was as expected.

This paper will describe the process and results of deploying portable stimulus on a live project to increase regression efficiency. It will describe how high-value integration points for portable-stimulus generation are identified in new and existing testbench environments, what elements of the existing testbench can be leveraged to create the portable stimulus model and integration, and what elements must be created specifically to support the use of portable stimulus. It will describe the methodology used to identify and address gaps between the test intent model and the coverage model and will describe how the test intent model is mapped to coverage models that are

not identical to the test intent model. Finally, it will describe the effort involved, the overall regression efficiency and overall verification process benefits realized by applying portable stimulus in this manner.

II. PROBLEM STATEMENT

The video bitstream is spread across several units like - frame, tiles, slices, large coding units (LCU), macro block (MB) or coding units (CU), prediction units (PU), Transform units (TU) etc. There are several parameters which are inter dependent across these units and the behavior of those parameters also depend on the resolution of frames, types of inter and intra predictions and directions supported, global motion estimator used, search range area and estimation search algorithm used, number of parallel engines in the architecture (scalability), fps and performance requirements for the use cases etc [1]. Video IP has various functionalities like bitstream parsing, syntax element decoding/encoding, transform, prediction, motion compensation, integer/fractional search, filtering and memory management.

In Video core Memory buffer takes care about various internal data structures and fetch requirements of surrounding blocks, data manipulation necessities etc. Memory buffer (DUT) maintains common data memory to manage shared data and for custom data structure requirements by other processing blocks. Recently, the architecture updates in the DUT increased complexity and added more use cases which ended up in new multiple scenarios and corner cases for the random verification environment. This ended up in a complex functional coverage model around various supported tile combinations, frame resolutions, global motion, types of frames, number of hardware pipes etc. Out of the total cover point bins, 90% are cross bins. Multiple stimuli classes which are randomized at different places and cross cover points defined across those classes added to the complexity. Practically coverage closure process involved fine tuning the stimulus manually to hit some corner scenarios, though we wanted to target to specific scenarios of interest. So historically coverage closure took a huge set of random regressions to reach the 100% target on the DUT and the last 5% usually necessitated manual interventions to generate directed scenarios. The functional coverage closure for the last 2% bins took 4 to 7 weeks of rigorous direct scenario creation effort on the last architecture. So, it anticipated a much more effort this time, if the problem is approached in the usual manner.

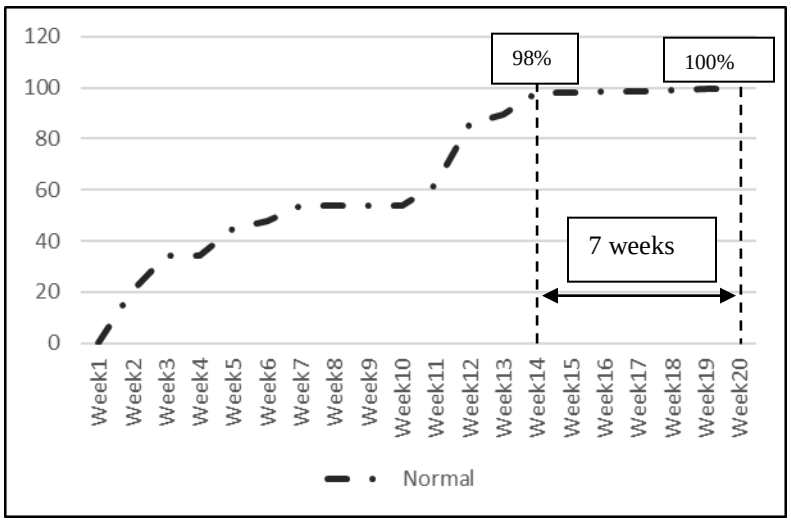

Figure 1. It took 7 weeks for closing last 2% functional coverage

Fundamentally the number of cover bins and cross cover points that can be added to the coverage model are challenged by this effort involved and time consumed. Constraint solvers lack the knowledge of how to smartly generate the stimuli when we wanted to hit cross cover points with more parameters and inefficiently coded constraints restrict generation of real use case scenarios. Memory buffer block (DUT) has crosses with even eight parameters. Furthermore, there were no clear estimate possible on the number of random seeds or runs needed to achieve the 100% functional coverage target with these complexities.

The numerous unalignments between stimuli (valid values, maximas, dependencies between random variable etc.) and cover point definitions are reviewed and corrected only during the last stages of coverage closure activity. But this also affected efficiency of overall coverage closure cycle. This paper covers the experience and benefits of

adopting a Portable Test and Stimulus Standard (PSS) based coverage closure methodology, which solved all above issues associated with functional coverage closure in the random verification environment.

III. EXISTING APPROACH

The verification environment for memory buffer had two stimulus classes (frame_cfg and ref_cfg) which were randomized at different places in the sequence. But since there is no direct control on the variables between these classes, there is no control on the stimulus generation and the cross-coverage bins defined across variables from these classes were hitting very slowly.

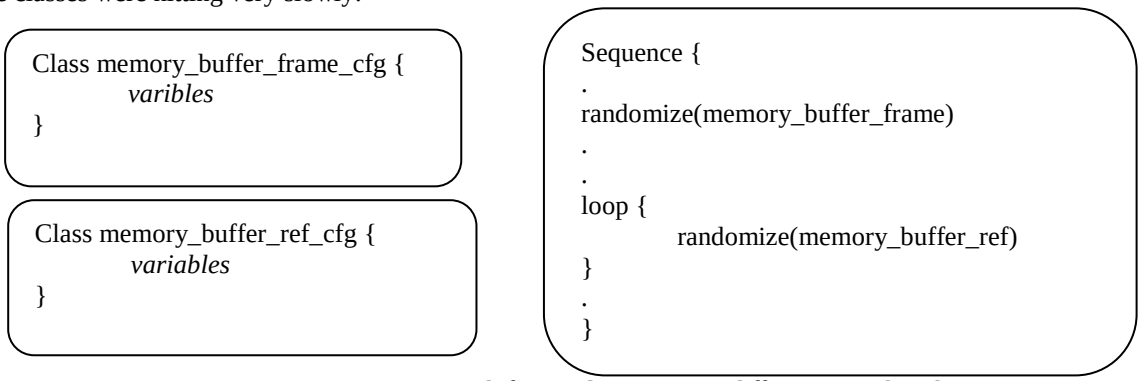

Figure 2. Existing approach for randomizing two different stimulus classes

IV. MOTIVATION

Video codec syntax parameters are very highly interlinked and inter dependent for a bitstream. This dependency needs to be exploited intelligently so that thousands of tests defined in regression (which corresponds to a bunch of scenarios and that mapped to several cross-cover points) can be generated efficiently without repetitions. The removal of repetitive stimulus will help to improve regression efficiency.

IV. SOLUTION

The new approach of PSS based coverage closure methodology deals the whole problem in a different way. The methodology empowers generation of stimuli with coverage targets as the goal and generate non-repeating stimuli scenarios considering all cross-coverage dependencies in the coverage model. The major steps are as below,

A. Container class is introduced to instantiate frame class (for frame level configuration of parameters) and reference class (child class) in the random environment, which require minor changes in the sequence (now need to randomize only one class) but has the advantage of controllability of stimuli generation and hitting of cross bins defined across those classes.

```
class memory_buffer_container extends uvm_sequence_item;
        rand memory_buffer_frame_cfg    u_ memory_buffer_frame_cfg;
        rand memory_buffer_ref_cfg      u_ memory_buffer_ref_cfg;

        constraint c1 {
                u_ memory_buffer_ref_cfg.standard = = u_ memory_buffer_frame_cfg.u_c1.standard;
                u_ memory_buffer_ref_cfg.frame_w = = u_ memory_buffer_frame_cfg.frame_w;
                u_ memory_buffer_ref_cfg.frame_h = = u_ memory_buffer_frame_cfg.frame_h;
                u_ memory_buffer_ref_cfg.data_pitch = = u_ memory_buffer_frame_cfg.u_c2.dp;
        }
endclass
```

Figure 3. Container class: memory_buffer_frame_cfg and memory_buffer_ref_cfg class dependencies are captured

B. A mapping is introduced between stimuli random variables in the container class and coverage class variables to enable this. It creates a System Verilog class with a special constraint randomization method, which takes care about generation of stimuli with coverage target goals.

Type	Stimulus	Coverage	Mapping
Simple	var_1 inside {0,1}	var_1_s_var {bins val_0 = {0};bins val_1 =1}};	var_1_s_var,var_1

Using expression	var_2 inside {96,192}	var_2_s_var {bins val_96 ={6};bins val_192={12}};	Var_2_s_var,EXPR:(var_2)/16
One stimulus mapped to multiple bins	var_3 inside {0,1}	var_3_s_var {bins val_0 = {0};bins val_1={1}}; var_4_s_var {bins val_0 = {0};bins val_1={1}};	var_3_s_var,var_3 var_4_s_var,var_3

Table 1. Three examples of coverage to stimuli mapping

C. The generated System Verilog class is analyzed further and report valid scenarios as stimuli possibilities. The stimuli possibility count characterizes the valid stimuli space for the constraints defined around the stimuli, their dependencies and coverage defined. This step aid constraint efficiency improvement and helped to identify,
   1. The mismatches between coverage variables and random variables in definition and mapping
   2. Detect over constraints which affected masking of any scenario crosses
   3. Identify loose constraints which created huge number of stimuli possibilities unnecessarily

The flowchart shown below depicts the over methodology followed,

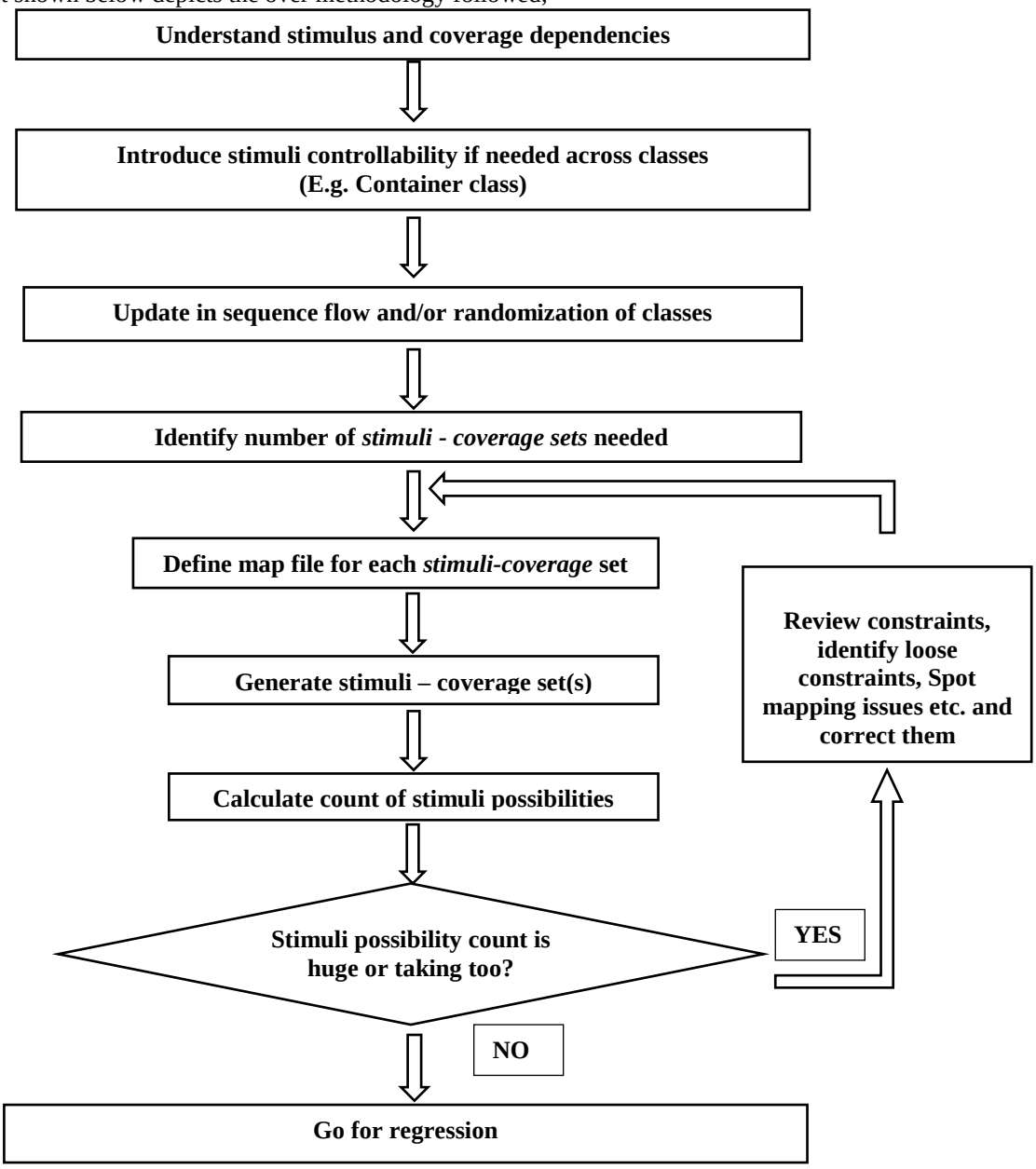

Figure 2. Flow Chart of the PSS based coverage closure flow

The balancing of total coverage bin count and stimuli possibility count facilitated to create the most efficient stimuli – constraint set for the targeted coverage. The methodology provides a simulation distribution manager which manages efficient stimuli generation without repetitions across various seeds and/or tests and ensure that each test is targeting a new required scenario defined in the coverage plan.

V. RESULTS

The PSS based coverage closure approach helped to improve the overall random regression efficiency by 75%. The approach ensured that each randomly generated tests/scenario are non-repetitive. The scenario generation was quick and non-repetitive, could find several design and system model issues in short span of time (five weeks in case of Memory buffer). It was almost impossible to confidently cover all scenarios around the new architecture in this time with a good confidence.

The table below summarizes the comparison of coverage closure using the normal approach and the new approach. The number of tests to be written and seeds are comparatively less which directly improve the productivity of the regression task.

Items of interest	Normal approach	Using PSS based approach
Total number of tests	25	1*
Cover points (approximate)	19245	19245
Total seeds (indicates run time)	75,000	20,000
Time taken for coverage closure	16 weeks	< 4 weeks (including initial setup time)

Table 2. Comparison of the approaches (One test is enough with the new approach as the stimuli is coverage target driven and methodology takes care about scenario generation)*

In summary, the new PSS based approach increased efficient and productivity, it produced good results and has following advantages,

1. Generates distinct stimuli and scenarios from day one of verification. The steepness in coverage progress in the below graph shows that,

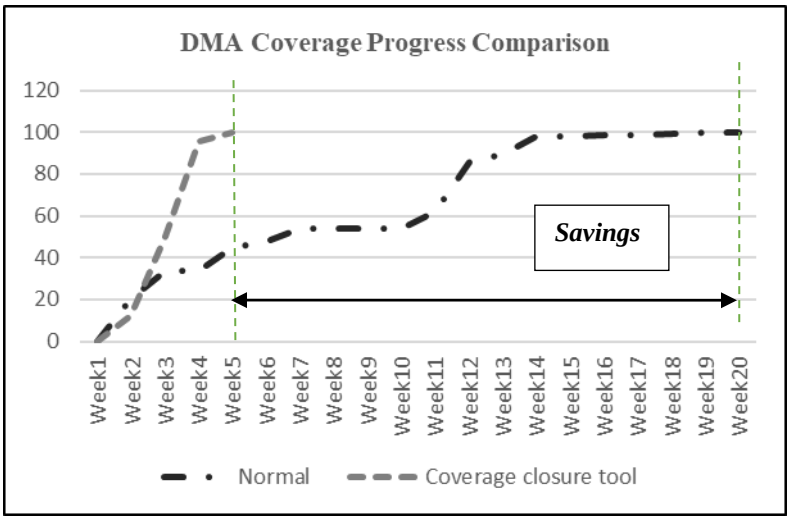

Figure 4. Coverage Progress Comparison

2. Over all time saving of 75% (4 weeks vs 16 weeks in case of Memory buffer block) on DV efforts and LSF machine usage.
3. Coverage closure time was predictable even with complex functional coverage targets.
4. Improvements in simulation run time after optimization of constraints

5. Quality of DV is improved – better and efficient of stimuli and constraints, reducing gap in stimuli and coverage model.
6. The methodology is very helpful when we want to close coverage for incremental projects with the same random stimuli set from a previous project, but without spending much time and in a predictable time. All the setup can be reused and even the minor coverage updates can be added quickly and close the coverage quickly.

VI. CONCLUSION

Overall 75% of savings on regression (testcase writing effort, scenario generation effort, debug efforts, LSF time etc.) was achieved with the adoption of the new PSS based coverage closure methodology. It gave very competitive results with promising reduction in run time and reduced user involvement in functional coverage closure. The PSS based coverage closure methodology is very promising one and can manage complex stimuli and coverage scenarios and reusable easily for future projects.

The methodology is powerful to manage multiple stimuli-coverage strategy sets in parallel. Many times, the stimuli are spread across several classes and there are cross cover points defined across those classes. Multiple stimuli-coverage System Verilog class intends can be built and instantiated in the environment and it has given good control over stimuli generation. The impact in this case is going to be much higher than the Memory buffer block. So, this methodology can be potentially used for various other blocks too.

VII. ACKNOWLEDGMENT

Thanks for all the help and support from my team for experimenting and proving this.

VIII. REFERENCES

[1] https://www.vcodex.com/an-overview-of-h264-advanced-video-coding/
[2] https://www.accellera.org/activities/working-groups/portable-stimulus