Application of Built-In-Testing in Component-Based Embedded Systems [*]

Irena Pavlova
Faculty of Math. and Informatics
Sofia University
Sofia,Bulgaria
i_ pavlova@gbg.bg

Mikael Åkerholm
CC-Systems, and Mälardalen University
Västerås, Sweden
mikael.akerholm@cc-systems.se

Johan Fredriksson
Mälardalen University
Västerås, Sweden
johan.fredriksson@mdh.se

Abstract

This work-in-progress paper discusses challenges with application of Built-In Testing (BIT) in component-based embedded-systems. Testing constitutes a large part of the time and budget in development of embedded software systems. Such systems are often mission-critical, making testing highly important, and at the same time testing em-bedded systems is challenging because of their limited observability. We investigate the possible application of BIT in components for embedded systems, as a technique to advance the technology and knowledge for analysis and verification of functional correctness, real-time behavior, safety, and reliability of these systems.

1 Introduction

Component-Based Software Engineering (CBSE) is known as the discipline of assembling new software systems by reuse of existing components, which should imply reduced development costs. Ideally every component corresponds exactly according to a specification that contains no questions of interpretation, and behaves exactly the same in any environment it is deployed. However, this ideal picture is often not the case, specifications may be ambigous, components may contain bugs and the bahaviour may vary in different contexts. This forces the execution of lengthy and costly reverification activities directly undermining the benefits of reusing the design and implementation efforts put into a component. Furthermore, most embedded systems have additional requirements not present in other systems, e.g., timeliness, low footprint, low energy consumption, etc. Such *extra-functional properties* also needs to be verified, adding even more (re-)testing needs to the system [4, 1].

To improve the overall benefits with reusing components we aim to investigate the application of Built-In Testing (BIT) in component-based embedded-systems. By developing a methodology for integrating BIT into COTS software, COMPONENT+ project has delivered a significant step on widening the application of self-test techniques for component-based software [3]. The proposed approach draws attention to built-test mechanism into components during design and coding, so that the successive testing and maintenance processes can be simplified. The results of the project prove that BIT application in the overall software process, enforcing testing and integration consideration from the very early stages of the development, help to build more reliable testing environments and support the realization of more comprehensive testing.

In the follwing section we give concrete forms of these areas through the presentation of a number of reserach quetsions, fore a more detailed presentation of these questions refer to [2].

2 Tailoring BIT for embedded systems

Different test methods have been used in software development for several decades. Practice has shown that standard testing techniques are inefficient for component based systems; especially for dynamic, reconfigurable and evolvable architectures like embedded software, where testing should be able to handle domain specific issues and should be committed with and complement to different types of analyses. Moving from traditional testing techniques to BIT in the specific domain of embedded systems presumes a lot of challenges and in the following section we try to identify them through the presentation of a number of research questions.

Q1: What is the benefit of BIT compared to standard testing?

A characterizing property of many embedded systems is that they are mission critical, e.g., control systems in vehicles, power plants and industrial automation equipment. In some systems faults may cause human injury, in others loss of money. Thus, to increase understanding, analysability and testability of systems, simplicity and static solutions is prioritised in many design decisions. To gain acceptance for BIT techniques it is important to find a positive answer to the question:

Q2: Can BIT techniques be applied with a net gain of verifiability?

It can be deemed practically impossible to include more functionality in a system without raising the complexity, and introducing BIT implies including more functionality. However, BIT techniques are supposed to increase testability. For a successful introduction of BIT techniques, an integration of techniques that clearly do the right prioritisation in the tradeoffs between the increased testability offered by BIT vs. the possible problems with decreased understanding, analysability, and testability. The embedded systems industry is generally interested in applying more analysis and formal methods for verification. However, because of the complexity in using these methods combined with scalability problems to deal with problems of industry size, the use is limited. A relevant question is:

Q3: How can BIT facilitate analysis?

Real-time aspects is one of the major differences between embedded software and PC or internet software. Embedded software systems often interact and controls physical processes with real-time requirements. Timing is often of first priority in testing efforts. To use BIT in embedded real-time systems it is important to understand the relation between the two. Hence we form the question:

Q4: Can BIT techniques be used to test timing, e.g., worst-case response-time and jitter?

Testing worst-case response-time is not trivial. Typically, tests assess within what time the embedded system reacts (creates an output considering an input). The assessed time forms an end-to-end latency for the response. Internally, the system typically involves transactions of several execution threads that must cooperate to create the correct output. These threads can in turn experience interference from other parallel activities in the system. Thus, the goal for the test is to measure the time from a certain change in the input until a certain output is produced under maximum disturbance from other parallel activities. These techniques not only have to set up a worst-case scenario (which is challenging in itself), but also have to measure the system non-intrusively, i.e., make sure that the measurement does not affect the test (probe-effects). It is often difficult to prove that removing test probes from the code has no side effects. A solution sometimes used in standard testing is to leave test probes in the code even after deployment. It is controversial since resource consumption should be kept at a minimum. Thus we form the next question considering this:

Q5: Can BIT be removed at deployment time without side effects?

Another distinguishing characteristic in many embedded systems is that they are resource constrained. They are often produced in large volumes, and it is economically beneficial to spend development-time on using techniques that uses a minimum of hardware resources (e.g., processor, network, and memory). The next question is related to these issues:

Q6: How much extra hardware resources are required for integrating BIT techniques?

Analysis, when applied, is often used in early phases; while testing in used in the later phases to verify that the system behaves according to analysis and specifications. Analysis could possibly be developed to identify the most critical points to be tested in order to create and integrate the proper Built-in mechanisms. The other way around is another interesting possibility, can BIT mechanisms be utilised to get a more precise models of actual systems properties?

BIT may have positive “side-effects” on software quality; one of these possibilities is assed in the following question:

Q7: Can testing concerns early in the development process, as enforced by BIT, affect positively on software quality?

Consideration of test issues early in the projects may increase quality, e.g., it leads to earlier detection of incomplete specifications. It would also be interesting to try to measure the impact of BIT on quality in other phases of development. Reuse is a basic concept in CBSE that promises to decrease development times. Reuse has still not been widely used, especially in the embedded systems segment; the reasons are many, but one of them may be the lack of test support in the reuse mechanism. Naturally, the first question on reuse and BIT is:

Q8: Does BIT facilitate component reuse in embedded component-based systems?

BIT may shorten the development-time through shorter verification time. This is perhaps only valid when tests or test results are reused, e.g., upon regression tests after maintenance or when a component is used in a new application. It might be a higher investment to develop a reusable BIT compared to standard tests, this investment must pay off when the component is touched next time. An interesting question here is:

Q9: To what extent is it possible to reuse the tests and test results themselves?

It is well-known that test-related activities constitute a large part of the total development time for software. Therefore reuse of test cases, and when possible even test results, is as important as reuse of the software itself. Otherwise the effect from reuse on the total development time will be minor. However in order to rely on reused test cases or test results, it must be ensured that the reused entity is compatible with the intended context for the component. Experience has shown that software reuse in new contexts is dangerous. It is mandatory that test-results and test suites that are possible to reuse are distinguished from those that must be changed.

From a pragmatic view it may be too difficult to achieve component reuse in industrial software projects, at least in a short perspective. There may be benefits in terms of shortening the total development-time even if components are not reused. Such benefits would originate from from increased quality and earlier detection of specification faults. Hence, the last question is formed:

Q10: Can testing concerns early in the development process, as enforced by BIT, shorten development-times?

The key research areas that we have defined based on the above questions can be summarized in the following list:

- Suitable BIT techniques for testing embedded systems
- Integration of BIT techniques that do not change system behaviour or raise complexity
- Resource aware BIT techniques for highly resource constrained systems
- Built-in analysis techinques, especially for timing properties
- Synergies between analysis and BIT
- Safe methods to enable reusing of tests and test results

Our expectation is that a careful integration of BIT techniques in the context of CBSE for embedded systems has high potential to improve the quality of the software and shorten the development-time, compared to traditional non-component based development and component-based development.

3 References

[1] I. Crnkovic and M. Larsson. *Building Reliable Component-Based Software Systems*. Artech House publisher, 2002. ISBN 1-58053-327-2.

[2] M. Åkerholm, I. Pavlova, and J. Fredriksson. Application of built-in-testing in component-based embedded systems. Technical report, Technical report, Malardalen university, May 2006. http://www.mrtc.mdh.se/publications/1125.pdf.

[3] C. E. F. Project, 2006. http://www.component-plus.org.

[4] W. Wolf. What is embedded computing? *IEEE Computer*, 35(1):136–137, January 2002.