Measuring the Gain of Automatic Debug

Presented at the Microprocessor Test and Verification Conference (MTVCon13)

Daniel Hansson
Verifyter AB
Lund, Sweden
daniel.hansson@verifyter.com

Heli Uronen-Hansson, PhD
Department of Experimental Sciences and Medicine
University of Lund
Lund, Sweden
heli.uronen-hansson@med.lu.se

***Abstract*—The purpose of regression testing is to quickly catch any deterioration in quality of a product under development. The more frequently tests are run, the earlier new issues can be detected resulting in a larger burden for the engineers who need to manually debug all test failures, many of which are failing due to the same underlying bug. However, there are software tools that automatically debug the test failures back to the faulty change and notifies the engineer who made this change. By analyzing data from a real commercial ASIC project we aimed to measure whether bugs are fixed faster when using automatic debug tools compared to manual debugging.**

All bugs in an ASIC development project were analyzed over a period of 3 months in order to determine the time it took the bug to be fixed and to compare the results from both automatic and manual debug. By measuring the time from when the bug report was sent out by the automatic debug tool until the bug was fixed, we can show that bugs are fixed 4 times faster with automatic debug enabled. Bug fixing time was on average 5.7 hours with automatic debug and 23.0 hours for manual debug. The result was achieved by comparing bugs that were automatically debugged to those issues that could not be debugged by the tool, because those issues were outside the defined scope of the device under test. Such issues are still reported by the automatic debug tool but marked as requiring manual debug and is consequently a good point of comparison.

A 4 times quicker bug fixing process is significant and can ultimately contribute to a shortening of a development project as the bug turnaround time is one of the key aspects defining the length of a project, especially in the later phase just before release.

Keywords—regression testing; system-on-chip verification; automatic debug; continuous integration

I. Introduction

During product development of ASIC's or software, new bugs are continuously introduced by mistake causing the quality of the product to deteriorate. In order to capture these bugs, regression tests are run regularly. This is often done manually by the test engineers and is labour intensive.

Software tools that automatically map the test failures to the faulty change currently exist. These tools are also able to notify the engineer who made the faulty change. There are two ways to do automatic testing: the first method is to test each change individually before it is allowed to be integrated into the product, a methodology called continuous integration and is supported by tools such as Jenkins[1], BuildBot[2] and CruiseControl[3]. The second approach is used by the automatic debug tool PinDown[4], which retests failing tests on older revisions of the product until it finds the first change that caused the test to fail. The advantage of PinDown is that it can handle larger test suites and random tests, whereas the continuous integration tools rely on running a short non-random test suite on every change in order to directly compare the results achieved before the change was made. The advantage of the continous integration tools is that they are often open source and simple to use, making them perfect for many types of software development where random testing is not used and test times are limited. It is also possible to use both methodologies: running continuous integration as a pre-integration filter and PinDown for automatic debug of the larger post-integration test suites. The results in this paper are valid for both types of tools.

In this paper we measure the time between the failure report is emailed out until the bug is fixed. We measure how the time differs depending on whether the regression failure has been automatically debugged, i.e. whether the failure report contains information about the change that caused the test failures, including who made the change.

II. Methodology

A. *How the measurement was done*

We manually analyzed all regression bugs, i.e. bugs that introduced new failures in previosly working functionality, in a commercial ASIC development project based on random testing over a period of 3 months. For each bug we measured the time between the failure report was emailed out until the bug was fixed in the revision control system. We grouped the results depending on whether automatic debug had been performed, i.e. whether the report contained information about the change that caused the test to fail and who was responsible for that change.

This project used PinDown for automatic debug of regression tests. PinDown was set up to automatically debug failures in RTL and the testbenches, but not for example internally developped tools (a setup choice, not a tool limitation), the latter having to be manually debugged. This was also the case with some other test environment aspects. Some of these tools contained a model of the expected behaviour, which was compared to the actual outcome of the

RTL simulation. This meant it was not possible to immediately determine from a test failure report whether the failure was due to the failures in the RTL, testbench, tools or elsewhere. Most regression bugs were automatically debugged, but due to this setup some had to still be manually debugged, allowing us to measuring the difference in bug fix time. In both cases PinDown emails out the report, giving us a common starting point, which is independent of whether automatic debug had been performed. All failure reports were sent to all project members, but the bug was only assigned to an engineer if automatic debug had been performed.

The difference in bug fix time cannot be explained by a difference in complexity between the areas that were automatically debugged, e.g. RTL and the testbench, and those areas that were manually debugged, e.g. the internally developped tools. Regression bugs are most often mistakes, where the engineer forgot or was unaware of some important aspect of the design. Even if the product may be very advanced, the regression bugs are often not complex. If the regression bug is complex, then it is always possible to simply undo the change. In this project the policy was to keep the quality of the product high at all times and in the few cases when the engineers could not fix the problem reasonably fast they undid the faulty change and worked on the problem locally. From a regression testing point of view the bug fix is the change that makes the tests pass again, independent of whether it is an acutal fix of the problem or if the faulty change is simple removed.

Availability of the engineers was not a problem in this project. This was the project with the highest priority and fixing regression bugs had the highest priority, as it hampers the development and verification of the other team members.

B. *What we did not measure*

In this paper we don't examine the time from when the bug is introduced until it is reported. We only measure the time from the failure is reported until the bug is fixed. We do not measure how much time it takes to run the tests in order to detect the test failures in the first place and we also do not measure how much additional test time was required in order to perform the automatic debug. The test time varies a lot between projects, mostly dependendent of how long time it takes to run the entire test suite, but the time between report and fix is independent of the size of the test suite, as it is only dependent on the time it takes to run one failing test during either manual or automatic debug. Consequently the results in this paper is valid independent of the time it takes to run the entire test suite.

Using continuous integration tools or an automatic debug tool (such as PinDown) introduces extra test time, i.e. extra computer time, which delays the failure report. This means the failures are seen later by the engineers. This paper does not measure how much this delay in reporting affects the overall time to fix the bugs. However, this reporting delay is compensated by two other aspects which we don't measure either: 1) the fact that automatic debug occurs 24/7 unlike manual debug, which only occurs during office hours. Such delay in reporting may not noticeable by the engineer who will fix the bug because at the report time the engineer may not be working.

2) automatic debug allows more frequent testing as the project is not limited by human manpower to debug the test failures. More frequent testing means bugs will be found earlier.

To summarize, we don't measure the time from the introduction of the bug until the report is sent out. If automatic debug occurs during office hours it may delay when the engineers are made aware of the failure. On the other hand, automatic debug is expected to reduce the time to detection of a new failures by making more frequent testing possible.

Statistical Analysis

The graphs were generated and the statistical significance estimated using GraphPad Prism software (GraphPad), with unpaired two-tailed Student's t test. P values <0.05 were considered statistically significant.

III. Results

A. *4x Faster Bug Fixing with Automatic Debug*

In total 39 bugs over a 3 month period were measured. In figure 1 we show the time from when the report was sent out until the bug was fixed. The average time for failures that had been automatically debugged by PinDown was 5.7h. These reports were assigned directly to the engineer who had caused the faulty change. The corresponding times for reports that only contained test failures and required manual debug was 23h on average. Thus automatic debug speeds up bug fixing approximately 4 times.

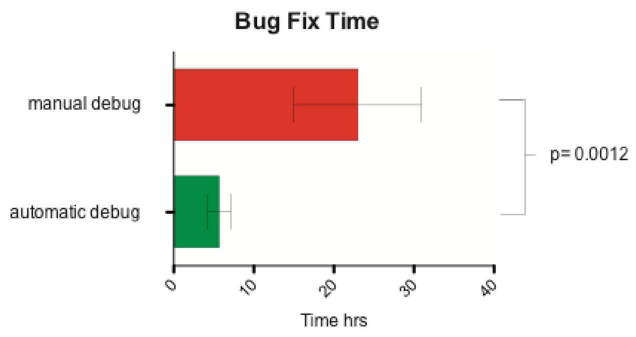

Figure 1. Comparison of manual and automatic debug time. Time was measured from failure report until issue was fixed in the revision control system. A statisctically significant difference (p=0.0012) between manual and automatic debug was found. With automatic debug, bug fixing time is 4 times faster.

ASIC development projects are exposed to two different types of functional failures: build failures and test failures. A build failure is a failure to assemble the product, e.g. compilation errors. A tests failure is defined as any failure due to a mismatch of the expected data while exercising the functionality of the device under test. Build failures are faster to detect, as the build step must be done prior to running the tests. They are often easier to debug as there is often information about the exact line of code that has caused the failure. Tests can typically only be running once a successful build has been assembled. Thus test failures take more time to detect and are harder to debug as there is rarely any information about which line numbers that are the cause of the these failures.

Removing the build failures and only looking at the test failures, the bug fix time was 7.4h for automatic debug from when the report was sent out and 26.1h when manually debugging. This corresponds to total speed-up of bug fixing by 3.52 times or 4 times using just one significant figure, which better reflects the precision in these kind of measurements.

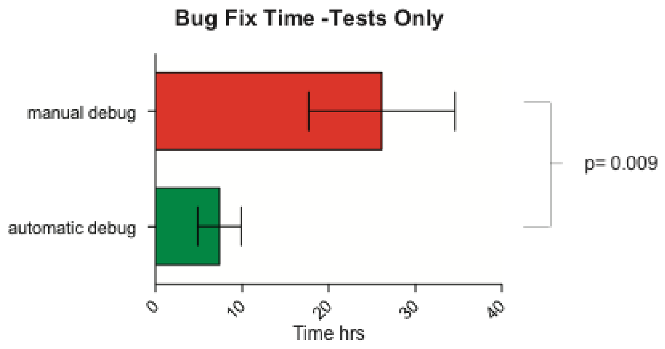

Figure 2. Comparison of manual and automatic debug time including only test failures and no build failures. Time was measured from failure report until issue was fixed in the revision control system. A statistically significant difference (p=0.009) between manual and automatic debug was found. With automatic debug, bug fixing time is approximately 4 times faster, same as when build failures were included see Fig 1.

B. *Why is Bug Fixing 4x Faster with Automatic Debug?*

Next, we aimed to understand why automatic debug is faster than manual debug. For this purpose we first measured the amount of emails discussing each failure. We measured the number of replies to the email bug report sent out before the bug was fixed, not counting out-of-office emails. In figure 3 we show that failures that were automatically debugged generated 0,5 emails per report on average, compared to 2,6 emails per report for failures that were manually debugged. These results show that manual debug generates 5 times more email discussions than automatic debug before the issue is fixed.

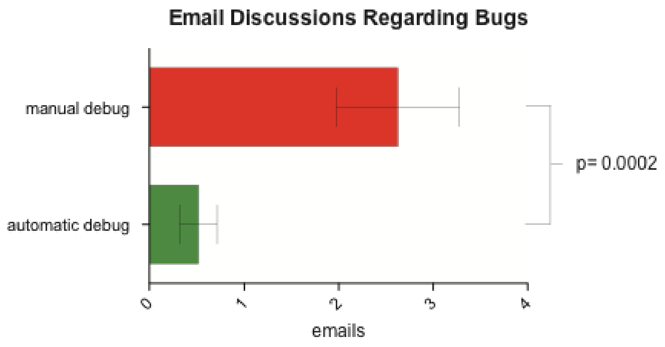

Figure 3. Comparison of the number of emails concerning test failures after manual and automatic debug. A statistically significant difference (p=0.0002) between manual and automatic debug was found. With manual debug there are 5 times more emails discussing failures compared to automatic debug.

In order for the issue to be manually debugged the same day, somebody has to be in the office to do the work. Due to this practical constraint, we next measured at what time of the day bugs are usually inserted as this limits how fast failures can be manually debugged. We listed number of bugs introduced per office hour, where the time zone is that of the responsible engineer. By analysing the results we found that most bugs are introduced late afternoon, evening, with a median and average at 4 pm. Very few bugs are introduced before lunch.

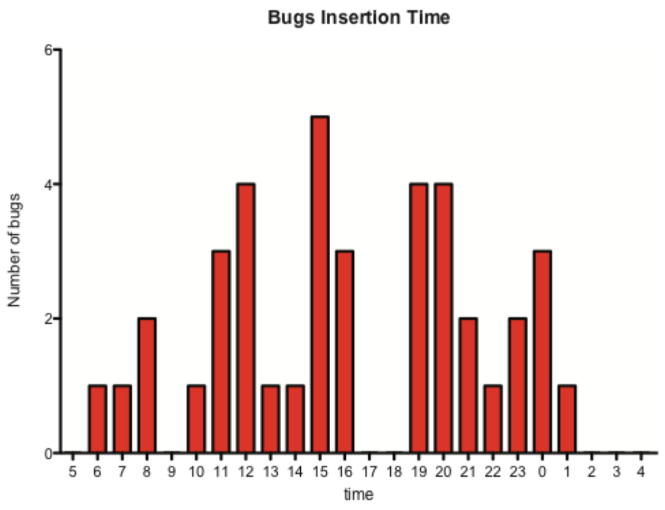

Figure 4. Time of day when bugs are introduced in the revision control system. The time zone is that of the engineer creating the bug. The average and median time of day is 4 pm.

Taken together, automatic debug is faster than manual debug: Bugs are introduced late in the day, leaving little time for same day manual debug, whereas automatic debug continues around the clock and is not sensitive to the concept of office hours. If the issue has been automatically debugged, the assigned engineer fixes the problem with little discussion with the collegues. In contrast, manual debug first generates discussion before the issue can be fixed.

IV. SUMMARY

In this real commercial ASIC project we saw that regression bugs were fixed 4 times faster when the reported failures had been automatically debugged. Manual debug required 5 times more discussion before the issue was solved, compared to automatic debug. The advantage of automatic debug is that the causing change is known and it is also known who made that particular change, thus reducing the need and scope for discussions. As bugs were inserted late in the day, 4 pm on average, there was little time for same day manual debug, which meant the bug was fixed some time next day (23h after the report on average). Automatic debug on the other hand is performed 24/7 and was able to report errors independent of time of the day, directly to the responsible engineer, which resulted in a fix within hours (5.7h on average).

As the bug turnaround time is one of the key aspects defining the length of a project, especially in the later phase before the release, fixing the bugs 4 times faster is bound to make a difference to the project release date.

REFERENCES

[1] http://jenkins-ci.org

[2] http://buildbot.net

[3] http://cruisecontrol.sourceforge.net

[4] http://www.verifyter.com