Advanced Diagnosis: SBST and BIST Integration in Automotive E/E Architectures

Reimann, Felix; Glaß, Michael; Teich, Jürgen; Cook, Alejandro; Rodríguez Gómez, Laura; Ull, Dominik; Wunderlich, Hans-Joachim; Abelein, Ulrich; Engelke, Piet

Proceedings of the 51st ACM/IEEE Design Automation Conference (DAC'14) San Francisco, California, USA, 1-5 June 2014

doi: http://dx.doi.org/10.1145/2593069.2602971

Abstract: The constantly growing amount of semiconductors in automotive systems increases the number of possible defect mechanisms, and therefore raises also the effort to maintain a sufficient level of quality and reliability. A promising solution to this problem is the on-line application of structural tests in key components, typically ECUs. In this work, an approach for the optimized integration of both Software-Based Self-Tests (SBST) and Built-In Self-Tests (BIST) into E/E architectures is presented. The approach integrates the execution of the tests non-intrusively, i. e., it (a) does not affect functional applications and (b) does not require costly changes in the communication schedules or additional communication overhead. Via design space exploration, optimized implementations with respect to multiple conflicting objectives, i. e., monetary costs, safety, test quality, and required execution time are derived.

[1] **ACM COPYRIGHT NOTICE**
©2014 ACM. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Advanced Diagnosis: SBST and BIST Integration in Automotive E/E Architectures

Felix Reimann, Michael Glaß, Jürgen Teich
University of Erlangen-Nuremberg, Germany
{felix.reimann, glass, teich}@cs.fau.de

Alejandro Cook, Laura Rodríguez Gómez, Dominik Ull, Hans-Joachim Wunderlich
University of Stuttgart, Germany
{cook, rodrigla, ull, wu} @iti.uni-stuttgart.de

Ulrich Abelein
AUDI AG, Ingolstadt, Germany
ulrich.abelein@audi.de

Piet Engelke
Infineon Technologies AG, Neubiberg, Germany
piet.engelke@infineon.com

ABSTRACT

The constantly growing amount of semiconductors in automotive systems increases the number of possible defect mechanisms, and therefore raises also the effort to maintain a sufficient level of quality and reliability. A promising solution to this problem is the on-line application of structural tests in key components, typically ECUs. In this work, an approach for the optimized integration of both Software-Based Self-Tests (SBST) and Built-In Self-Tests (BIST) into E/E architectures is presented. The approach integrates the execution of the tests non-intrusively, i. e., it (a) does not affect functional applications and (b) does not require costly changes in the communication schedules or additional communication overhead. Via design space exploration, optimized implementations with respect to multiple conflicting objectives, i. e., monetary costs, safety, test quality, and required execution time are derived.

1. INTRODUCTION

Shrinking transistor sizes allow to enhance system performance at low cost. However, upcoming manufacturing technologies are more susceptible to variations, degradation, and aging effects. Besides semiconductor faults which were not revealed during manufacturing test (test escapes), latent hardware faults may become active in the field as a result of variability, harsh environmental conditions, or degradation. These faults pose a threat to system safety and quality.

A significant step towards safe and high-quality hardware is the integration of advanced test and diagnostic applications which are able to identify problems at the semiconductor level. Particularly for safety reasons, it is, however, extremely important to avoid unintended interactions between such diagnostic procedures and the system applications. In this realm, the work at hand tackles the problem of design time integration and optimization of advanced diagnostic features at the semiconductor level from a system-level perspective.

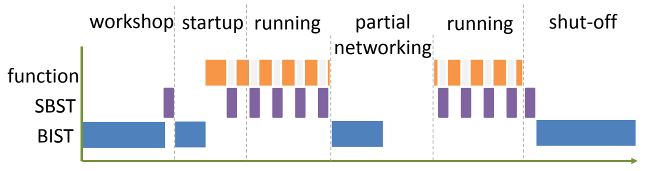

Figure 1: The applicability of advanced diagnostic features depends on the operational mode of the automotive system.

For this purpose, structural test and diagnostic methods are combined with two different goals in mind: (1) *Safety*: In order to guarantee safe operation of the vehicle, hardware defects have to be promptly detected, before they compromise the state of the system. Thus, structural tests need to be applied alongside standard applications at relatively high rate, enabling early fault detection. (2) *Quality*: To increase system quality, any systematic hardware issue in the assembly line has to be identified and corrected [1].

Broadly speaking, two major classes of structural tests lend themselves well to system test in the automotive domain, see Fig. 1: *Software-based Self Tests* (SBSTs) [2] can be carefully generated for a processor architecture and executed during regular system operation. Consequently, they may detect hardware faults shortly after they become visible. Conversely, scan-based *Built-In Self-Tests* (BIST) [3] offer a more general test methodology for any integrated circuit and typically deliver higher fault coverages than SBST. However, BIST application destroys the internal state of the device-under-test, which needs to be restored before resuming system operation, i. e., a reboot is required. Therefore, BIST can be applied only if the normal functionality is not required for a fixed time period.

The integration of these structural test techniques for system-level diagnosis can exploit several operational modes in a typical automotive system. For example, the *startup* of the system can be used for structural tests in order to enable any available countermeasure even before driving in case a safety critical component is found to be compromised. Similarly, BIST can be applied before system boot-up as no state has to be restored yet. However, as the driver shall not wait too long for the system to boot, the startup time has to be kept reasonable short. Also, the *shut-off* of the system allows to perform advanced diagnostic features but, as this consumes battery power, this phase has to be minimized as well. Similarly, BIST can be applied during *partial networking*, cf. AUTOSAR v4.0.3, to invoke BIST sessions before an electronic control unit (ECU) goes to power-down mode. Unfortunately, this reduces the energy-saving during power-down. Moreover, partial networking is not implemented by every ECU.

Additionally, SBST programs and BIST procedures can be generated for different performance goals in terms of fault coverage, runtime, and memory footprint. In combination with the different possible operational modes outlined above the optimal selection and scheduling of automotive diagnostic features poses a challenging question.

By exploring such a vast design space, our aim is to achieve a non-intrusive integration of advanced diagnosis features. We refer to the term non-intrusive here in the sense that the integration of advanced diagnosis capabilities must not affect system applications, particularly including their communication. This is a crucial aspect for the certification of the bus schedule as changing schedules for diagnosis execution during runtime is prohibited. In particular, the presented approach is not meant to only achieve a feasible integration, but to reveal the various trade-offs arising from different possible integrations with respect to multiple design objectives like monetary costs, safety, test quality, and shut-off time.

In the following section previous work is discussed. In Section 3, SBST and BIST are presented in detail. Based on their identified characteristics with respect to system integration, the design objectives are defined. Section 4 presents the integration of the introduced model and objectives into system-level design exploration. Section 5 contains an industrial case study, while Section 6 concludes the paper.

2. RELATED WORK

Structural tests are mandatory during manufacturing test in order to verify the hardware integrity of every produced chip. BIST is already a mature technique for manufacturing test, which can be easily reused in the field, as long as the required test access mechanisms are made available for system test.

[4, 5, 6] made BIST accessible in the field without the need for external test devices. Still, these approaches require predefined operational modes of the system which are exclusively devoted to self-test and maintenance, like power-up/down tests and workshop test in the case of the automotive domain.

The *online* non-concurrent structural test of automotive ASICs has been explored in [7]. However, this approach requires a special test architecture with significant additional costs. In [8], an on-chip infrastructure is presented, which is well suited for the online test of embedded memory cores, while [9] presents a self-test chip application for the application of structural tests. None of these approaches exploits any optimization potential during system level design.

SBST have also been proposed for online test as well: In [10, 11], the execution of SBST routines has been analyzed in the context of schedulability and error detection latency on a uniprocessor system.

Thus, SBST as well as BIST allow to tremendously increase diagnostic capabilities in the field, as they directly

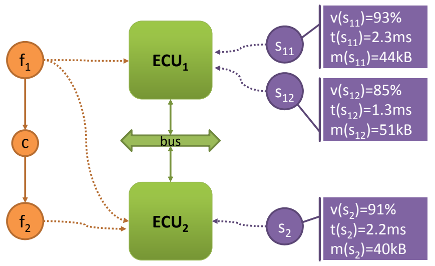

Figure 2: Besides the functional task t_1, which may also be mapped to ECU$_2$, two SBST tasks s_{11} and s_{21} are available for ECU$_1$. Both have different characteristics according to their fault coverage, execution time, and memory footprint. During system design, it has to be decided which SBST task s to select and schedule it by assigning a period.

identify an erroneous component. However, for both techniques, the question of their integration at system level remains to be answered. Here, the work at hand explores the integration of structural tests for digital logic during system-level design. Currently, only [12] investigates the integration of SBST into a system-level design methodology. However, the authors only consider SBSTs that are introduced intrusively in the sense that their data dependencies are realized by additional communication messages. In [1], an approach to integrate BIST is presented. Opposed to that, this work aims at non-intrusively integrating a set of SBSTs S and set of BISTs B by modeling them as a set of diagnosis tasks $D = S \cup B$ in addition to the functional tasks F. This leads to a holistic system model which considers all tasks $\Sigma = F \cup D$ in the system and, thus, allows to exploit the benefits of both techniques.

3. DESIGN FOR DIAGNOSABILITY

In this section, SBST and BIST are introduced in greater detail and also their characteristics regarding the diagnosability of the system is outlined. During the design of the E/E architecture, it is important to select for each component in the system the set of tests which lead to an optimal implementation with respect to diagnostic objectives as introduced in Section 3.3, namely test quality, safety, shut-off time, and monetary costs. In Section 4, these non-functional objectives are used for the design space exploration to reveal optimal implementations with respect to both functional as well as the above diagnostic objectives.

3.1 SBST Application

SBST programs are carefully generated sequences of assembly code, which exercise a large part of the underlying hardware implementation and are able to identify faults in the target processor. Different approaches for SBST rely on different methodologies like automatic test pattern generation (ATPG) [13], formal methods [14], evolutionary algorithms [15], etc. Although these techniques exploit functional properties of the processor, they activate and propagate faults at the structural (gate) level. For the integration of SBST programs, the following aspects have to be analyzed: (a) if it is possible to schedule SBST routines in the regular operation of the ECU and (b) how often the SBST program should be executed during normal operation. Note that these decisions heavily depend on the workload, i. e.,

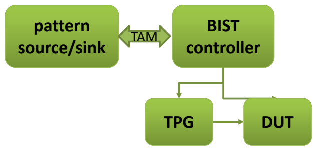

Figure 3: Simplified BIST infrastructure

the tasks mapped to an ECU.

Thus, each SBST task $s_r \in S \subset D$ available for integration on ECU r, see Fig. 2, has the following characteristics:

Coverage: Let the coverage $v(s_r)$ denote the achieved *stuck-at fault coverage*. Depending on the target processor, the coverage the aforementioned approaches achieve typically amounts to 80–95 % [2].

Runtime: An SBST is designed to deterministically exercise the structure of the processor and any performance features, like associative caches, branch prediction, forwarding, speculative or superscalar execution are carefully controlled. Thus, the *execution time* $t(s_r)$ of the SBST can be measured accurately. This also includes the initial processor setup phase, which is required before the actual test program execution. In this phase, the processor is set to a known state from which the effectiveness of an SBST partition is guaranteed. Typical examples for achieving a suitable initial state are cache invalidation, pipeline flushing and register initialization.

Size: The *memory footprint* $m(s_r)$ of the SBST application can be evaluated directly from its assembly code size.

3.2 BIST Application

To apply BIST patterns, a chip has to enter a special test mode in which it does not adhere to its functional specification. Moreover, the application of a BIST session arbitrarily changes the state of the chip, which has to be restored to a known state before the enclosing ECU can perform any useful work again.

The BIST strategy considered in this work is *mixed-mode BIST* [16]. The test pattern generation process of a mixed-mode BIST session comprises two steps: First, a pseudo-random test pattern generator produces a large number of inexpensive test patterns on-the-fly. For the hard-to-detect faults, additional deterministic test patterns are encoded and stored in the system. Second, the deterministic patterns are decoded and applied to the *design under test* (DUT).

For the application of BIST, the on-chip test infrastructure is made available to an on-chip controller. This controller manages the application of pseudo-random patterns, fetches the encoded test data from memory, and applies the deterministic patterns.

Figure 3 shows a simplified block diagram for the proposed mixed-mode BIST approach. Before any BIST session, the pseudo-random *test pattern generator* (TPG) is initialized with a known seed so that it produces a fixed sequence of patterns. Memory is used to store the encoded deterministic patterns while a BIST controller is employed to manage the test application process, i. e., to generate a given number of pseudo-random patterns and apply them to the DUT. As the figure shows, the encoded deterministic pattern information is accessed by means of a *test access mechanism* (TAM). In a typical BIST session, the encoded deterministic patterns are stored in on-chip ROM and the TAM is a simple memory interface. In order to explore all implementation options, the

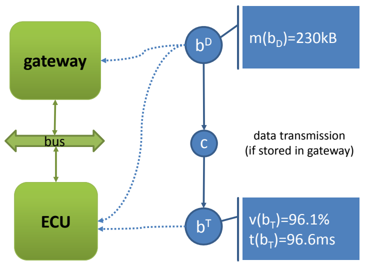

Figure 4: The BIST data task b^D can be mapped to the gateway or to the same resource as the corresponding BIST test task b^T.

architecture supports the storage of encoded deterministic patterns in any persistent memory inside the vehicle. In this case, any functional bus (e. g., CAN) can be used as TAM and any on-chip memory can be used to buffer the required data for test application. Consequently, if the encoded test data is stored externally, the corresponding TAM must also be made available to the test controller. This is the only nonstandard feature of the considered BIST architecture.

Similarly, test responses have to be compacted on-chip and stored either locally or on a remote non-volative memory in the vehicle. However, the test responses necessary for diagnosis can be drastically compacted without any significant impact on the diagnostic accuracy [17]. During system design, it has then to be decided (a) if it is beneficial to use a possibly more costly ECU with BIST support, (b) which BIST program to select (their achieved fault coverage, test time and cost depends on the ratio of pseudo-random and deterministic test patterns), and (c) where to store the deterministic test patterns.

Accordingly, a BIST program available for integration on an ECU r is modeled as follows, see Fig. 4: (1) a BIST task $b_r^T \in B \subset D$ models the BIST itself, (2) the data storage task $b_r^D \in D$ models where the encoded deterministic patterns are stored and can be mapped either to the ECU r or to a central component like the gateway, and (3) the message $c \in C$ with which the both tasks communicate. A BIST task has the following characteristics:

Coverage: Let the fault coverage $v(b_r^T)$ denote the achieved stuck-at fault coverage. It can be estimated by means of fault simulation.

Runtime: The execution time $t(b_r^T)$ of the BIST program depends on the number of test patterns, the performance of the TAM and the duration of the state restore procedure after test. If the BIST session is applied during ECU shut-off, the restore procedure takes place during the next power-up sequence and, therefore, is not taken into account. Alternatively, for partial networking the ECU state restore is achieved by means of an ECU reset, for which timing models are usually available.

Size: The size of the encoded data for deterministic pattern generation is denoted by $m(b_r^D)$.

3.3 Diagnosis-related Design Objectives

Design objectives determine the optimization goals for the system implementation. We introduce three new objectives to reflect the test capabilities of the system, which, together with a common cost objective, drive the optimization efforts:

Test Quality: The *test quality* is defined as the average stuck-at fault coverage achieved for all the ICs in the

ECUs of a given implementation. In case of BIST, the test quality is also a measure of the system's diagnostic capabilities. The implicit assumption is that any detected fault can be correctly diagnosed. This assumption is reasonable since the success rate of the underlying diagnostic algorithm is sufficiently high [18].

In contrast to the safety goal, the *test quality* Q is independent of time. If more than one diagnostic task is mapped to an ECU, the compound fault coverage may be higher as that of the individual tasks. However, test procedures are usually developed independently. The work at hand uses a pessimistic approach by counting only the maximum fault coverage of these tasks:

$$Q = \max \sum_{r \in R} \max_{\substack{d \in D \\ (d,r) \in M}} v(d), \tag{1}$$

where $(d, r) \in M$ denotes a selected mapping of diagnosis task d to resource r.

Safety: A means to support safety are fault detection mechanisms. For each application, a safety goal is defined, which quantifies the maximum fault-detection latency for the application. If a fault in the underlying ECU's hardware is detected within the safety goal, the fault can be mitigated and system safety is not compromised. The optimal implementation with respect to safety is achieved if the safety goal of each of its applications is satisfied.

Given a *safety goal* $g(f)$ for each functional task $f \in \Sigma \setminus D$ (or whole applications), if an SBST task s is mapped to the same resource r and the explored period $p(s) \leq g(f)$, a hardware fault could be detected in time. In this case, typical countermeasures like degradation to a safe state can be performed. Thus, the objective *safety* Θ to be maximized counts the number of tasks with fulfilled safety goal:

$$\Theta = \max |\{\sigma \in \Sigma \setminus D| \quad \forall s \in S: \quad (s,r),(\sigma,r) \in M \quad \wedge \quad p(s) \leq g(\sigma)\}| \tag{2}$$

Note that only SBSTs are considered here as BISTs can only applied sporadically.

Shut-off Time: The shut-off time is defined as the maximum amount of extra time an ECU has to stay active in order to complete its BIST session. As the ECU would otherwise be powered-down, shut-off time has to be kept reasonably low. This makes it possible to apply BIST during partial networking and ensures a fast shut-off of the vehicle after driving. If the patterns of a BIST session are stored locally in the ECU, the session can be executed as soon as the ECU is no longer actively used. Otherwise, the corresponding patterns have to be transmitted first, which requires $t(b_r^D)$, see Section 4.1. Therefore, the objective *shut-off time* O is given as time the system has to stay awake longer to run BISTs:

$$O = \min \max_{b_r^T \in B} \begin{cases} t(b_r^T) & \text{if } (b_r^D, r) \in M \\ t(b_r^T) + t(b_r^D) & \text{if } (b_r^D, r') \in M, r \neq r' \\ 0 & \text{else.} \end{cases} \tag{3}$$

Hardware Costs: SBST programs are stored locally in each ECU, while encoded BIST patterns can be stored anywhere in the system and transmitted over a field bus. The local storage of encoded BIST patterns requires corresponding hardware modifications in each of the target ECUs. A less intrusive approach is to store the encoded information in a central gateway, see Fig. 4. This can also save memory costs if the same encoded patterns can be used for different ECUs. The memory costs are added to the objective *monetary cost* Φ, which considers also the costs of the allocated hardware.

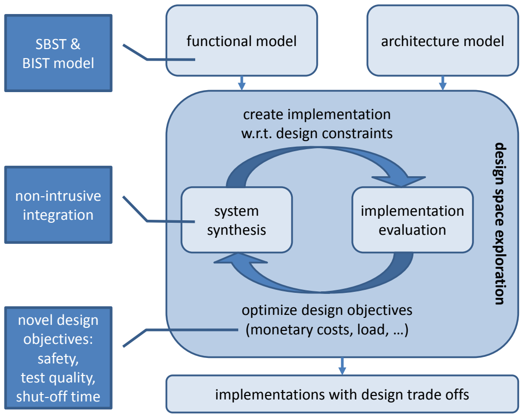
SBST & BIST model
functional model
architecture model
create implementation w.r.t. design constraints
non-intrusive integration
system synthesis
implementation evaluation
design space exploration
novel design objectives: safety, test quality, shut-off time
optimize design objectives (monetary costs, load, …)
implementations with design trade offs

Figure 5: Holistic design approach for creating inherently diagnosis-capable E/E architectures. The functional model containing the application tasks F is augmented with the available BIST tasks B and SBST tasks S supported by the architecture. System synthesis creates implementations which respect common design constraints as well as diagnosis-specific constraints to integrate diagnosis non-intrusively. The found implementations are evaluated according to classic design objectives as well as additional diagnostic and safety-related design goals.

4. INTEGRATION OF DIAGNOSIS IN THE DESIGN SPACE EXPLORATION

To achieve a high-quality optimization able to consider several objectives in parallel, many feasible implementations have to be created and evaluated with respect to multiple design objectives, see [19]. A *feasible* implementation, however, has to fulfill several constraints, like binding all functional tasks to resources, routing the messages in the system accordingly, etc. For the encoding of the allocation of resource, the mapping of diagnosis and mandatory tasks, as well as for the routing of messages, we apply the approach presented in [1]. This technique allows the optimization of the overall system considering several distinct objectives.

For the encoding of the periods of the SBST task activations, a lower bound l_s and an upper bound $u_s, u_s \geq l_s$, for the period of the execution of an SBST task s shall be given, as well as a step size Δ_s. Thus, the period search space P for the SBST tasks S is given as

$$P = \prod_{s \in S} \{0, \ldots, \lfloor \frac{u_s - l_s}{\Delta_s} \rfloor\}. \tag{4}$$

Now, this search space is encoded with an integer genotype and combined with the remaining search space from [1] using composite genotypes, see [20].

4.1 Non-intrusive Integration of Diagnostic Applications

If the BIST data is not stored locally, cf. Fig. 6, the deterministic test patterns have to be transmitted over the

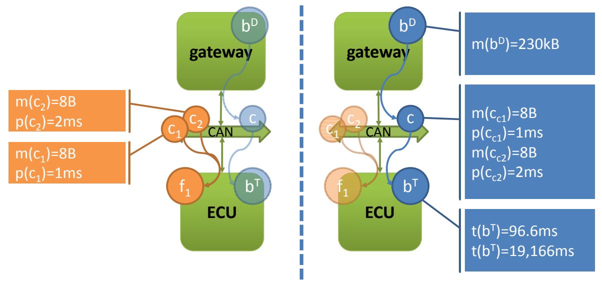

Figure 6: The message c is only active, if the functional task t_1 as well as its messages c_1 and c_2 are inactive. By mirroring their sending patterns, the test patterns can be transmitted transparently for other subscribers.

system bus. However, sending the patterns in a large burst (even with lowest priority) would obviously affect the timing of other messages. As BIST can only be applied to inactive ECUs, free bandwidth resources on the bus which are reserved for the ECU's functional operation, can be reused for test-data transfers. For this reason, we do not change an originally certified bus schedule when conducting a BIST session. As shown in Fig. 6, the sending pattern of the ECU's functional messages (e.g. c) is mirrored when transmitting test data messages (e.g. c'). Message c' receives a different CAN-ID than c, but keeps the same timing properties on the bus – size, period, and relative priority – as c, while staying distinguishable. As the size of the test patterns is typically large (see Chapter 5), the described technique is exerted on all messages during this test session. In the following, the work at hand concentrates on CAN busses, while the described concept is extensible to other automotive field busses. For a BIST test b_r^T, the time $t(b_r^D)$ is required to transmit the test patterns and, therefore, solely depends on the bandwidth of the functional messages I of ECU r:

$$t(b_r^D) = \frac{m(b_r^D)}{\sum_{c \in I} \frac{m(c)}{p(c)}} \tag{5}$$

5. CASE STUDY

The following case study models an automotive subnet. Four control-centric applications with 45 tasks and 41 messages have to be implemented. For the architecture, 15 ECUs, 9 sensors, and 5 actuators connected with three distinct CAN buses are available. For the ECUs in this case study, both BIST and SBST applications have been developed. The target DUT in both cases is an automotive microprocessor from Infineon Technologies AG. In the next subsections, the properties of the diagnostic applications are detailed.

5.1 BIST Profiles

To generate a BIST application, the necessary on-chip infrastructure was inserted into the design (full-scan). The characteristics of the final DUT are: 371,900 collapsed faults, 100 scan chains with a maximum length of 77, and a test frequency of 40 MHz. An automatic test generation approach produced 36 test profiles, see [1]. Each test profile exhibits different performance characteristics in terms of fault coverage (95.1–99.9 %), test execution time (1.7–965 ms) and costs (154–971 kB). For each of the 15 ECUs, one of these BIST programs can be selected.

5.2 SBST Profiles

An SBST program was developed for the integer pipeline of the target processor. The methodology for test program generation is based on constrained ATPG [13]. A program consisting of all supported instructions was then simulated and the control signals of the target pipeline corresponding to each instruction were identified. This information was then used to constrain sequential ATPG with a commercial tool. After binary ATPG-patterns are created, they are parsed to generate a test program. This can be performed since the location of the instruction opcode in the pattern and its encoding are known. These constraints guarantee that the generated test pattern is a valid sequence of instructions and the faulty test responses are propagated to the register file. The processor can then take appropriate action.

The characteristics of the generated test program are: 69,890 target faults, a fault coverage of 91.54 %, 62,771 executed instructions, 44.7 kB of required program memory, and a test execution time of 1.587 ms at a functional clock frequency of 100 MHz. The given test duration refers to an execution without interruption. For this calculation, a setup overhead of 1 ms is considered. An instance of this SBST task can be mapped to each of the 15 ECUs.

5.3 Results

The optimization was performed on an 8-core Intel Core i7 with 16 GB RAM. Evaluating 25,000 implementations took 59 minutes, while implementations with overloaded resources were discarded. 322 not Pareto-dominated implementations according to all objectives (monetary costs, test quality, shut-off time, and safety) were found. Fig. 7 bottom shows that some of these implementations (marked with ▲) require a very long shut-off time of more than 200 seconds. However, these are the implementations which in contrast have a high fault coverage with only a minor increase in monetary costs, cf. Fig. 7 top, as their deterministic test patterns are stored centrally at the gateway. As the cost impact of the gateway memory is relatively low compared to the costs of the whole allocated hardware, a quite good test quality can be achieved with only a little increase in overall costs.

Both outlined sets of implementations have tradeoffs with zero up to three applications, for which the safety goals of their corresponding tasks are fulfilled. The implementations for which the safety goals of all three applications are fulfilled are marked with ◇. The results show that the safety goal is an orthogonal objective compared to quality. This means, adding SBST tasks has only a small impact on system cost. For one application, the available SBST could not be integrated fittingly by our methodology and, hence, the safety goal is not met. To overcome this issue, the SBST program generation could be optimized for execution time and included in the model. All in all the approach is successful in providing optimized tradeoffs between diagnostic and classic design objectives.

For the decision making, these trade-offs provide all implementations to find the sweet spot according to the designer's needs. For example, the maximum shut-off time could be constrained to 5 seconds and a met safety goal for three applications is required. In this case, the set of high-quality implementations contains eight candidates with relatively high monetary costs of about 1200 to 1500 and a test qual-

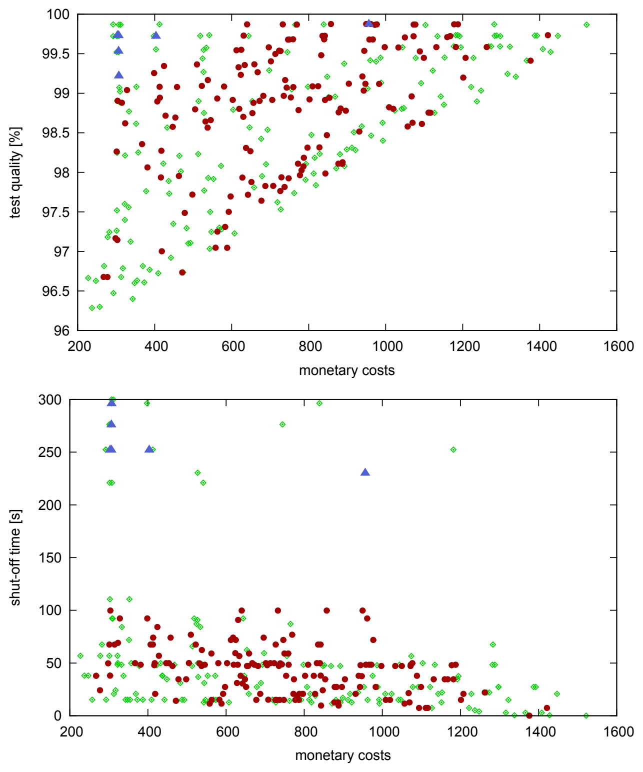

Figure 7: 322 implementations showing the tradeoffs between monetary costs Φ versus test quality Q (top) and monetary costs Φ versus shut-off time O (bottom). Implementations with three of four applications with fulfilled safety goal are marked with ◇, remaining implementations with shut-off time less than 200 seconds are marked with ●, implementations with higher shut-off time are marked with ▲.

ity of about 98.7 % to 99.8 %. Relaxing the shut-off time to a maximum of 30 seconds offers a particularly low-price implementation with monetary costs of 271, a test quality of 96.7 %, but a shut-off time of about 27 seconds.

6. CONCLUSION

The work at hand incorporates both BIST and SBST in the design of automotive E/E architectures as means to improve system quality and support safety. SBST programs are used to improve fault detection latency, while BIST sessions improve the diagnostic capabilities of the system during failure analysis. The several implementation choices comprise different tradeoffs in terms of hardware costs, test quality, and safety. All design options are integrated in a holistic design space exploration to provide high-quality tradeoffs, out of which system designers can select their preferred solution.

7. REFERENCES

[1] U. Abelein, A. Cook, P. Engelke, M. Glaß, F. Reimann, L. Rodríguez Gómez, T. Russ, J. Teich, D. Ull, and H.-J. Wunderlich, "Non-Intrusive Integration of Advanced Diagnosis Features in Automotive E/E-Architectures," in *Proc. of Design, Automation and Test in Europe Conference & Exhibition (to appear)*, 2014.

[2] M. Psarakis, D. Gizopoulos, E. Sanchez, and M. Reorda, "Microprocessor Software-Based Self-Testing," *IEEE Design & Test of Computers*, vol. 27, pp. 4–19, 2010.

[3] H.-J. Wunderlich, "BIST for systems-on-a-chip." *Integration, the VLSI Journal*, vol. 26, pp. 55–78, 1998.

[4] T. Vo, Z. Wang, T. Eaton, P. Ghosh, H. Li, Y. Lee, W. Wang, H. Jun, R. Fang, D. Singletary, and X. Gu, "Design for Board and System Level Structural Test and Diagnosis," in *Proc. of IEEE International Test Conference (ITC'06)*. IEEE, 2006, pp. 1–10.

[5] J. Qian, X. Wang, Q. Yang, F. Zhuang, J. Jia, X. Li, Y. Zuo, J. Mekkoth, J. Liu, H.-J. Chao, S. Wu, H. Yang, L. Yu, F. Zhao, and L.-T. Wang, "Logic BIST Architecture for System-Level Test and Diagnosis," in *Proc. of Asian Test Symposium (ATS'09)*, 2009, pp. 21–26.

[6] A. Cook, D. Ull, M. Elm, H.-J. Wunderlich, H. Randoll, and S. Döhren, "Reuse of Structural Volume Test Methods for In-System Testing of Automotive ASICs," in *Proc. Asian Test Symposium (ATS'12)*, 2012, pp. 214–219.

[7] A. Dutta, M. Shah, G. Swathi, and R. Parekhji, "Design Techniques and Tradeoffs in Implementing Non-Destructive Field Test using Logic BIST Self-Test," in *Proc. of International On-Line Testing Symposium (IOLTS'09)*, 2009, pp. 237–242.

[8] P. Bernardi and M. S. Reorda, "A new Architecture to Cross-Fertilize On-line and Manufacturing Testing," in *Proc. of Asian Asian Test Symposium (ATS'11)*, 2011, pp. 142–147.

[9] Y. Li, S. Makar, and S. Mitra, "CASP: Concurrent Autonomous Chip Self-Test Using Stored Test Patterns," in *Proc. of Design, Automation and Test in Europe, Conference & Exhibition (DATE'08)*, 2008, pp. 885–890.

[10] A. Paschalis and D. Gizopoulos, "Effective Software-Based Self-Test Strategies for On-line Periodic Testing of Embedded Processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, pp. 88–99, 2005.

[11] D. Gizopoulos, "Online Periodic Self-Test Scheduling for Real-Time Processor-Based Systems Dependability Enhancement," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, pp. 152–158, 2009.

[12] M. Eberl, M. Glaß, J. Teich, and U. Abelein, "Considering Diagnosis Functionality during Automatic System-Level Design of Automotive Networks," in *Proc. of Design Automation Conference (DAC'12)*, 2012, pp. 205–213.

[13] L. Chen, S. Ravi, A. Raghunathan, and S. Dey, "A Scalable Software-Based Self-Test Methodology for Programmable Processors," in *Proc. of Design Automation Conference (DAC'03)*, 2003, pp. 548–553.

[14] S. Gurumurthy, S. Vasudevan, and J. A. Abraham, "Automatic Generation of Instruction Sequences Targeting Hard-to-Detect Structural Faults in a Processor," in *Proc. of International Test Conference (ITC'06)*, 2006, pp. 1–9.

[15] F. Corno, E. Sanchez, M. Reorda, and G. Squillero, "Automatic Test Program Generation: a Case Study," *IEEE Design & Test of Computers*, vol. 21, pp. 102–109, 2004.

[16] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois, "Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers," in *IEEE Trans. on Computers (TC)*, vol. 44, 1995, pp. 223–233.

[17] A. Cook, M. Elm, H. Wunderlich, and U. Abelein, "Structural In-Field Diagnosis for Random Logic Circuits," in *Proc. of European Test Symposium (ETS'11)*.

[18] A. Cook, S. Hellebrand, and H.-J. Wunderlich, "Built-in Self-Diagnosis Exploiting Strong Diagnostic Windows in Mixed-Mode Test," in *Proc. European Test Symposium (ETS'12)*, 2012, pp. 1–6.

[19] J. Teich, "Hardware/Software Co-Design: Past, Present, and Predicting the Future," *Proc. of the IEEE*, vol. 100, pp. 1411–1430, 2012.

[20] M. Lukasiewycz, M. Glaß, F. Reimann, and J. Teich, "Opt4J – A Modular Framework for Meta-heuristic Optimization," in *Proc. of GECCO*, 2011, pp. 1723–1730.