KEY LEARNINGS FROM PRE-SILICON SAFETY COMPLIANT BOOTROM FIRMWARE DEVELOPMENT

Chidambaram Baskaran, Pawan Nayak, R.Manoj,
Sampath Shantanu and Karuppiah Aravindhan

Texas Instruments India Ltd, Bangalore, India

ABSTRACT

Safety needs of real-time embedded devices are becoming a must in automotive and industrial markets. The BootROM firmware being part of the device drives the need for the firmware to adhere to required safety standards for these end markers. Most software practices for safety compliance assume that software development is carried out once the devices are available. The BootROM firmware development discussed in this paper involves meeting safety compliance need while device on which it is to be executed is being designed concurrently. In this case, the firmware development is done primarily on pre-silicon development environments which are slow and developers have limited access. These aspects present a unique challenge to developing safety compliant BootROM firmware. Hence, it is important to understand the challenges and identify the right methodology for ensuring that the firmware meets the safety compliance with right level of efficiency. The authors in this paper share their learnings from three safety compliant BootROM firmware development and propose an iterative development flow including safety artefacts generation iteratively. Concurrent firmware development along with device design may sound risky for iterative development and one may wonder it may lead to more effort but the learnings suggests that iterative development is ideal. All the three BootROM firmware development has so far not resulted in any critical bugs that needed another update of the firmware and refabrication of the device.

KEYWORDS

Concurrent development, Firmware development, Safety compliance, Pre-silicon software development.

1. INTRODUCTION

The challenges to coordinate the different product developments activities have dramatically increased with Concurrent Engineering and Integrated Product and Process Development [1]. In order to accelerate the time to market, firms attempt to overlap the different activities in product design and development – leading to iterative overlapped development. Safety software development has typically followed the traditional highly-structured approaches such as V-model or waterfall [2]. The V-model [3] is composed of well-defined 9 steps through project initiation, design, test, maintenance and phase-out. A recent study of the safety software development and agile development [4] indicates that the agile methods have been not adopted significantly. When there is a need to adopt these methods to specific safety software development with constraints such as concurrent development and limited access to test environments, there is not much study done that can be beneficial and reused. Authors in this paper attempt to provide few key learnings

David C. Wyld et al. (Eds): EMSA, SEA, AIFU, NLCAI, BDML, BIoT, NCOM, CLOUD, CCSEA, SIPRO - 2022
pp. 47-54, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.120605

from safety firmware development concurrent with hardware design in a constrained pre-silicon environment and show efficient ways to meet the safety compliance.

2. SAFETY SOFTWARE DEVELOPMENT

Automotive industry has adopted usage of electronic control units (ECUs) in a large scale within a very short period of time. Large number of processors are heartbeats of these ECUs and they perform several safety critical functions [5]. One of the key requirements for processors or devices being used in these functions is for firmware in the ROM of these devices to be safety compliant. The most popular standard for safety compliance is the ISO 26262 standard titled "Road vehicles — functional safety" [6]. The compliance to this standard needs' adoption of software practices and tools that to demonstrate the compliance to standard and ensuring quality of the software. This needs compliance across OEMs, their suppliers, and developers of automotive components. Part 6 of this standard [6] details the practices to be adopted by software developers. The standard requires well documented and detailed requirements followed by design details documentation and finally good test plans. These artefacts need to be thoroughly reviewed and also traceability of the requirements to design to test is critical to ensure quality of the delivered software. It is very essential to prove that the development meets the compliance requirements. Addition to the detailing the implementation aspects, compliance to coding standards though MISRA-C [7] and dynamic coverage of the code through testing is also mandatory. The final resulting firmware must be well tested and test results produced to show that the firmware has zero possible bugs. Most of the literature details methodology and practices for safety software development that is significantly different from the pre-silicon firmware development presented by the authors in this paper. The authors discuss about safety compliance for firmware development while the device is being designed.

3. PRE-SILICON SOFTWARE DEVELOPMENT CONSTRAINTS

The firmware development discussed in this paper involves concurrent development while the device on which it is to be run is still being designed. This concurrent development of the device and the firmware enables shorter time to market as the firmware is put into the ROM as soon as the device design is completed and hence built into the fabricated device ROM. However, this poses several challenges in terms of availability of testing platform for firmware development as the actual device is still being designed. The testing platforms used for these scenarios are referred to as pre-silicon testing platforms. Several challenges of pre-silicon testing platform are listed below.

3.1. Speed of the pre-silicon Platform

Software developed pre-silicon needs a testing environment to test to ensure it has near zero defects. These test environments are very slow since the entire design of the device is emulated using another hardware. For example, for the devices for which the authors have developed the firmware these environments run at 100 KHz while the real device can run at close to 100 MHz. This slowness has a direct impact on the amount of time spent on testing. For example, for the firmware development needed almost 10 days of testing time due to the slowness of the environment.

3.2. Cost of the pre-silicon Platform and access time

The pre-silicon platform is very costly and typically only couple of platforms are available for each device design. These platforms are used by multiple teams due to hardware-software

concurrent development and hence different teams are provided a very short period of access to these platforms. For instance, the firmware team of 3 software engineers in total had access time of 40 hours per week – approximately 14 hours per week per engineer. Safety compliance needs several test results and artefacts to be generated and hence the slowness of the testing environment presents a unique challenge.

4. CONCURRENT DEVELOPMENT CHALLENGES

The authors worked on firmware development while the design of the device on which the firmware is expected to run was also being developed concurrently. This type of concurrent development introduces additional challenges to the safety compliance for firmware development.

4.1. Out of sequencing of features development

In this type of concurrent development - some of the features of the device may be available towards end of the hardware design and hence software team will have to develop these features without having any platform to validate them since the testing environment is built from the completed hardware design. This results in quite a bit of time gap between completion of design, implementation and testing of the firmware.

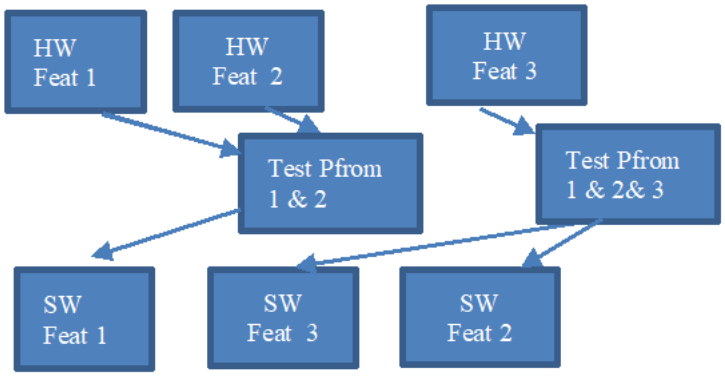

Figure 1. Out of sequencing of HW and SW features

Figure 1 shows timeline sequence of a scenario where the device hardware features implementation is in a different sequence compared to firmware features. This can happen as the effort to design the hardware feature and the related firmware feature may not be very similar and also team sizes working on these can differ. Due to these reasons out of sequence of development was found to be very common in all the 3 firmware projects. As a result, the test platforms for firmware testing may be available at a later point in time well beyond the implementation of the firmware.

4.2. Cross functional team bandwidth for reviews

Firmware software is usually reviewed by teams that are also involved in design of the device and the testing of the device. Many aspects of the firmware also pertain to aspects like device qualification, device characterization and hence the firmware design and implementation needs to be carefully reviewed by cross-functional teams. As the different teams involved in the device development concurrently, the availability of different team members for reviews is a challenge. For safety compliance it is important to review the design, implementation, test plans and test

results at the right time with right level of rigor. The reviews need to be recorded and quality of the reviews have to be met.

5. KEY LEARNINGS

Most of the literature discuss the challenges in meeting safety compliance in software development that is typically carried out on a platform where the final device on which this software needs to run is already available (referred to as post-silicon software development). Development of safety compliant firmware while the device itself is being designed is very special case which opens up new challenges. Authors in this paper discuss the key learnings from three such firmware development projects. The understanding of the constraints of the development environment, concurrent development and safety compliance challenges can enable in efficient and repeatable methodology for pre-silicon safety compliant firmware development

5.1. Safety process challenges for pre-silicon safety compliance in concurrent development

In this type of concurrent development - some of the features of the device may be available towards end of the hardware design and hence firmware team will have to develop these features without having any platform to validate them since the testing environment is built from the completed hardware design. This results in quite a bit of time gap between completion of design, implementation and testing. Authors in their first firmware development found that during the critical phases of the design and implementation cross functional teams were also nearing completion of their milestones leading to time constraints. This resulted in delays in reviews and feedback which are very essential for safety compliance. **Learning 1** – Ensuring the cross-functional team plans are well synchronized on a periodic basis and not just at the beginning of planning for dependencies on deliverables but also at completion of design feature wise helped in streamlining the development. The traditional firmware development focused on completing the entire design and then focused on reviews but the authors soon found out that each feature level review was more productive from better reviews as well as planning perspective.

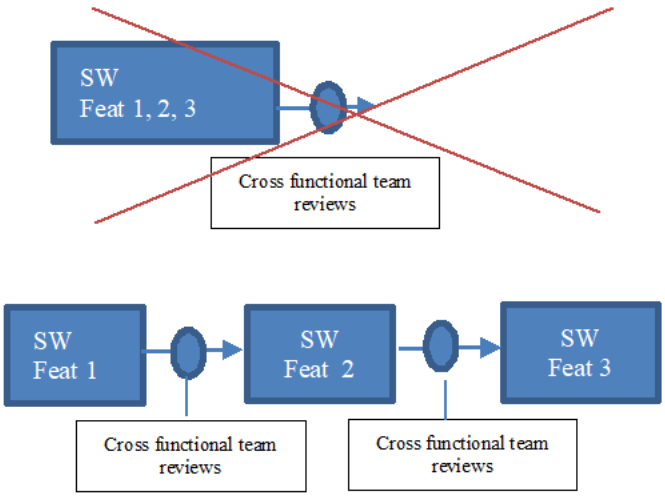

Figure 2. Balancing the reviews – periodic reviews

It is recommended that the availability of team members especially multi-functional team members who are also involved actively in their own domain deliverables is available for

reviews. Cross functional teams bring more insight into the design - the architect of the device has good overview of the usage requirements of a customer, while the team that is involved in device characterization can provide inputs on testing aspects. **Learning 2** – The sequence of the reviews is also very critical. Typically, firmware adds few new features while most of the other features are reused from prior devices. Focusing on the new features early - design review, test plan reviews enabled effective reviews early, better quality of the design and also provided sufficient inputs to improve the implementation for safety compliance. An incremental review process with new features being reviewed early has been found to be very effective. Interestingly these new features need to be reviewed also towards end of device design as the other teams involved in the design would have learnt a lot more of the details as well. This is a very unique review flow that authors identified to be effective in firmware development that is carried out concurrently with device design.

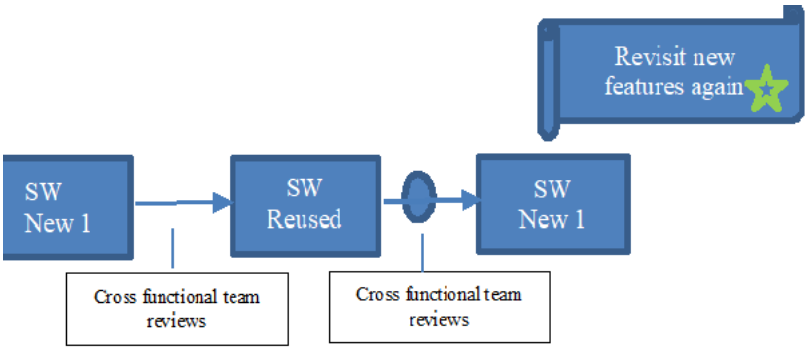

Figure 3. New features early and revisit reviews again at end

5.2. Safety process challenges for pre-silicon safety compliance in concurrent development – artefacts generation

Several artefacts need to be generated for safety compliance. It is important to understand the limitation of the testing environment and speed so that the generation of these artefacts can be planned better. **Learning 3** – The time for artefacts generation were overlooked in the first firmware development. The initial thought process was to generate some of the artefacts like code coverage report towards end of the firmware development so that final reports needed for meeting safety compliance can be made available. The time taken to generate these dynamic analysis report almost took 1.5x of the total testing time as the testing environment was not available continuously and the tests had to run and re-run to generate for any coverage gaps. Authors recommend that these dynamic analysis report generation be done module wise as and when they are completed to look for any code coverage gaps. This not only shortens the time to run (since it is done at a smaller module level) but also to quickly address the gaps to generate new tests to run.

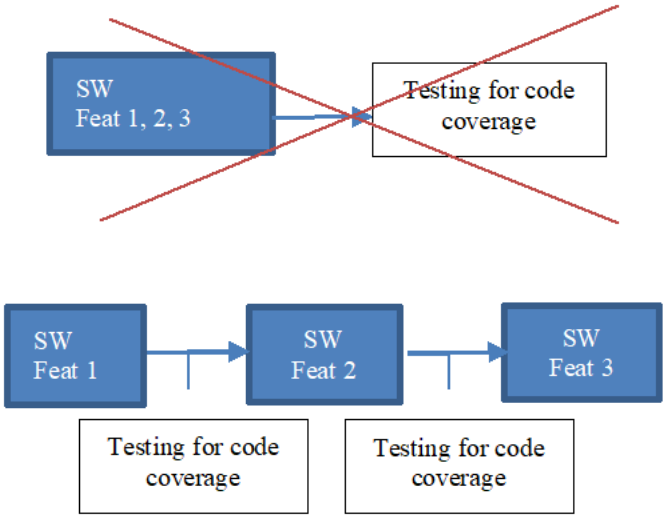

Figure 4. Dynamic analysis – code coverage iteratively

Authors recommend that the artefacts needed for safety compliance be classified into 2 classes – one that needs the test environment and one that is static – without needing any test environment. The artefacts that need the testing environment needs to be planned well for pre-silicon firmware development. It is highly recommended that these artefacts if they can be generated incrementally through the development cycle then they need to be generated periodically.

- Static artefacts – requirements, design document, test plan, traceability from requirements to design to test, MISRA-C compliance report
- Dynamic – needing test environment – test results, dynamic analysis

The dynamic analysis report (code coverage from testing) generation is heavily dependent on the testing environment and hence it is a key item to be planned well ahead. Each team member does not get a continuous access to the pre-silicon development environment and hence the generation of tests for coverage has to be planned well. In a post-silicon software development this is a not a key constraint as the environment to test is always available and each member may have exclusive test setup.

5.3. Safety process challenges for firmware code that is reused from non-safety development

The firmware development is usually is not written from scratch and multiple parts of the firmware is reused from older devices as well. One of the challenges in this reuse is that those reused pieces of firmware may not have gone through the safety compliance needs. Authors in their firmware development had significant portions of reused software and identified several artefacts that can help in identifying the quality of these reused software through mapping the functionality of reused software to safety features expected and identifying the level of rigor needed for safety compliance. **Learning 4** - The pieces of firmware that needed rigor was found to be portion of software that is involved in configuring registers in the device that can cause the functionality failure at run-time. Focusing on these aspects enabled building the rigor for the reused firmware pieces. The start-up booting time failures were made to return error values that can be handled at the application level and hence less rigor was needed for these failures. Further the reused firmware features were covered 100% with tests and traceability reports were

generated to ensure that these were fully tested. Dynamic analysis coverage was also another aspect added to ensure that the coverage of the reused firmware was close to 100%. These efforts saved significant time without having to go through code reviews and design reviews of the several thousand lines of code that were reused.

6. RECOMMENDATION FOR ITERATIVE DEVELOPMENT

The authors through their learnings from three safety compliant BootROM firmware development projects recommend that the development must be carried out iteratively. Concurrent development along with device design and development may sound risky for iterative development and one may wonder it may lead to more effort but the learnings suggests that iterative development is ideal.

- Suggested methodology is to first start with new features, complete and then move to reused or known features. The iterative development with new features designed, reviewed, tested followed by reused features ensures review rigor and early identification of problems.
- Iterative generation of safety collaterals - Iteratively generate the safety collaterals like design document, test cases and also generate reports from testing like dynamic code coverage through the feature development given the pre-silicon environment challenges.
- Revisit the new features design, test cases one more time towards end of the firmware development to look for newer understanding from the cross-functional teams as those teams also would have completed their implementations and tests for the new features. Several new findings and improvements were seen during the second round of reviews.

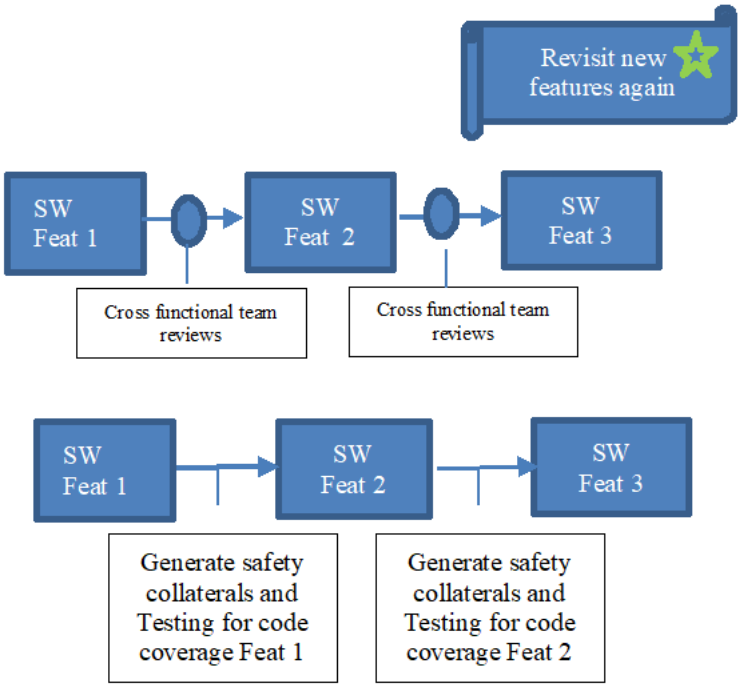

Figure 5. Iterative development flow

7. CONCLUSION

In order to accelerate the time to market, firms attempt to overlap the different activities in product design and development – leading to iterative overlapped development. Authors in this paper present learnings from such a development where there were additional challenges in needing to develop firmware concurrently with device design along with the limitation of pre-silicon platform. Interestingly it was observed that the iterative development of the firmware through new features first and then towards reused features provided optimal usage of time and effort. The constraints of the pre-silicon environment pushed for early test reports generation in an incremental manner so that the environment could be used efficiently. It was also found that it is essential to revisit the design, testing of the new features at the end of the firmware development to incorporate any new learnings from cross-functional teams as these teams also would have learnt from their own work. The synchronization of design and testing is a huge challenge due to different team sizes and efforts and hence ensuring one final design and test review when cross-functional teams have also progressed helped in identifying errors and solidifying the new features in firmware. This is another unique aspect recommended by the authors.

REFERENCES

[1] Browning, T.R., Eppinger, S.D., 2002. "Modeling impacts of process architecture on cost and schedule risk in product development". IEEE Transactions on Engineering Management 49 (4), 428–442.

[2] Roger S. Pressman, "Software Engineering: A Practitioner's Approach 7th Ed", MacGrawHill, p.40-41, 2010.

[3] Paul Rook, "Controlling software projects", IEEE Software Engineering Journal, vol. 1, no. 1, p.7-16, 1986.

[4] Rashidah Kasauli, Eric Knauss, Benjamin Kanagwa, Agneta Nilsson and Gul Calikli Chalmers "Critical Systems and Agile Development: A Mapping Study", 2018 44th Euromicro Conference on Software Engineering and Advanced Applications, 470-477.

[5] Georg Georgakos, Ulf Schlichtmann, Reinhard Schneider, and Samarjit Chakraborty. "Reliability challenges for electric vehicles: from devices to architecture and systems software" In Proceedings of the 50th Annual Design Automation Conference, page 98. ACM, 2013.

[6] ISO 26262-6:2018 Road vehicles — Functional safety — Part 6: Product development at the software level.

[7] Motor Industry Software Reliability Association et al. MISRA-C: 2004: Guidelines for the Use of the C Language in Critical Systems.

[8] Xiaocheng Ge, Richard F Paige, and John A McDermid, "An iterative approach for development of safety-critical software and safety arguments", In *Proc. of AGILE Conf.*, pages 35–43, Nashville, TN, USA, 2010. IEEE.

AUTHORS

Chidambaram Baskaran, Pawan Nayak, R.Manoj, Sampath Shantanu and Karuppiah Aravindhan are part of the Texas Instruments India (Ltd) with key areas of interest being embedded software development, ROM development and driver development for peripherals.