Non-Intrusive Integration of Advanced Diagnosis Features in Automotive E/E-Architectures

[1]Ulrich Abelein, [2]Alejandro Cook, [3]Piet Engelke, [4]Michael Glaß, [4]Felix Reimann
[2]Laura Rodríguez Gómez, [4]Thomas Russ, [4]Jürgen Teich, [2]Dominik Ull, [2]Hans-Joachim Wunderlich
[1]AUDI AG, Ingolstadt, Germany. Email: ulrich.abelein@audi.de
[2]University of Stuttgart, Germany. Email: {cookao, rodrigla, ulldk, wu}@iti.uni-stuttgart.de
[3]Infineon Technologies AG, Neubiberg, Germany. Email: Piet.Engelke@infineon.com
[4]University of Erlangen-Nuremberg, Germany. Email: {glass, felix.reimann, teich}@cs.fau.de

***Abstract*—With ever more complex automotive systems, the current approach of using functional tests to locate faulty components results in very long analysis procedures and poor diagnostic accuracy. Built-In Self-Test (BIST) offers a promising alternative to collect structural diagnostic information during E/E-architecture test. However, as the automotive industry is quite cost-driven, structural diagnosis shall not deteriorate traditional design objectives. With this goal in mind, the work at hand proposes a design space exploration to integrate structural diagnostic capabilities into an E/E-architecture design. The proposed integration is performed non-intrusively, i. e., the addition and execution of tests (a) does not affect any functional applications and (b) does not require any costly changes in the communication schedules.**

I. Introduction

Newer generations of automotive systems integrate an impressive amount of electronics in order to realize very complex features [1]. As semiconductors play a central role in the automotive industry, very tight quality standards are usually enforced during chip manufacturing. However, the integrity of an automotive E/E architecture can still be affected by test escapes, latent hardware faults, which become active after system assembly, harsh environmental conditions, or degradation.

The automotive industry traditionally relies on *functional tests* to detect any hardware issue in the field. This kind of tests verifies the integrity of the system by comparing the system's behavior to its specification. This is achieved, for example, by means of concurrent test routines and power-up tests, that evaluate functional system properties, e.g., plausibility checks or field bus interconnection checks. Unfortunately, these techniques may not suffice to account for complex semiconductor failure behavior, since they do not fully exercise the complete system functionality and typically only cover a few corner-cases in the specification.

For instance, in a distributed automotive subsystem consisting of several interacting electronic control units (ECU), functional tests may help to disclose many system malfunctions. However, their structural fault coverage is hard to measure, and roughly amounts to 47% [2]. As a consequence, functional tests have to be complemented by *structural tests* to meet the high quality demands of the automotive domain.

Structural tests identify faulty components in a device according to its internal building blocks and their connectivity. For example, they may check for unintended connections, or *bridges*, between neighboring signals in the circuit layout. In contrast to functional tests, which can only provide a pass/fail test outcome, structural tests offer the opportunity to distinguish fault locations. This structural diagnostic capability is extremely useful for semiconductor quality improvement in the automotive industry.

978-3-9815370-2-4/DATE14/©2014 EDAA

The goal of this paper is to architect the efficient periodic application of structural diagnostic tests of the E/E architecture of a vehicle and to enable the collection of diagnostic data for subsequent detailed semiconductor diagnosis. The proposed methodology serves two different purposes:

- *Workshop repair*. Structural tests directly identify the faulty ECU to be replaced in the workshop. Therefore, repair and service costs can be greatly reduced, as fault-free units are only rarely discarded.
- *Failure analysis*. This process is performed for all field returns once a faulty ECU is identified. It encompasses all activities to identify the root-cause of the failure and it spans diagnostic measures taken by the original equipment manufacturer (OEM), tier 1 and 2 suppliers as well as semiconductor manufacturers. Structural diagnostic information helps the OEM and tier 1 and 2 suppliers to identify a faulty chip and accelerate system and board diagnosis. For the chip manufacturer, the collected diagnostic information can be used to understand systematic problems in the fabrication process and take adequate measures during chip production and manufacturing test in order to improve chip quality.

The periodic application of autonomous structural tests during regular system operation allows chip testing under very similar conditions under which an error occurs. This helps to capture the effects of a hardware problem without having to remove a chip from its enclosing system. Consequently, the "No Trouble Found" problem [3], in which a faulty chip does not produce any errors when tested in isolation, can be drastically reduced and failure analysis can be performed much more efficiently.

In this work, we make use of available Built-In Self-Test (BIST) [4] capabilities and require only minor architectural modifications. We propose to invoke BIST during operational shut-off of the vehicle's ECUs, in which all applications are inactive. Similarly, the same approach can be applied during *partial networking*, cf. AUTOSAR v4.0.3, to invoke BIST sessions before the ECU goes to power-down mode. The time interval between operational shut-off and real power-down is

kept reasonably short by introducing the *shut-off time* as an additional diagnostic design objective besides *test quality*.

Based on these assumptions, the work at hand performs a *non-intrusive* integration of BIST into existing and future automotive architectures. We refer to the term non-intrusive here in the sense that the integration of structural diagnostic capabilities shall not affect system applications, particularly including their communication. This is a crucial aspect for the certification of bus schedules, as changing schedules for diagnosis execution during runtime is prohibited. To achieve this, we use a holistic system model that captures system-wide properties that arise from networked ECUs, i. e., bus schedules as well as intra-ECU properties like local schedules, available memory, etc. By exploring the design space, we aim not only to achieve a feasible integration, but also to reveal achievable tradeoffs with respect to multiple design objectives, i.e., a) test quality, b) shut-off time, and c) monetary costs.

In the following section, related work is discussed. In Section III, a model suitable to reflect the target system as well as its diagnostic applications is introduced. Also, the integration of the introduced model and objectives into system-level design space exploration is presented. Section IV contains an industrial case study, while Section V concludes the paper.

II. Related Work

Logic built-in self-test (BIST) [4] integrates *design-for-test* resources for autonomous test pattern generation and on-chip response compaction.

The most widely used BIST architecture is represented in Fig. 1, commonly referred to as the *STUMPS* architecture (Self-Testing Unit Using MISR and Sequence Generator) [5]. This architecture has been widely used to support system-level tests [6], [7], [8].

The basic building blocks of this architecture are a *test pattern generator* (TPG), the *circuit under test* (CUT) and a *Test Response Evaluator* (TRE). The TPG produces pseudo-random test patterns to be applied to the CUT. As most designs cannot be sufficiently tested only with random patterns, deterministic patterns can also be encoded and stored on-chip, which are reconstructed during test application. The use of both random and encoded deterministic patterns is known as *mixed-mode BIST*.

The TRE usually compacts the whole test application into a single test *signature*, which is later compared to the expected test result. However, a single signature does not provide enough information to enable semiconductor diagnosis. Recently, the STUMPS architecture has been extended in order to enhance its diagnostic capabilities [9] and efficient logic diagnosis algorithms have emerged that require only a few test signatures [10]. The *response* and *fail data* in Fig. 1 represent the architectural extensions for the collection of intermediate diagnostic signatures during test application.

The obtained intermediate signatures are compared to the expected *response data*. If they differ, the obtained signature is included in the *fail data*, together with a signature index to identify the faulty signature in the test sequence. A BIST controller manages the generation of intermediate signatures by asserting relevant control signals for the TPG, TRE, scan chains in the CUT and embedded memories. The application of a BIST session requires that a chip enters a special test mode in which the chip does not conform to its functional

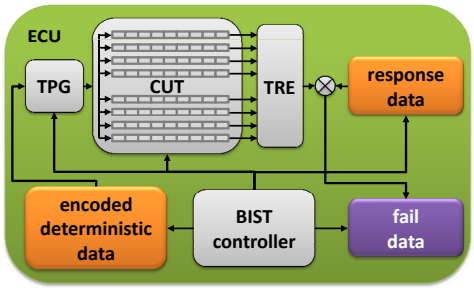

Figure 1. STUMPS architecture. Deterministic as well as randomly generated test patterns are applied to the scan chains of the *circuit under test* (CUT). The outcome is compared to the expected responses, storing the resulting differences in the fail memory for further treatment.

specification. Moreover, test application arbitrarily changes the state of the chip, which has to be restored to a known state before the enclosing ECU can make use of the chip to perform any functional operation.

In [11], [12], [13], the integration of built-in structural tests into regular system operation takes advantage of fixed idle slots, where the CUT does not realize any functional task. However, the use of a predefined and independent test schedule for each component may not provide an optimal test strategy for complex systems. More recently, the work in [14] considers the integration of software-based self-tests into the design of automotive networks. As explained in the next section, BIST integration is more elaborate, as some system components are no longer in functional mode and diagnostic tasks may require the exchange of diagnostic data.

III. Modeling and Integration of Diagnostic Applications

Contrary to traditional functional tests, the proposed collection of global diagnostic knowledge has a stronger influence on the behavior of the system. Additional system properties have to be considered to enable some chips to go into test mode for BIST application, while others remain in functional mode.

In order to fully exploit the system's optimization potential, we make use of the diagnostic architecture presented in Fig. 1. The *encoded deterministic test data* and the *response data* are stored together either locally or in a centralized memory in the system. In the latter case, an available field bus like, for example CAN, is used as *test access mechanism* (TAM) to transfer the necessary test information.

The performance of a BIST session b for a given CUT can then be characterized as follows:

- The *fault coverage* $c(b)$ is the achieved stuck-at fault coverage [15] and can be estimated by means of fault simulation. The fault coverage is a measure of the achievable diagnostic capabilities. However, the underlying logic diagnosis algorithm is not limited to this fault model.
- The *runtime* $l(b)$ of the BIST session depends on the number of applied test patterns, the performance of the TAM and the duration of the state restore procedure after test.
- The *memory size* $s(b)$ of the encoded data for deterministic pattern generation.

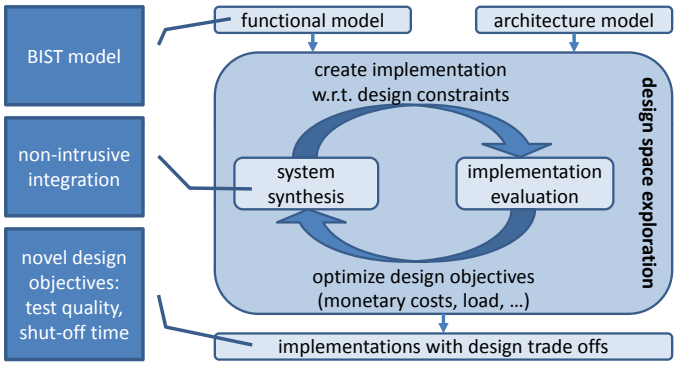

Figure 2. E/E-architecture design approach for creating diagnosis-capable systems. The functional model containing the applications is augmented with the available BIST tasks supported by the CUT. The system synthesis creates implementations which respect traditional design constraints as well as diagnosis-specific constraints for the non-intrusive integration of diagnostic features. The obtained implementation is evaluated according to classic design objectives as well as diagnostic design goals.

By changing the number of random and deterministic test patterns comprising the mixed-mode BIST session, many different tradeoffs with respect to these properties need to be considered.

After completion of a BIST session, the fail data is transmitted to a central gateway, where it is stored. Contrary to functional tests, for which a relatively large number of diagnostic trouble codes (DTC) are stored locally in each ECU [16], the collected fail data amounts to only a few bytes of information. As the fail data provides the test outcome for each integrated circuit (IC) of a given ECU and, thus, the definitive location of a fault, it is advantageous to make all fail information available to a central controller in order to coordinate any available error avoidance countermeasure at system-level.

Thus, to augment existing ECUs with extended diagnostic capabilities and not deteriorate traditional design objectives, a holistic design flow for the integration of BIST in automotive E/E-architectures becomes mandatory.

The proposed design flow is presented in this section and outlined in Fig. 2. First, an extended system model that represents the traditional E/E-architecture and applications together with the newly introduced diagnostic capabilities is detailed in Section III-A. Then, in Section III-B, design considerations are presented for the non-intrusive system integration of BIST. Here, the test application shall not affect the timing properties of the functional applications, since any variation in the system's functional behavior becomes a critical aspect for its certification. The model and test integration requirements define the search space for the optimization presented in Section III-C, which is able to consider simultaneously the three design objectives of test quality, shutoff time, and monetary costs (Section III-D).

A. Holistic System Model

Due to the characteristics of BIST the system design process has to evaluate in addition to traditional design options, (a) if it is beneficial to use a possibly more costly ECU with BIST support, (b) which BIST program to use, and (c) where to store the deterministic test patterns.

Therefore, we employ the general graph-based specification $g_S(g_T, g_A, M)$ of a design space exploration problem

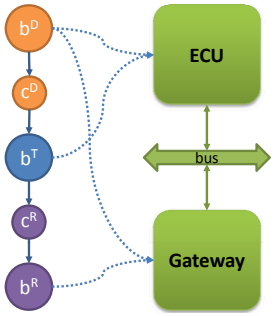

Figure 3. The BIST data task b^D models the permanent memory required to store the *encoded deterministic test data* and the *response data* for testing the shown ECU. It can be either mapped to the gateway or to the same resource as the corresponding BIST test task b^T. The resulting *fail data* is transmitted to the gateway where task b^R gathers the reported failures of all ECUs.

as proposed by [17], which includes all design options of both functional and, as we will show, diagnostic parts of the system. The functional model is a bipartite *application graph* $g_T = (T \cup C, E_T)$, see Fig. 3 left. The set of tasks T includes the set of functional tasks $F \subset T$ and diagnostic tasks $D \subset T$, both represented by vertices $t \in T$. Each data dependency between tasks is modeled by a vertex $c \in C$. Edges $e \in E_T \subseteq (T \times C) \cup (C \times T)$ model their dependencies. The set of available resources is described by the *architecture graph* $g_A = (R, E_A)$, see Fig. 3 right. Vertices $r \in R$ model resources like processors, memories, or buses. Edges $E_A \subseteq R \times R$ model connections between these. Beside the application graph g_T and the architecture graph g_A, a set of *mapping edges* $M \subseteq T \times R$ is given. Each mapping edge $m = (t, r) \in M$ indicates a possible mapping of a computational task t onto a resource r.

An *implementation* $x = (A, B, W) \in X$ is one possible solution of the design space exploration problem. The allocation A is the subset of resources used in the implementation. The binding B is the subset of mappings that assign tasks to resources. For each communication c, the routing W contains a subset of resources W_c over which c is routed.

On this basis, a BIST program available for integration on ECU r is modeled as follows, see Fig. 3: (1) a BIST task $b_r^T \in B \subset D$ for the test application, (2) a data storage task $b_r^D \in D$ which contains the encoded deterministic test data and the response data and can be mapped either to the ECU r or to a central component like a gateway, (3) the mandatory collection task $b^R \in F$ that stores the fail data of each BIST session, and finally (4) the messages $c^D, c^R \in C$ over which BIST tasks communicate.

B. Non-intrusive Integration of Diagnostic Applications

If the BIST data is not stored locally, cf. Fig. 4, the deterministic test patterns have to be transmitted over a system bus. However, sending the patterns in a large burst (even with lowest priority) could affect the timing of functional messages of other resources on the same bus. Since the functional applications of the CUT are inactive when the BIST is applied, the resulting free bandwidth resources on the bus can be reused for test-data transfers. Consequently, we are able to retain the originally certified bus schedule when conducting a BIST session.

As shown in Fig. 4, the communication properties of the ECU's functional messages c_1 and c_2 are mirrored when

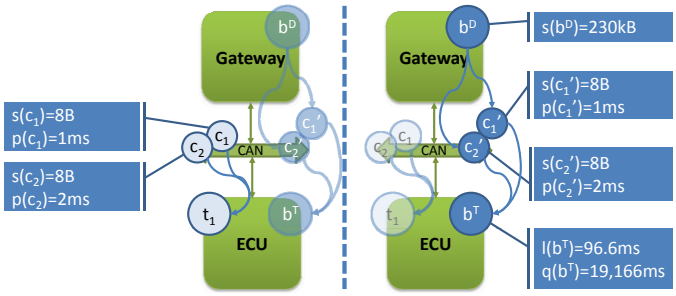

Figure 4. The messages c_1' and c_2' are only active, if the functional task t_1 as well as its messages c_1 and c_2 are inactive. By mirroring their communication properties, the test patterns can be transmitted transparently for other subscribers.

transmitting the test data messages c_1' and c_2'. The messages c_1' and c_2' receive different CAN-IDs than c_1 and c_2, but keep the same timing properties on the bus – size, period, and relative priority – as c_1 and c_2, while remaining distinguishable. As the size of the test patterns is typically large (see Chapter IV), the described technique is exerted on all messages during this test session. In the following, the work at hand concentrates on CAN buses, while the described concept is extensible to other automotive field buses. Therefore, for a BIST test b_r^T, the time q required to transmit the test patterns from b_r^D solely depends on the size s and period p of the functional messages I of ECU r:

$$q(b_r^T) = \frac{s(b_r^D)}{\sum_{c \in I} \frac{s(c)}{p(c)}} \tag{1}$$

C. Integration of Diagnosis in the Design Space Exploration by ILP Encoding

To achieve high-quality optimization able to consider several objectives in parallel, many feasible implementations have to be created and evaluated with respect to design objectives. A *feasible* implementation has to fulfill several constraints, like binding all functional tasks to resources, routing the messages in the system accordingly, etc. For the optimization, this work employs *SAT-decoding*, cf. [17], which is based on the combination of a *Integer Linear Program* (ILP) solver and a *Multi-Objective Evolutionary Algorithm* (MOEA). This technique allows the optimization of the overall system considering several distinct objectives. To utilize this optimization approach, it is necessary to formulate proper constraints of the system as an ILP. In the following, the constraints for diagnostic tasks are presented. These can be combined with the characteristic function Ψ_F as given in [17] which takes into account allocation, binding, and routing for functional tasks being mandatory to be mapped.

For the encoding, a *characteristic function* $\Psi : 2^\Theta \rightarrow \{0,1\}$ is defined that encodes all feasible implementations:

$$\begin{aligned} \Theta \quad = \quad & \{\mathbf{m} \mid \forall m = (t,r) \in M : \mathbf{m} \in \{0,1\}\} \ \cup \\ & \{\mathbf{c}_r \mid \forall c \in C, r \in R : \mathbf{c}_r \in \{0,1\}\} \ \cup \\ & \{\mathbf{c}_{r\tau} \mid \forall c \in C, r \in R, \tau \in \mathbb{T} : \mathbf{c}_{r\tau} \in \{0,1\}\} \end{aligned}$$

Here, the selection of a mapping possibility $m = (t,r) \in M$ is encoded by the truth assignment $\mathbf{m} := 1$, binding task t on resource r. If a communication task c is routed over a resource r it is encoded by $\mathbf{c}_r := 1$ and if this happens at the time step $\tau \in \mathbb{T}$, $\mathbf{c}_{r\tau} := 1$. Using these Boolean variables, a satisfying truth assignment ϑ for the following constraints corresponds to a feasible implementation x in respect to diagnosis tasks:
$\forall d \in D$:

$$\sum_{m=(d,r) \in M} \mathbf{m} \leq 1 \tag{2a}$$

$\forall c \in C, d \in D : (d,c) \in E_T, m = (d,r) \in M$:

$$\mathbf{c}_{r0} - \mathbf{m} = 0 \tag{2b}$$

$\forall c \in C, r \in R, d \in D, t \in T \setminus D$:
$(d,c),(c,t) \in E_T, m_{t,r}, m_{d,r'} \in M$:

$$\mathbf{c}_r - \mathbf{m}_{d,r'} - \mathbf{m}_{t,r} \geq -1 \tag{2c}$$

$\forall c \in C, r \in R$:

$$\sum_{\tau \in \mathbb{T}} \mathbf{c}_{r\tau} \leq 1 \tag{2d}$$

$$\sum_{\tau \in \mathbb{T}} \mathbf{c}_{r\tau} \ - \mathbf{c}_r \geq 0 \tag{2e}$$

$\forall c \in C, r \in R, \tau \in \mathbb{T}$:

$$\mathbf{c}_r - \mathbf{c}_{r\tau} \geq 0 \tag{2f}$$

$\forall c \in C, r \in R, \tau = \{1, \ldots, |\mathbb{T}|\}$:

$$\left(\sum_{\tilde{r} \in R, (\tilde{r},r) \in E_A} \mathbf{c}_{\tilde{r}\tau}\right) - \mathbf{c}_{r(\tau+1)} \geq 0 \tag{2g}$$

$\forall r \in R, m = (d,r) \in M$:

$$-\mathbf{m} + \sum_{m'=(t,r) \in M, t \in T \setminus D} \mathbf{m}' \geq 0 \tag{2h}$$

Eq. (2a) ensures that each diagnosis task is mapped at most once. Note that diagnosis tasks $d \in D, D \subset T$ are only optional, they do not need to be mapped. This allows to allocate ECU resources without any diagnostic capability. Accordingly, Eq. (2b) ensures that a message c is sent if and only if the sending diagnostic task d is bound and that the route of c starts at the resource r where d is mapped to.

The test response of the structural test is gathered by a mandatory task on the gateway. Thus, the communication between optional diagnosis tasks and such mandatory tasks must be enabled. A mandatory task t which receives data from a diagnosis task d may obtain these only if and only if d is selected. Thus, Eq. (2c) states that the communication c arrives at the resource r where t is bound to if d is bound. In accordance with [17], Eq. (2d) prohibits any cycles as in each time step τ only one $\mathbf{c}_{\cdot\tau}$ may be one. Eq. (2e) and Eq. (2f) force c_r variables to one iff one $c_{r\tau}$ is one, and Eq. (2g) ensures that routes follow adjacent resources. To forbid the allocation of a resource solely for purpose of diagnosis, Eq. (2h) finally prevents the activation of a diagnosis task $d \in D$ if no normal task $t \in T \setminus D$ is mapped.

The above constraints hold for all optional tasks. However, typically only one BIST task $b^T \in B \subset D$ should be selected for each component, which is ensured by Eq. (3a). To explore where the test patterns for a BIST task $b^T \in B$ are stored, we model an additional BIST data task $b^D \in D$ which sends the test patterns to b^T while being not necessarily mapped to the same resource as b^T. Thus, both diagnosis tasks are optional, however, if b^T is bound, b^D also has to be bound, as stated by Eq. (3b).
$\forall r \in R$:

$$\sum_{m=(b^T,r) \in M, b^T \in B} \mathbf{m} \leq 1 \tag{3a}$$

$\forall c \in C, b^T \in B, b^D \in D$:
$(b^D,c),(c,b^T) \in E_T, m_{b^D,\cdot}, m_{b^T,\cdot} \in M$:

$$\mathbf{m}_{b^D,\cdot} - \mathbf{m}_{b^T,\cdot} = 0 \tag{3b}$$

In combination with the constraints for functional tasks as given, e. g., in [17], any consistent truth assignment such that the characteristic function Ψ_F evaluates to 1 represents a feasible allocation, binding, and routing with respect to functional as well as diagnostic constraints.

D. Diagnosis-related Design Objectives

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

- The *test quality* to be maximized is defined as the average stuck-at fault coverage achieved for all the ICs in the ECUs of a given implementation. As Eq. (3a) ensures that only one structural test b^T is selected per ECU r, it is sufficient to summarize the test coverage $c(b^T)$ per ECU and divide it by number of selected ECUs:

$$\frac{\sum_{m=(b^T,r)\in M} c(b^T)\cdot \mathbf{m}}{\sum_{r\in R} \bigvee_{m=(.,r)\in M} \mathbf{m}} \tag{4}$$

- 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 inactive, shut-off time has to be kept reasonably short to ensure a fast shut-down of the vehicle after driving. A short shut-off time also represents a necessary condition to apply BIST during partial networking.
  If the patterns of a BIST session are stored locally on the corresponding 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. Therefore, the shut-off time to be minimized is defined as the extra amount of time the system has to stay awake to run BISTs:

$$\max_{b_r^T\in B} \begin{cases} l(b_r^T) & \text{if } (b_r^D, r)\in M \\ l(b_r^T)+q(b_r^D) & \text{if } (b_r^D, r')\in M, r\neq r' \\ 0 & \text{else.} \end{cases} \tag{5}$$

- The *monetary costs* summarize the costs of the allocated hardware resources and the costs of the required memory for both the functional applications and the additional encoded test patterns. If the design space exploration decides to store BIST data locally at an ECU, additional permanent memory needs to be considered. A less costly approach is to store the encoded information at the central gateway, as the same encoded patterns can be used for different ECUs.

IV. Case Study

The case study consists of an automotive E/E architecture 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. The case study evaluates the integration of diagnostic capabilities into the design of E/E architectures. For this purpose BIST applications have been developed for all ECUs. On the one hand, these applications enable the identification of a faulty ECU for system-level diagnosis. On the other hand, the gathered diagnostic information can be used to distinguish faulty ICs and locate faulty structures within an IC. The target CUT in both cases is an automotive microprocessor from Infineon Technologies AG. In the following, the properties of the diagnostic applications are detailed.

A. BIST Profiles

To generate a BIST application, the necessary on-chip infrastructure was inserted into the design. The characteristics of the final CUT are: 371,900 collapsed faults, 100 scan chains with a maximum length of 77, and a test frequency of 40 MHz. For each ECU, 36 test profiles are available, thus, resulting in 36 BIST tasks per ECU. Each test profile exhibits different performance characteristics in terms of fault coverage, test time and costs. Table I shows the attributes of the selectable test profiles. These profiles were generated to achieve maximum fault coverage, as well as 98 % and 95 % fault coverage. The second column shows the number of applied pseudo-random patterns, the third and fourth column show the obtained fault coverage and the duration of the test, respectively. The last column shows the amount of information in bytes that needs to be transferred or stored on-chip in order to generate and evaluate test patterns (encoded deterministic test data and response data). The fail data is transferred to a central memory. This information is fixed for all test profiles and amounts roughly to 638 Bytes. For each of the 15 ECUs, at most one of these BIST profiles is selected.

B. Experimental Results

The multi-objective optimization was performed on an 8-core Intel Core i7 with 16 GB RAM. Evaluating 100,000 implementations took roughly 29 minutes. 176 not Pareto-dominated implementations according to all objectives (monetary costs[1], test quality, and shut-off time) were found. Fig. 5 shows that some of these implementations (marked with ▲) require a shut-off time of more than 20 seconds. However, these are the implementations which have a high fault coverage with only a minor increase in monetary costs, 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. More specifically, our approach is able to find a feasible implementation with 80.7 % test quality for which the additional costs are less than 3.7 % of the cost of a design without structural tests at all. That is to say, in case of a fault, this implementation identifies the faulty ECU with an 80.7 % success rate. As a next step, logic diagnosis of the faulty IC can proceed with the collected information in the fail memory in order to find the responsible faulty location.

Fig. 6 shows that implementations 1, 3, and 7, which have nearly the same test quality, provide a fine tradeoff between shut-off time and costs: In comparison to implementations 1 and 3, implementation 7 requires less memory which, in turn, reduces monetary costs. However, this also leads to a higher shut-off time, as most diagnostic data is stored at the gateway. Implementations 2, 4, 5, and 6 achieve higher test quality. In these implementations the ratio between gateway memory and the total amount of distributed memory is lower than in implementation 7. This is a direct consequence of the

[1] A virtual cost metric is used which estimates the additional costs required for the distributed pattern memory in addition to the overall hardware costs.

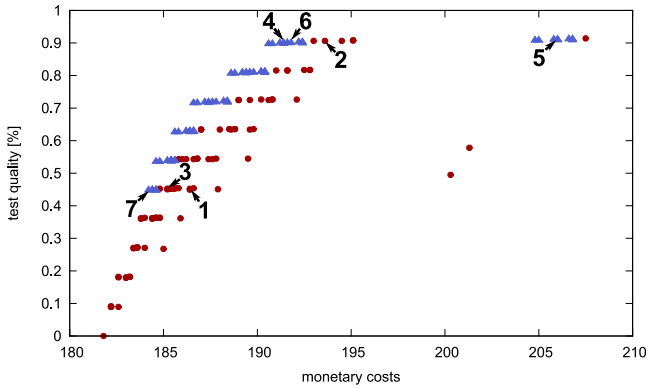

Figure 5. 151 implementations showing the tradeoffs between monetary costs versus test quality. Implementations with shut-off time less than 20 seconds are marked with ●, implementations with higher shut-off time are marked with ▲.

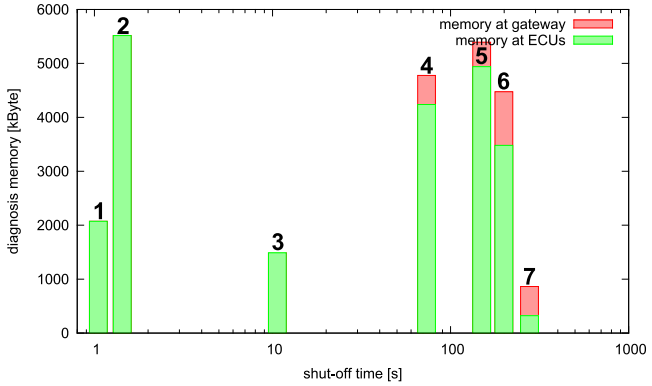

Figure 6. Gateway and distributed memory for diagnosis for the seven implementations marked in Fig. 5. The shut-off time of the implementations is given in log scale.

communication demands of the distributed approach, which cannot be fulfilled in reasonable time for some ECUs. The automatic generation of such fine tradeoffs enable the system designer to choose the best implementation for his goals.

V. Conclusion

The work at hand incorporates BIST into the design of automotive E/E-architectures to improve structural diagnosis in the automotive domain. In contrast to currently used functional tests, the presented approach can accurately identify a faulty ECU for workshop repair and collect sufficient failure information to diagnose hardware faults at ECU and chip-level. The presented methodology assures non-intrusive integration of BIST which does not influence the timing properties of functional applications in the system. To cope with these additional design options, the proposed design space exploration automatically provides high-quality tradeoffs in terms of hardware costs, test quality, and shut-off time. Experimental results for an automotive case study show that the structural diagnostic capabilities of a vehicle can be greatly improved at low additional costs.

References

[1] U. Abelein, H. Lochner, D. Hahn, and S. Straube, "Complexity, quality and robustness – the challenges of tomorrow's automotive electronics," in *Proc. of DATE'12*, 2012, pp. 870–871.

[2] P. Maxwell, I. Hartanto, and L. Bentz, "Comparing functional and structural tests," in *Proc. of ITC'00*, 2000, pp. 400–407.

[3] H. Fang, K. Chakrabarty, Z. Wang, and X. Gu, "Reproduction and detection of board-level functional failure," *IEEE TCAD*, vol. 31, pp. 630–643, 2012.

[4] H.-J. Wunderlich, "BIST for systems-on-a-chip." *INTEGRATION, the VLSI Journal*, vol. 26, no. 1-2, pp. 55–78, 1998.

[5] P. H. Bardell and W. H. McAnney, "Self-Testing of Multichip Logic Modules," in *Proc. of ITC'82*, 1982, pp. 200–204.

[6] S. Pateras and P. McHugh, "BIST: a test & diagnosis methodology for complex, high reliability electronics systems," in *IEEE Autotestcon Proc.*, 1997, pp. 398–402.

[7] T. Vo, Z. Wang, T. Eaton, P. Ghosh *et al.*, "Design for board and system level structural test and diagnosis," in *Proc. of ITC'06*. IEEE, 2006, pp. 1–10.

[8] J. Qian, X. Wang, Q. Yang, F. Zhuang *et al.*, "Logic BIST architecture for system-level test and diagnosis," in *Proc. IEEE Asian Test Symposium (ATS'09)*. IEEE, 2009, pp. 21–26.

[9] A. Cook, S. Hellebrand, and H.-J. Wunderlich, "Built-in Self-Diagnosis Exploiting Strong Windows in Mixed-Mode Test," in *Proc. European Test Symposium (ETS'12)*, Annecy, France, may 2012.

[10] A. Cook, M. Elm, H. Wunderlich, and U. Abelein, "Structural in-field diagnosis for random logic circuits," in *Proc. of ETS'11*, 2011, pp. 111–116.

[11] 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 IOLTS'09*, 2009, pp. 237–242.

[12] Y. Li, S. Makar, and S. Mitra, "CASP: Concurrent Autonomous Chip Self-Test Using Stored Test Patterns," in *Proc. of DATE'08*, 2008, pp. 885–890.

[13] P. Bernardi and M. S. Reorda, "A new architecture to cross-fertilize on-line and manufacturing testing," in *Proc. of ATS'11*, 2011, pp. 142–147.

[14] M. Eberl, M. Glaß, J. Teich, and U. Abelein, "Considering diagnosis functionality during automatic system-level design of automotive networks," in *Proc. of DAC'12*, 2012, pp. 205–213.

[15] R. D. Eldred, "Test Routines Based on Symbolic Logical Statements," *Journal of the ACM (JACM)*, vol. 6, no. 1, pp. 33–37, 1959.

[16] *SAE International. Standard J/1979*, Std.

[17] M. Lukasiewycz, M. Streubühr, M. Glaß, C. Haubelt, and J. Teich, "Combined system synthesis and communication architecture exploration for MPSoCs," in *Proc. of DATE'09*, 2009, pp. 472–477.

Table I. BIST profiles b and their number of random patterns, fault coverage c, runtime l, and the size of the encoded deterministic and response data s.

profile number	number of PRPs	$c(b_i^T)$ [%]	$l(b_i^T)$ [ms]	$s(b_i^D)$ [Bytes]
1	500	99.83	4.87	2,399,185
2	500	99.84	4.87	2,401,554
3	500	98.17	2.81	994,156
4	500	95.73	1.71	455,061
5	1,000	99.84	5.79	2,370,883
6	1,000	99.84	5.74	2,340,080
7	1,000	98.15	3.66	918,895
8	1,000	96.13	2.67	455,193
9	5,000	99.87	13.37	2,300,488
10	5,000	99.87	13.31	2,263,762
11	5,000	98.21	11.23	772,886
12	5,000	95.61	10.25	311,258
13	10,000	99.87	22.93	2,261,705
14	10,000	99.87	22.85	2,210,762
15	10,000	98.06	20.61	834,119
16	10,000	95.97	19.75	304,549
17	20,000	99.88	42.11	2,216,126
18	20,000	99.88	42.05	2,180,585
19	20,000	97.62	39.74	757,737
20	20,000	95.16	38.88	229,353
21	50,000	99.87	99.59	2,054,510
22	50,000	99.87	99.53	2,018,968
23	50,000	97.93	97.24	610,337
24	50,000	96.11	96.63	231,227
25	100,000	99.87	195.84	2,054,081
26	100,000	99.87	195.74	1,994,845
27	100,000	98.10	193.49	611,093
28	100,000	95.36	192.76	158,531
29	200,000	99.89	388.06	1,888,552
30	200,000	99.89	387.99	1,843,533
31	200,000	98.13	385.87	540,342
32	200,000	95.99	385.26	162,417
33	500,000	99.89	965.35	1,767,609
34	500,000	99.89	965.31	1,741,544
35	500,000	98.28	963.25	475,080
36	500,000	96.69	962.76	171,792