Proceedings of the Second International Conference on Inventive Research in Computing Applications (ICIRCA-2020)
IEEE Xplore Part Number: CFP20N67-ART; ISBN: 978-1-7281-5374-2

Optimization of Regressions in L2 Cache Verification

Basaweshwari
Department of ECE,
RV college of Engineering,
Bengaluru-560059, India
Email: basaveshwaribiradar@gmail.com

Dr.H.V.Ravish Aradhya
Professor & Associate Dean
Department of ECE,
RV college of Engineering,
Bengaluru-560059, India
Email: ravisharadhya@rvce.edu.in

Robert Chan
Engineer, Principal/Mgr
Qualcomm Technologies, Inc.
San Diego, USA
Email: rchan@qti.qualcomm.com

Jerry Dai
Engineer, Senior Staff
Qualcomm Technologies, Inc.
San Diego, USA
Email: jianzhen@qti.qualcomm.com

Pawan Yenamandra
Lead Engineer, Sr
Qualcomm Technologies, Inc.
Bengaluru, India
Email: ysastry@qti.qualcomm.com

***Abstract* - In a repetitive and progressive development of configurable IP core environment, Regression testing in design verification plays a vital role. As size of the design continues to grow accordingly increases the complexity of verification and regression time. However, it is a very well known fact that maintaining regression quality and reducing regression time is very difficult. In this paper, a novel technique is proposed for configurable L2 cache of IP cores, in which moving all plusargs from command line of simulation launching regression scripts into the testbench as an option groups and then randomizing the option groups during simulation for each test case. From the proposed regression optimization technique overall 42.33% machine cycles, 42.01% of CPU time is saved with coverage more or less similar as traditional one for multiple rounds of runs.**

Keywords — L2 cache; plusargs; coverage; machine cycle; CPU time.

I. INTRODUCTION

In the field of design verification, the size of the Regression test suite may range from thousands of test cases to several ten thousands of test cases, The number of test cases are mainly depends on the complexity and features of the design under test. Therefore Regression run time and quality is relevant when design is under the incremental changes with the limitations of time and computational resources.

Even for a small change in design, which leads to the running of all functional, coverage and performance regressions with several thousands of regression test cases for many days to ensure the reliability of the design .But these regression runs consumes lots of resource and time. This directly impacts on the cost. Saving time, resources and money plays a very important role in the competitive world of semiconductor industry.

Constrained random verification (CRV) is an effective method for achieving coverage target faster. CRV plays an important role when it comes to hitting a complete corner case bug, which is very hard to hit with directed test cases. The whole regression testing approach consists of the regression test suite of test cases which are run repeatedly. Due to the test stimuli random nature coverage goal is achieved by repeated runs of regressions for many days. To hit a particular corner case bug may requires several days of regression runs.

This several days of regression runs consumes lot of computational resources and machine cycles. Also, with increase in regression test run count for hitting corner cases, the previously covered features are tested repeatedly and may not contribute to the coverage at all. In this scenario the constrained random verification regression test suites needs to be optimized. So, that the optimized regressions are able to save machine cycles and computational resources.

In the proposed optimization technique instead of running all the regressions for several days, single regression will be run by adding the flavors of all the features in a calculated approach using strategy of weight assignments to features by smart randomization logic yielding better results in terms of machine cycle and CPU time with maintaining same quality as of traditional regression techniques.

II. CONVENTIONAL WAYS OF REGRESSION TESTING

Design code is a process which contains error corrections, enhancements, optimization and removal of existing features. This modification will make the system to work wrongly. So, Regression Testing becomes important here. Regression Testing is done with the following conventional techniques shown in Fig. 1.

Basically there are three conventional regression techniques.

A. Retest All

Retest All technique includes all the test cases, Which checks all test cases contained in the regressions on the current code to check the integrity of it. As it needs to re run all the test cases which becomes costly .But retest all ensures there is no errors due to modifications.

B. Regression test selection

Regression test selection technique is not like as retest all. In retest all the all regression test cases are run. But here in regression test selection only a selected part of the regression test cases are executed if the cost of the selected test cases is less than the retest all.

C. Prioritization of Test Cases

Prioritization of tests makes the number of faults less than in regression test selection. Also it makes sure that the high priority test cases are run before the lower priority to increase the fault detection rate. There are two types in test case prioritization which are as follows.

a. *General prioritization:* Prioritizing test cases which are useful in all the versions.

b. *Prioritization based on Version specific* : Prioritizing test cases for particular version of design.

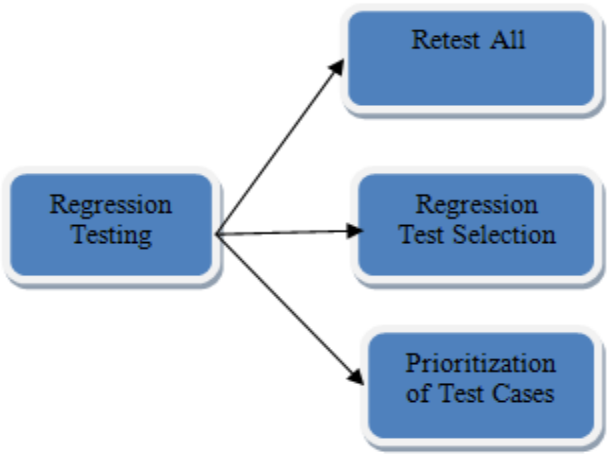

Fig 1. Conventional ways of regression techniques

In the upcoming part of the paper will discuss in detail about the Traditional way of regressions in configurable IP core L2 cache and Optimized regression technique for the same.

III. TRADITIONAL REGRESSIONS IN CONFIGURABLE IP CORE L2 CACHE

In configurable IP core L2 cache even for a small change in code or configuration .Demands running of complete set of regressions for many days to make sure that the recent change in the code or configuration has not affected the design adversely.

A. Traditional Regressions Types in Configurable IP Core L2 Cache

The traditional regressions are broadly classified into three types in our configurable IP core L2 cache design verification.

1. Functional Regressions
2. Performance Regressions
3. Coverage Regressions

Functional regressions consist of regression scripts to check functionality of the design. Performance regressions consist of regression scripts to check performance of different features of the design. Similarly coverage regressions consist of coverage scripts to cover different features of the design.

These scripts are cron scripts which consist of some design feature specific plusargs to be passed from the command line of the script. Cron scripts are the regression simulation enabling scripts .Which will be scheduled in cron to run a particular regression periodically for a particular day, interval or time.

B. Computational complexity of Traditional Regressions in configurable IP

To run these many regressions first requires lots of lsf slots of machine, to meet this requirement many engineers are required to schedule and run these jobs on their machine which leads to consumption of both computational and human resource. Secondly , As there are too many regressions needs to be run which is not possible to complete one round of these regressions in a single day. So, to cover all regressions particular regression will be scheduled on a fixed time of particular day of a week in cron.

As these regression runs are constrained random regression tests they needs to be run repeatedly to achieve coverage goal and hit complete corner cases. This will leads to consumption of too many resources, machine cycles and CPU time. To reduce resource consumption and save machine cycle, CPU time by not much effecting coverage quality going for novel regression optimization technique.

IV. PROPOSED REGRESSION OPTIMIZATION TECHNIQUE

Here to optimize the regressions, the plusargs from the command line of regression simulation enabling cron scripts are combined into option groups. The created option groups are moved into the testbench. Now, inside testbench option groups are randomized during the simulation. The proposed technique is illustrated as follows. To illustrate the technique consider the performance regressions.

In traditional regressions, performance regressions have as many performance scripts as many features present in the design. To explain this proposed technique and simplicity purpose let us assume here only four features for the design. For four features of design four performance regression scripts are there. Each script will have some plusargs in the command line of the script. Now, combine all plusargs from command

line into the script and move them into the testbench. As four scripts are there for four features, just for simplicity purpose name the feature and option group name as A,B,C and D.

Now, consider these four performance regressions scheduling in traditional way. A regression is scheduled at a particular fixed time and day of a week in cron. Similarly, for B, C and D by different engineers or if one engineer is running then he should make sure each regression will have at least two hour gap in between them. To run four regressions four lsf slots and minimum eight hours required to get all regression run results. In the proposed technique moving all the four regressions option groups into the testbench and randomize these option groups during simulation. For example consider the following pseudo code.

```
if (performance regression =1)
begin
randcase
Weight A: A;
Weight B: B;
Weight C: C;
Weight D: D;
endcase
end
```

So, Created switches to enable the particular regressions randomization. Here three main regressions are there functional, performance and coverage regressions. So, three randomization enabling switches required to create and pass them from the command line of cron regression script of particular regression.

In the above pseudo code the performance regression is enabled. And based on weight passed particular option group is randomized during simulation. Assume equal weight is passed to each four of the option groups. Then at the end of simulation of performance regression 25% flavor of each feature regressions will be there. This means in one regression run can have all feature regression flavor. If a particular feature regression flavor is required in the result just increase the weight of that particular feature option group. That is to schedule performance regression only one engineer and one lsf slot required. Result of this performance regression will be obtained much faster than traditional one.

To achieve coverage goal the regressions needs to be run multiple times. This is explained by following algorithm of Fig 2.

Similarly, Functional and Coverage regressions are optimized. After optimization of all major three regressions the lsf slots and resource required is less than the traditional performance regressions. Also improves qualification efficiency. During early face of project whenever changes in design takes place only full regression from functional regressions need to be run for qualification.

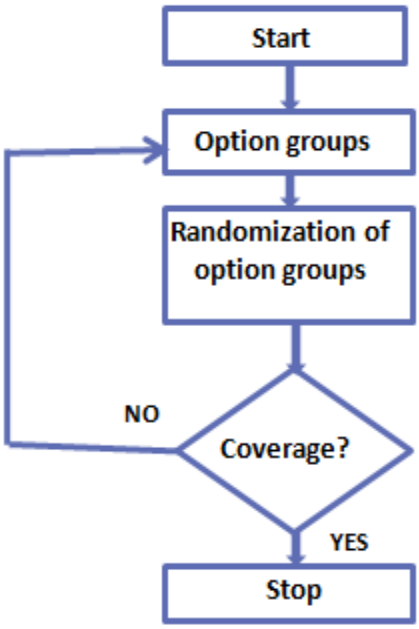

Fig 2.Option groups randomization algorithm

If some bug is missed from full regression or found from other regressions run. Then after fixing the bug again need to run all regressions. This generally requires seven day. But now from proposed regression optimization technique, which can have a flavor of all regressions in one day by making regression qualification efficiency more. This saves time of running all regressions once again after bug fix. This novel technique can also be used for design qualification purpose to save time and resource.

V. RESULTS AND DISCUSSION

In this paper proposed regression optimization technique's four day run results as machine cycles, CPU time and coverage are compared against the traditional regression four day run results of the configurable IP core L2 cache.

The Table 1 contains the result of four day regression runs of both traditional and proposed optimization techniques. When first day results are compared 43.56% machine cycles, 44.02% CPU time is saved from regression optimization side but there is a very minimal loss of coverage 0.52% with respect to traditional coverage.

However, as run count increases day by day regression optimization is able to hit more or less same coverage as the traditional one .From Table 1 it can be observed that for second day run the coverage loss decreases where as for third and fourth day run result there is a coverage gain of o.13% and 0.03% by saving nearly 40% machine cycles and CPU time.

Similarly on the fourth day it can be observed that there is a slight decrease in gain of coverage .This doesn't shows the inefficiency of the proposed optimization technique on the fourth day, Instead it is due to covering all the easy hit cover bins till third day after that on fourth day to cover hard to hit one requires more cycles and this similar trend can be might repeat if the regressions are run further few more days.
As the target coverage will be already covered on the initial days itself leaving over the hard to hit one for the left over days. In summary it is wise to stop regression runs once the target coverage is achieved.

Table 1.Comparison of Traditional and Proposed regression optimization techniques results for four day runs in terms of machine cycle, CPU Time and coverage.

Day		Traditional Regression Result	Proposed Regression Optimization Technique Result	Inference
1st Day	Machine Cycles	1,330,854,818	751046144	43.56% saved
	CPU Time (sec)	43,691,574	24456718	44.02% saved
	Coverage	71.83%	71.45%	0.52% loss
2nd Day	Machine Cycles	2,734,723,751	1497450291	45.24% saved
	CPU Time (sec)	88,565,492	49015570	44.65% saved
	Coverage	72.17%	71.94%	0.31% loss
3rd day	Machine Cycles	3,890,116,281	2245852448	42.26% saved
	CPU Time (sec)	125,615,234	73483764	41.5% saved
	Coverage	72.19%	72.29%	0.13% gain
4th day	Machine Cycles	4,849,921,413	2992540516	38.29% saved
	CPU Time (sec)	157,589,881	97908154	37.87 %saved
	Coverage	72.28%	72.3%	0.03% gain

The four day run vdbs are merged and generated urg report for both traditional and proposed regression optimization technique which is shown in Fig 3 and Fig 4.

However, as run count increases the coverage also comes closer to the traditional coverage run this because of randomization. In traditional run for multiple count run the previously covered features are executed multiple times.

Due to which those features contribute nothing to the coverage and multiple runs to hit corner cases only consumes too many machine cycles, CPU time and resources.

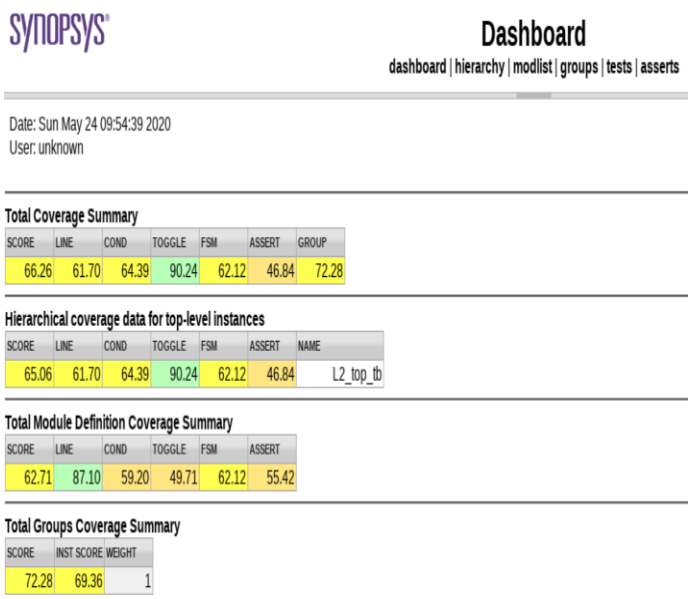

Fig 3.Coverage urg report from traditional regression result for four day run

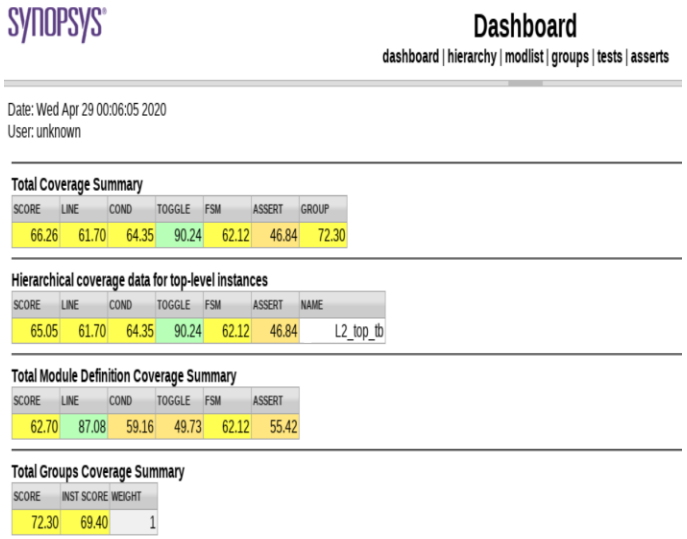

Fig 4.Coverage urg report from proposed regression optimization technique result for four day run

In case of the proposed regression optimization technique for multiple runs able to hit nearly more or less coverage as traditional one by saving machine cycles, CPU time and recourses for same count of runs.

The Fig 5.Shows the graphical representation of the Machine Cycles consumption for day to day run of both Traditional and proposed optimization technique of regressions.

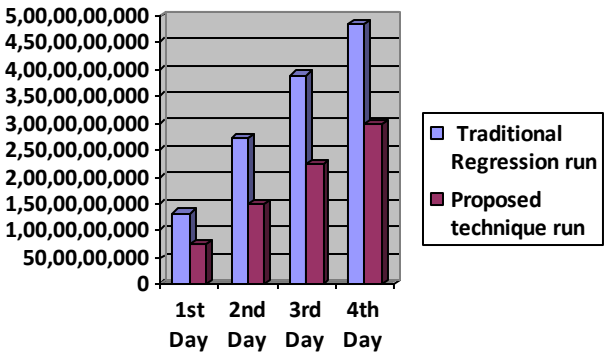

Fig 5.Number of Machine Cycles Consumed for four days regression runs for traditional and proposed optimization technique.

The Fig 6.Shows the graphical representation of the CPU Time consumption for day to day run of both Traditional and proposed optimization technique of regressions.

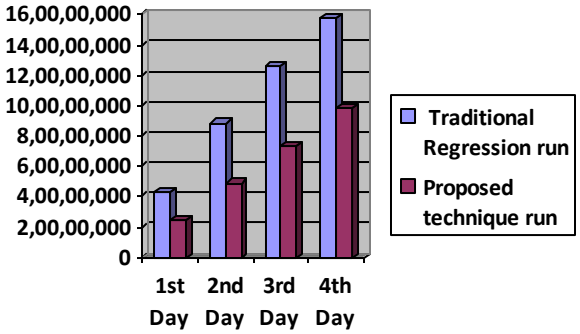

Fig 6.Number of CPU Time Consumed for four days regression runs for traditional and proposed optimization technique.

The Fig 7.Shows the graphical representation of the Coverage hit for day to day run of both Traditional and proposed optimization technique of regressions.

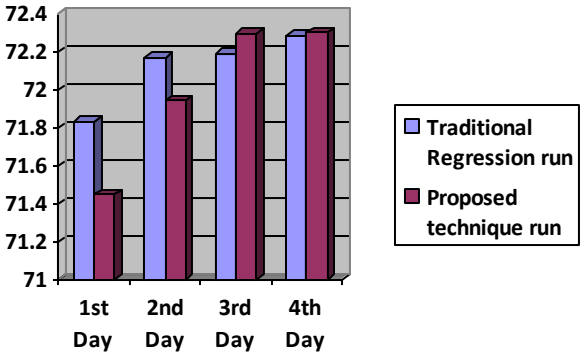

Fig 7.Coverage results of four days regression runs for traditional and proposed optimization technique.

The following Table 2.Shows how the optimized regression differs from the Traditional regressions and states its liberations. It also describes the capability of optimized regression technique over to the Traditional regression technique.

Table 2. Use case Comparison between Traditional and Proposed Optimized techniques

S.No	Use Cases	Traditional Technique	Proposed Optimized technique
1	Run Time	Requires More	Requires Less
2	Machine Cycle Consumption	More	Less
3	CPU Utilization	More	Less
4	Last Minute Changes Verification	Requires to run for many days to complete all regressions	Few days required as the regressions result has flavors of all the regressions
5	Qualification Test Efficiency	Less	More

V.CONCLUSION AND FUTURE WORK

In this paper results of configurable IP core L2 cache traditional regression runs are compared with proposed regression optimization technique. Both regressions are run for four days and observed that proposed regression optimization technique is consistently producing better result over the period of time. It can be observed from the regression runs results that proposed technique able to achieve nearly same coverage as of traditional regression technique with the overall saving of 42.33% machine cycles, 42.01% of CPU time.

Future work of proposed regression optimization technique aims to grade the test cases based on the coverage contribution which is further expected to save considerable number of machine cycles and CPU time by eliminating the test cases which are not contributing to the coverage making regression runs more compact and efficient.

ACKNOWLEDGMENT

The author would like to thank and acknowledge with gratitude, Both L2 and QDSP team of Qualcomm Technologies, Inc for providing an opportunity to carry out this work and for supporting at each and every phase of the work.

REFERENCES

[1] Marek Cieplucha.2019. "Metric-Driven Verification Methodology with Regression Management", Journal of Electronic Testing, Volume 35, Issue 1, pp 101–110.
[2] Manaswini B, Rama Mohan Reddy A.2019. "A Shuffled Frog Leap Algorithm Based Test Case Prioritization Technique to perform Regression Testing". International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249-8958, Volume 8 Issue 5.
[3] Paula Branco, Luis Torgo, Rita P. Ribeiro, Eibe Frank, Bernhard Pfahringer, Markus Michael Rau.2018. "Learning Through Utility Optimization in Regression Tasks". IEEE International Conference on Data Science and Advanced Analytics (DSAA), DOI: 10.1109/DSAA.2017.63.
[4] Raul H. Rosero, Omar S. Gomez, Glen Rodriguez. 2017. "An approach for regression testing of database applications in incremental development settings". IEEE 6th International Conference on Software Process Improvement (CIMPS), DOI: 10.1109/CIMPS.2017.8169952.
[5] Marcela Zachariaova, Michaela Kekelyova-Beleova, Zdenek Kotasek.2016. "Regression Test Suites Optimization for Application specific Instruction Set Processors and Their Use for Dependability Analysis". IEEE Euromicro Conference on Digital System Design (DSD), DOI: 10.1109/DSD.2016.50.
[6] Ramzi A. Haraty, Nashat Mansour, Lama Moukahal, and Iman Khalil.2016. "Regression Test Cases Prioritization Using Clustering and Code Change Relevance," International Journal of Software Engineering and Knowledge Engineering, Volume 26, Issue 5, pp 733–768.
[7] Erik Rogstad , Lionel Briand.2015. "Clustering Deviations for Black Box Regression Testing of Database Applications", IEEE Transaction on Reliability, Volume 65, Issue 1, pp 4-18.
[8] Daniel Di Nardo, Nadia Alshahwan, Lionel C Briand, Yvan Labiche.2015. "Coverage based regression test case selection, minimization and prioritization: a case study on an industrial system", Software Testing, Verification & Reliability, Volume 25, Issue 4, pp. 371-396.
[9] J. Anderson, S. Salem, and H. Do.2014. "Improving the Effectiveness of Test Suite Through Mining Historical Data", Proceedings of the 11th Working Conference on Mining Software Repositories, pp 142–151.
[10] Albert Pravin, Subramanian Srinivasan.2013. "Effective Test Case Selection And Prioritization In Regression Testing", Journal of Computer Science, Volume 9, Issue 5, pp. 654-659.
[11] Erik Rogstad, Lionel C Briand and Richard Torkar.2013. "Test case selection for black-box regression testing of database applications", Information and Software Technology, Volume 55, Issue 10, pp 1781-1795.
[12] S. Suman, Seema.2012. "A Genetic Algorithm for Regression Test Sequence Optimization". In International Journal of Advanced Research in Computer and Communication Engineering, pp. 478–481.
[13] Bo Guo, Mahadevan Subramaniam, Parvathi Chundi .2012. "Analysis of Test Clusters for Regression Testing", IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST), pp. 736-736.
[14] Mark Harman.2011. "Making the Case for MORTO: Multi Objective Regression Test Optimization". IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, DOI: 10.1109/ICSTW.2011.60.
[15] Vipindeep Vangala, Jacek Czerwonka, Phani Talluri.2009. "Test Case Comparison and Clustering Using Program Profiles and Static Execution", 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering, pp.293–294.