Accelerated Coverage Closure by Utilizing Local Structure in the RTL Code

Rhys Buggy, Engineer, QT Technologies Ireland Limited, Cork, Ireland (*rbuggy@qti.qualcomm.com*)*

Gokce Sarar, Engineer, Qualcomm Technologies, Inc., San Diego, USA (*gsarar@qti.qualcomm.com*)*

Guillaume Shippee, Engineer, Qualcomm Technologies, Inc., San Diego, USA (*shippee@qti.qualcomm.com*)*

Han Nuon, Engineer, Senior Staff, Qualcomm Technologies, Inc., San Diego, USA (*hnuon@qti.qualcomm.com*)

Vishal Karna, Engineer, Principal/Mgr, Qualcomm Technologies, Inc., San Diego, USA (*vkarna@qti.qualcomm.com*)

Tushit Jain, Sr Dir, Technology, Qualcomm Technologies, Inc., San Diego, USA (*tushitj@qti.qualcomm.com*)*

***Abstract*— Coverage closure constitutes a large portion of hardware design verification process. In order to cover the hard-to-hit coverpoints, the DV engineer has to rely on running randomized tests for long durations. In this paper we describe a technique to accelerate randomized testing by parametrizing the test constraints and finding the parameters that maximize coverage. We take advantage of the signal subgraph of the target signals in order to find the constraint parameters, which are more likely to make the target coverpoint toggle.**

***Keywords*— Hardware Design Verification; Genetic Algorithm**

I. INTRODUCTION

Before committing the RTL code of a hardware system to manufacturing, it requires to be checked against the specs. Design Verification (DV) currently takes ~6 months for industry level chips, requiring large time, effort and compute spent on running tests. One aspect of DV is to verify if the RTL design has been tested thoroughly, i.e. the coverage. The problem lies in covering hard to hit coverpoints, which most of the test vectors do not hit. The current methodology is to run randomized tests to hit these hard to hit coverpoints and they constitute the larger part of the design verification time. In this paper, our contribution is in accelerating the code coverage for design verification. We perform subgraph extraction of the RTL graph and utilize the subgraph signals to drive the target coverpoints.

II. RELATED WORK

Many approaches have been applied in the past to reduce coverage closure time. These approaches involve genetic algorithms [1], machine learning methods [2], non-simulation-based approaches using probabilistic inference [3], and formal techniques. However, these approaches either assume that the design is small, on the order of several thousand gates, or they assume that the coverage has already been achieved and provide solutions to achieve such coverage again more efficiently. We aim to find a technique that overcome both these limitations by addressing large designs (on the order of hundred thousand gates), and coverage points which have no history of being hit before.

III. APPLICATION

A. *Parametrizing Test Constraints*

To achieve coverage closure, directed tests are used frequently, where the DV engineer constrains certain test vector parameters to drive the random test vector generation to hit target coverpoints. When the target coverpoint is very hard to hit and is the output of a complex logic block, writing a directed test is hard even with a knowledge of the design. That's why the common approach for the hard to hit coverpoints is running the random tests for a

* These first authors contributed equally.

very long time. Our aim is finding the best combination of the test vector constraints, which defines the distribution from which the test vectors are sampled. To achieve this, we parametrize these constraints and measure their effectiveness in achieving coverage from statistics of the subgraph driving the coverpoint.

B. *Subgraph Extraction*

The challenge in achieving coverage lies in the lack of signal activity of the target coverpoints for most input test vectors. Since most of the test vectors don't activate them, ranking parameter settings with respect to target activity proves to be difficult. Instead of treating the signals as black box components, we utilized the RTL code, which describes the relationship of other signals to the coverpoint. The idea is when the target doesn't toggle, the neighboring signal activity can inform us about the goodness of the parameter settings, which we call the fitness score.

C. *Subgraph Statistics*

A list of signal statistics which are descriptive of signal behavior, is collected per signal in the target's subgraph within a pre-defined time-window. These statistics indicate which constraint parameters are more likely to make the target coverpoint toggle and are used to guide a genetic algorithm search.

D. *Genetic Algorithm (GA)*

We applied genetic algorithm as in [1] to generate the constraint parameters based on the subgraph statistics mentioned in section C. It exploits parameters settings that are likely to improve coverage, by combining them into new candidates, and explores the search space by randomly transforming the parameters to cover unseen parts of the solution space.

IV. Approach

Our approach focuses on IPs which use the standard UVM framework for the creation of its testbench. Specifically, when we refer to parameter settings, we are referring the transaction level parameters, which are assigned prior to being sent to the driver. Examples include setting the address range, or read/write command and the associated data, for an AHB transaction. The approach can be extended to test level parameters as well, however, this extension is not discussed in this paper. For each transaction parameter, the allowed ranges of values are identified, and our algorithm picks a subset of those parameters in (MIN, MAX) format for continuous parameters, and {A, B, C, …} for enumerated parameters where A, B, and C are a subset of the possible values for that parameter. For example, an address range of (0x2, 0x4) out of (0x0, 0xF) , and transaction type of {READ} out of {READ, WRITE} may be selected by our algorithm. Such a group of parameters settings is defined as a single parameterization to test.

The chosen parameterization is assessed by dumping the toggle activity of the design in the form of an FSDB and parsing to FSDB to extract the inputs necessary for our score functions. The score functions are evaluated over the neighborhood of the surrounding the target to provide a proxy for how close this parameterization got to hitting the desired coverage. Several score functions have been defined for our approach, each of which target a different quality we deem desirable for causing coverage closure. As such, higher scores are assigned to those parameterizations which led to coverage-inducing neighborhood behavior. The neighborhood is defined as some portion of the target signal's cone of influence, and is limited either by some net register count, or some net average toggle activity over all involved signals.

Once a parameterization has been determined, it is evaluated over so many time cycles, which is given by some hyperparameter, by dumping and analyzing the results of the FSDB, before trying out another parameterization. After so many parameterizations have been tried, the scores of each are assessed and passed into the genetic algorithm. A new set of parameterizations will be returned to try out, and the process is repeated until coverage is reached.

To be able to control the parameter settings during a simulation based on feedback from the genetic algorithm, we use the open-source PYVPI package, and add hooks directly to the System Verilog UVM code to interface with python. As a result, the process does not require any work on part of the DV engineer once the initial hooks have been added. Coverage statistics and parameter scores are kept and updated in a database, for easy querying and data management when multiple tests are being run in parallel. This integrated flow is summarized in Figure 1.

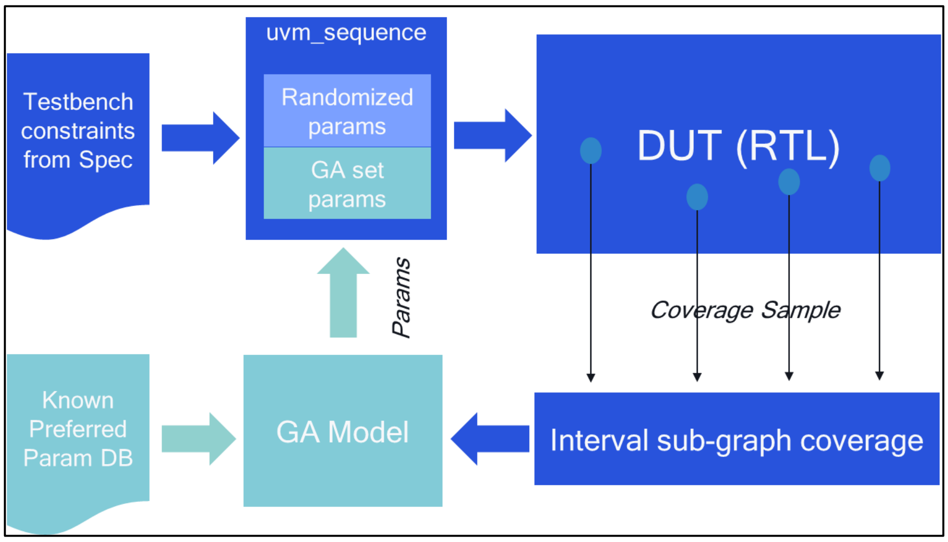

Figure 1. Integrated flow for GA directed stimulus. The Test stimulus is directed by means of GA with scoring functions derived from DUT behaviour.

V. Results

We assessed the performance of our approach for a design with 400,000 gates and 150,000 registers, and for a design with 50,000 gates and 20,000 registers. DV was performed with a UVM test bench, and we integrated our algorithm with the UVM test bench.

For the first IP, 10 hard-to-hit target signals, and for the second IP, 19 hard-to-hit signals were selected by running an initial baseline regression and identifying those signals which were most rarely hit. These signals ranged from being hit on the order of 1 in 100000 tests to 1 in 100 tests. For evaluation purposes, signals which were never hit were not selected for comparison due to the inability to determine whether such was the case because of poorly written code or not. Alternatively, a list of target signals could be provided by the DV engineer, if an initial regression is not desired to be run for baselining. Once the target signals were identified, both the non-constrained random test and the sub-graph statistics-based GA were run. To minimize noise, we applied a design reset after a certain number of transaction items had been created and sent into the design, and applied the same parameterization for 2 consecutive reset windows. The larger design consisted of 50 transactions per window, and the smaller design consisted of 8. The number of transactions is a hyperparameter which can be adjusted depending on the perceived depth of the design state machine, and the minimum number of transactions required to activate all such states.

With random test, all 10 target signals in the first IP are hit after *89,100* windows as can be seen in Figure 2 whereas our GA algorithm achieves full coverage in *2,020* windows, a 44x speed up. With the second IP these numbers are *1500* windows, compared to *60*, as seen in Figure 3, as 25x speedup. Our algorithm has not been compared to directed or semi-directed tests for the same design, however, many of the identified signals did not have such tests written for them. Additionally, overhead costs for running our algorithm have not been added, as they are minimal in comparison to the total runtime of all the tests. Targets were updated for coverage in a round robin approach.

VI. Limitations and Further work

In addition to this flow and utilizing the score function, alternative methods to search the parameter space can be developed. Using the score functions previously defined, a vanilla decision tree model was trained to predict the fitness score for a given input parameterization. This approach allows for a proxy score to be generated for parameterizations not yet simulated and expand the number of score-defined parameterizations. As future work we will evaluate these predicted scores, by replacing the GA model with these top-ranked predicted scored parameterizations.

Although results are promising, several limitations exist to our approach. Our methodology relies on the assumption that one can clearly distinguish the effect of one parameter setting from another. Because tests often cycle through several different parameter settings, the effect or one parameter setting may impact the effect of another, introducing noise. One can get around this be only running one parameter setting per test. However, this may not be feasible depending on the number of machines available, and for designs with deep state machines, perhaps a transition between parameter setting is required for coverage closure. Our approach is limited to coverage points that do not depend on this sort of behavior. An extension may be to include a score for parameter setting transitions in our genetic algorithm to capture these.

Further, defining the parameter space to explore has its difficulties. It is rare to find a design testbench which does not place constraints on the parameter settings, and sometimes those constraint solvers are not accessible to the DV engineer. Consequently, defining the set of valid parameter spaces can be difficult, and even if we manage to define a valid parameter space to search, often this space can be huge. A DV engineer can partially mitigate this by providing feedback on which parameters to explore for various coverage goals, or a hyperparameter which sets the range of parameters to explore in any given test can be modified, but even so, it may not be enough to overcome some large parameter spaces.

Another limitation of our approach is that it currently only targets toggle coverage. The reason for this is it is easier to collect aggregate statistics for this type of coverage than other types, and, unlike functional coverage, toggle coverage is defined over signals in the design RTL. Thus, extracting the subgraph to characterize neighborhood activity is feasible. Our approach can potentially be extended to other forms of code coverage, but further score functions will need to be defined, as well as other methods for data collection.

Lastly, identifying a scheme with which to identify new target signals has not been addressed yet. For the time being, we ignored signals which have never been toggled before, and only focused on those which had been toggled a few times over the course of regression. However, as these signals are hit more and more frequently through our algorithm, new targets can be identified. Determining the frequency with which to perform this update is an open item.

VII. Conclusions

Using sub-graph signal statistics to guide the search for constraint parameters with a genetic algorithm can achieve the same coverage as randomized testing with 44 times acceleration for a 10 hard-to-hit coverpoints in a 400,000 gate design. Our contribution lies in the fact that we don't assume the target signals were hit in previous simulations but utilize the subgraph signals statistics as an indicator of parameters that can toggle the target. We are currently applying this technique to a broader set of designs with hard-to-hit coverpoints.

References

[1] P. Faye, E. Cerny, "Improved Design Verification by Random Simultion Guided by Genetic Algorithms", In Book System-on-Chip Methodoligies & Design Languages 81-94, Springer, Boston MA, 2001

[2] F. Wang, H. Zhu, P. Popli, Y. Xiao, P. Bodgan, S. Nazarian "Accelerating Coverage Directed Test Generation for Functional Verification: A Neural Network-based Framework" 2018 Greate Lakes Symposium on VLSI (GLSVLSI '18), 2018.

[3] J. Fernandes, M. Santos, A. Oliveira, J. Teixeira, "A Probablisitc Method for the Computation of Testability of RTL Constructs" in Book 1. 176-181. 10.1109 Jan 2004

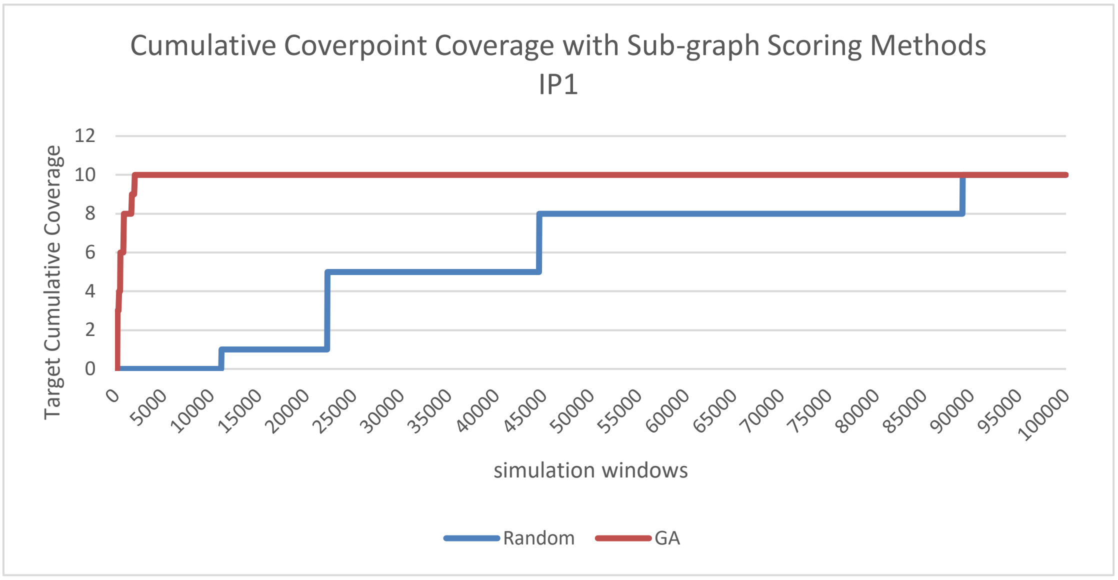

Figure 2. Comparison of random test vs. GA IP1

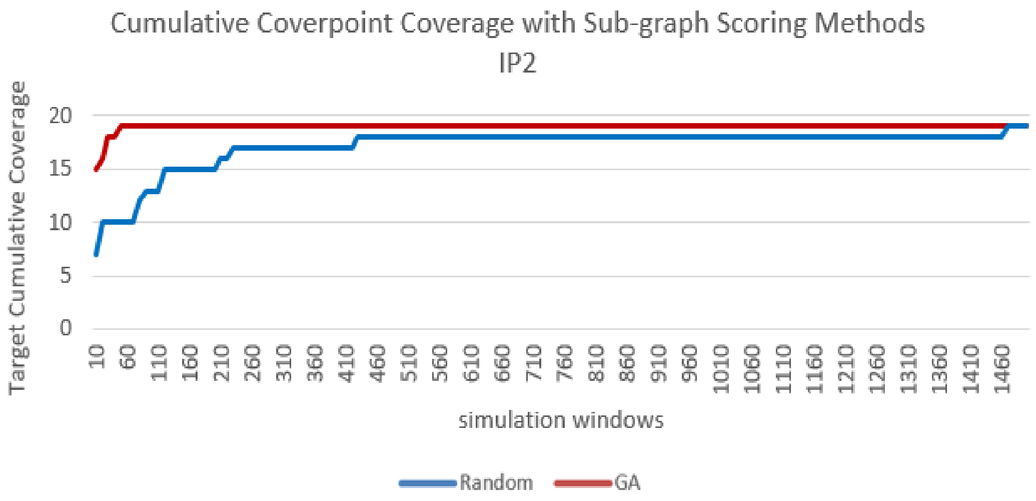

Figure 3. Comparison of random test vs. GA IP2