An Automated Collective Approach to Verification Coverage Merge

Xu Huang[1, a], YiRu Fu[2,b], LinTao Liu[3,c] and LunCai Liu[4,d]

[1]Sichuan Institute of Solid State Circuits, Chongqing, P.R. China

[2]Sichuan Institute of Solid State Circuits, Chongqing, P.R. China

[3]Sichuan Institute of Solid State Circuits, Chongqing, P.R. China

[4]Sichuan Institute of Solid State Circuits, Chongqing, P.R. China

[a]hxtt103@163.com, [b]fuyiru@sisc.com, [c]hgdllt@sisc.com, [d]llc@sisc.com

Keywords: Coverage,Verification

Abstract. Increasing design complexity has increased the number of tests required to verify our project, which makes merging of coverage databases inefficient and time-consuming. Using an automated iterative collective approach addresses that bottleneck by continuously merging with existing coverage totals the information from individual simulations as they finish. It also shortlists tests that add incremental value to overall coverage percentages.This Optimized testlist is used to qualify major design changes quickly and efficiently.

Introduction

This paper presents the results of our effort to improve the efficiency of merging coverage data for large regression runs.Our work resulted in an iterative method that merges coverage results on the fly rather than waiting for the entire set of verifications to complete before kicking off a massive parallel merge.

In the rest of this paper,Section Problems with Existing Approach describes the current implementation and its problems.Section An Automated Iterative Collective Approach describes how the automated iterative merge approach works.Section Regression Flow describes our regression flow, and how our approach fits into it.Results and conclusions are discussed in the final Section.

Problems with Existing Approach

The number of tests required to verify our project has increased over the years due to increasing design complexity. We run thousands of directed and random tests on our regression system with functional coverage enabled. These tests take varying amounts of time to run on our regression system, with some taking several days to finish. Conventional approaches require all tests in a particular regression run to complete before merging the coverage data.

The long latency for starting the coverage merge poses infrastructure and schedule issues. Generating a weekly coverage report with this conventional method is impossible because some simulations require nearly a week of run time, and the parallel merge takes an extra day to generate a report. Because we collect several hundred gigabytes worth of coverage data from one regression, these delays risk filling the disk. As a result, some of these long-running tests may not get merged into the weekly report. An alternative approach to omitting long-running tests from the report is overlapping regression runs, which requires enough resources to handle multiple runs[1].

Because corner cases often require many weeks of random simulation to verify a design completely, an individual run may have an unacceptably low coverage scores. This is exacerbated by the omission of coverage data from long-running tests due to resource issues. To obtain a complete picture of how well the design is covered, it is necessary to look at the cumulative coverage results over many different simulation runs.

An Automated Iterative Collective Approach

The process of dispatching a large regression, waiting for completion, and then merging the coverage data grows inefficient with each increase in the number of tests. we developed an iterative and automated flow that dynamically merges coverage data from individual completed simulations with existing coverage totals. With this method, waiting for all tests to complete before starting the parallel merge is no longer required[2].

we developed a novel design that dynamically merges coverage result databases as each individual test completes. In this paper, we refer to these databases, stored as .vdb files, as DBs. We have a centralized daemon that accepts incoming DBs from all the passing tests. Coverage data from failing tests (in black in Figure 1) are discarded. Each incoming DB is then merged into a single cumulative coverage database. We call this the cumlative merged database, or CMDB. We take advantage of VCS's flexible merging to overcome differences in coverage terms from multiple regression and random test runs with different models, using the CMDB's coverage model as the basis. Flexible merging allows the coverage reporting to adapt to changes in the coverage model as a project progresses. To utilize flexible merging fully, frequent promotion of the coverage model in the CMDB is employed so new coverage terms from more recent regression models will be included.

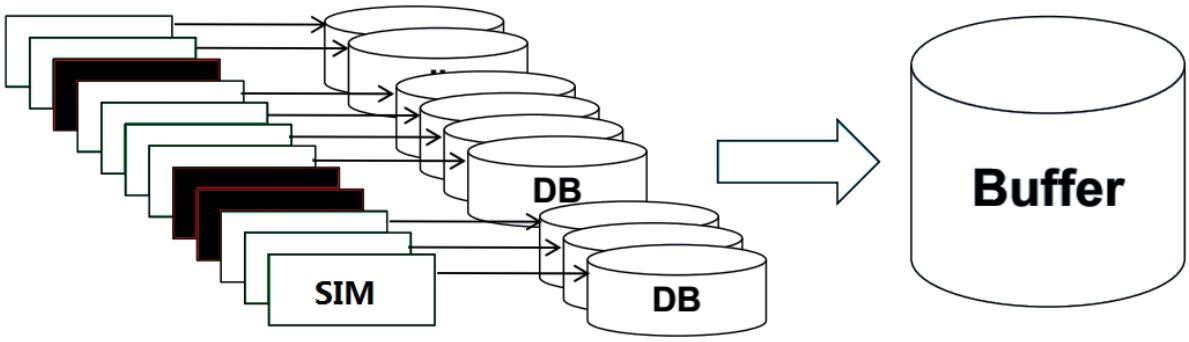

Figure.1. Sending individual DBs to the incoming buffer queue.

The automated iterative merge approach also identifies tests that add incremental value to the overall coverage results. The CMDB has the capability to register a callback to the test script that submitted the coverage database. We can immediately assess the incremental impact of the given database; if the incremental value meets a desired threshold, the callback script saves a copy of the incoming DB for further processing. We then use these tests to create a smaller, optimized testlist to qualify major design changes with a faster turn-around time. This provides our engineers more confidence when qualifying their changes with this reduced testlist. The flow logs all activity for debugging purposes and generates an hourly Unified Report Generator (URG)report to display the current coverage results [3]. Having real-time coverage results available allows us to determine the effectiveness of newly added tests quickly.

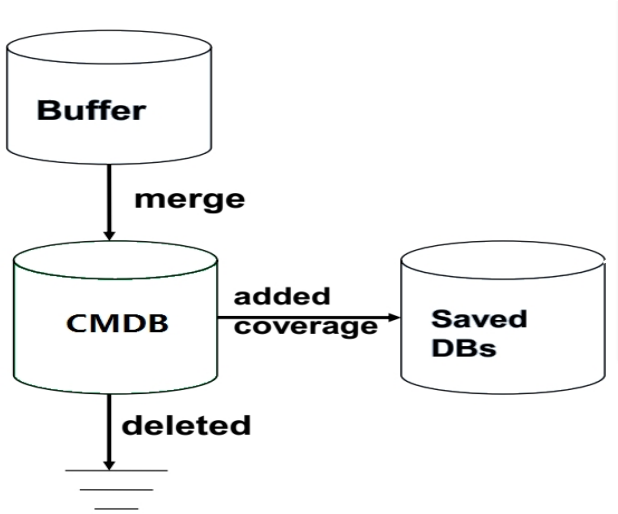

Figure.2. An Automated Iterative Collective Approach to Verification Coverage Merge

Coverage data from long-running tests will merge with existing coverage totals on eventual completion, thereby improving coverage results. Furthermore, by using our approach across multiple regression runs, we are able to plug holes covered by our random tests and create a cumulative functional coverage report. Our engineers use this cumulative report to decide if new tests need to be written to exercise hard-to-reach cases. Figure 2 shows how our iterative automation of VCS coverage merge works.

Regression Flow

we run daily regressions consisting of thousands of directed and random tests with functional coverage on a server farm. Each regression contains different design hierarchies and coverage terms depending on how many changes were made the prior day. Each test is different and has varying amounts of run time on the farm. Some tests take a few hours to complete, while others take a few days and some as long as a week. When a test finishes, it generates a DB in the directory of incoming DBs mentioned in Section An Automated Iterative Collective Approach. Our process buffers all these DBs and merges them, one by one, with our CMDB. All tests that improve coverage are saved and further graded to optimize the testlist of saved DBs. This optimized testlist will contain the fewest number of tests that exposes the most functional coverage. It is used by engineers to qualify their changes. Figure 3 shows how our approach of VCS coverage merge fits in our overall regression flow.

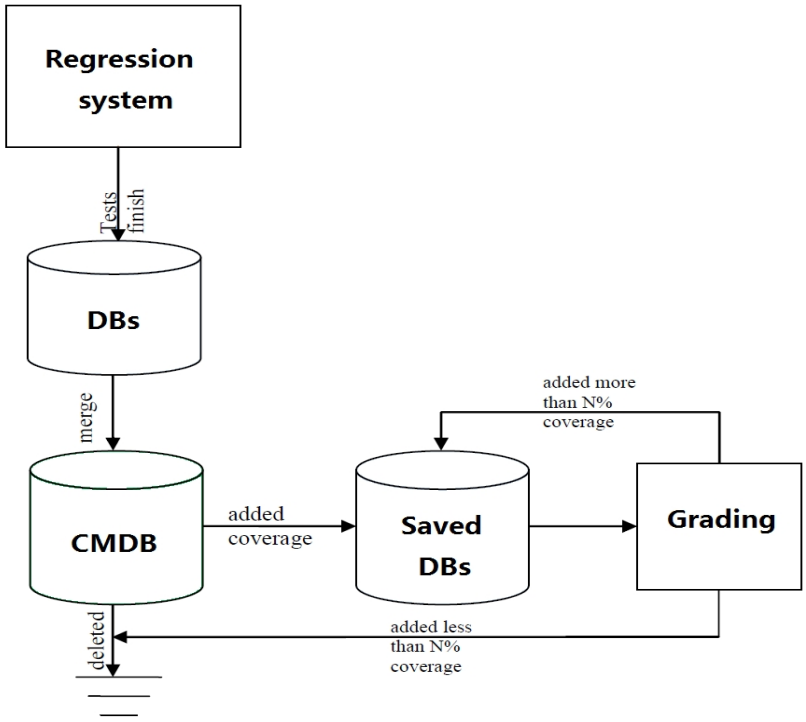

Figure.3. Using the automated iterative collective approach in our regression flow

Results and Conclusions

Our automated regression approach reduced peak usage by approximately 90% compared to the previous method of running a parallel merge on conclusion of the regression run. Peak compute requirements were also reduced, because a single machine used to run the coverage-merging daemon replaces the need to have a large set of machines and infrastructure available to support a large parallel merge process. Since the parallel merge is no longer in the critical path for obtaining results, it is also unnecessary to devote the fastest hardware for coverage merging.

The coverage analysis performed during the merge process acts as a filter to identify quickly a set of tests that cover the design well. We identified a set of 6% of our entire suite of tests that achieved a sufficient threshold of coverage for basic qualification of code check-ins. This set of tests is small enough that additional coverage grading can be used to create even smaller sets of tests suitable for first-order qualification of design change.

The continuous nature of the automated iterated approach means current progress can be monitored at any point during regression runs. It avoids problems caused by storing large amounts of data while waiting for the longest tests to complete, and allows overlapping regression runs to execute without the risk of filling up the disk. In addition, it allows us to accumulate results across multiple runs of random tests, giving us an accurate reading of cumulative coverage achieved thus far. Cumulative coverage is important to us because extremely hard-to-hit cases take time and several regression iterations before they get exercised, so having a cumulative report gives engineers better clarity on where our real coverage holes are.

References

[1] IEEE Computer Society, "IEEE Standard for SystemVerilog – Unified Hardware Design, Specifi-caton, and Verification Description Language", IEEE Std 1800-2009.

[2] Harry Foster, “Semantic Inconsistency and its effect on simulation”, IEE Electronics Systems and Software, April/May 2003

[3] VCS User Guide,Synopsys.Inc.