Logic Diagnosis Based on Logic Built-In Self-Test Signatures Collected In-Field from Failing System-on-Chips

electronics

MDPI

Article

Logic Diagnosis Based on Logic Built-In Self-Test Signatures Collected In-Field from Failing System-on-Chips †

Paolo Bernardi [1], Gabriele Filipponi [1,*], Giusy Iaria [1,*], Claudia Bertani [2] and Vincenzo Tancorre [2]

[1] Diparimento di Automatica e Informatica (DAUIN), Politecnico di Torino, 10129 Torino, Italy; paolo.bernardi@polito.it
[2] STMicroelectronics, 20864 Agrate Brianza, Italy; claudia.bertani@st.com (C.B.); vincenzo.tancorre@st.com (V.T.)
* Correspondence: gabriele.filipponi@polito.it (G.F.); giusy.iaria@polito.it (G.I.)
† This paper is an extended version of our paper published in Collecting diagnostic information through dichotomic search from Logic BIST of failing in-field automotive SoCs with delay faults. In Proceedings of the 2023 26th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS), Tallinn, Estonia, 3–5 May 2023.

Abstract: Modern embedded nanoelectronic devices, particularly in safety-critical sectors, require high dependability throughout their lifecycle. To address this, designers have started integrating extra circuitry for on-device self-testing, such as the Logic Built-In Self-Test (LBIST). However, while automatic test equipment (ATE) ensures exhaustive testing during manufacturing, in-field testing capabilities are limited. This study introduces a novel methodology for in-field data collection of failure information from LBIST engines and a subsequent logic diagnosis strategy to facilitate failure analysis of field returns. The information is collected from key-on and key-off self-tests, executed by central processing units (CPUs) with a fixed seed and different frequency configurations, primarily addressing transition delay (TRN) faults. The proposed approach capitalizes on the constrained in-field configurability of LBIST and does not require a custom architecture, making it highly practical and readily applicable to real-world devices. The logic diagnosis strategy significantly reduces the number of candidate faults by exploiting the first failing pattern index found during the in-field testing and data collection. Reducing fault candidates could enhance accuracy during failure analysis, especially when field return devices exhibit a "No Trouble Found" (NTF) behavior. The experimental results are reported for ITC'99 benchmarks and an industrial automotive system-on-chip (SoC) produced by STMicroelectronics, with about 20 million gates.

Keywords: functional safety; Logic Built-In Self-Test (LBIST); Silicon Lifecycle Management (SLM); logic diagnosis; Multiple Input Signature Register (MISR); in-field data collection

Citation: Bernardi, P.; Filipponi, G.; Iaria, G.; Bertani, C.; Tancorre, V. Logic Diagnosis Based on Logic Built-In Self-Test Signatures Collected In-Field from Failing System-on-Chips. *Electronics* **2024**, *13*, 4234. https://doi.org/10.3390/electronics13214234

Academic Editor: Costas Psychalinos

Received: 7 October 2024
Revised: 23 October 2024
Accepted: 27 October 2024
Published: 29 October 2024

1. Introduction

Embedded nanoelectronic devices have become increasingly pervasive in our daily lives over the past few decades [1]. The growing number of specifications required to meet current standards and accommodate expanded features has led to a notable surge in the complexity and scale of chip and embedded system manufacturing [2,3]. Particularly in safety-critical market sectors such as the automotive industry, handling system abnormalities and defects becomes a critical concern. Given the intricate nature of the modern system-on-chip (SoC), it is imperative to develop and explore novel approaches that can ensure high dependability throughout the entire lifecycle of these devices.

Quality and reliability are critical aspects of automotive SoCs. In the context of SoC manufacturing testing, methods, processes, and procedures rely heavily on external automated test equipment (ATE) [4], which conducts exhaustive SoC testing by utilizing scan-based techniques [5]. To meet the demands for reliability and adherence to stringent

requirements dictated by quality standards in short time, designers have integrated additional circuitry that is capable of conducting on-device self-tests, formally known as Built-In Self-Test (BIST) [6–8]. When the self-test targets the combinational part of the device, the BIST is called Logic BIST (LBIST) [6,7]. In the standard Self-Testing Using MISR (Multi Input Shift Register) and Parallel SRSG (Shift register sequence generator) (STUMPS) architecture [6–8], a Linear Feedback Shift Registers (LFSR) [6] is the engine for input pattern generation, whereas a signature is produced through an MISR [6] that represents the test outcome. During each LBIST cycle, the input patterns are generated by the LFSR and applied to the combinational elements of the circuit or through the scan chains [9]. The MISR updates the signature, which evolves till the final one can be read after the test. To identify irregularities, the final signature is compared with a known reference called the golden signature.

LBIST engines are configurable with different parameters, such as the initial seed, pattern generation quantity, and operating frequency. Such configurations are fully available during the manufacturing test phases, in which the LBIST is operated from the outside through test access ports [5] using ATE.

Once a device is shipped to its applicative mission, the ATE is no longer available as the device in its final "in-field" context. Therefore, a predominant solution for in-field testing is the use (or reuse after manufacturing) of the BIST [10], which brings testing capabilities directly onto the device itself. Within modern SoCs, there often exists a peripheral component called the Self-Test-Control-Unit (STCU), which exposes BIST's interface registers, through which the firmware run by the central processing units (CPUs) in the SoC can schedule and launch the LBISTs run directly on the chip, without the intervention of external equipment.

Within the automotive field, evaluating the reliability of processors is crucial. For a vehicle, the LBIST is usually launched at the key-on and -off of the vehicle, as mandated by recent standards such as IEC61805 [11] and ISO26262 [12]. Consequently, beyond the traditional presale evaluations conducted during the manufacturing phase [13,14], the health of the device is continually monitored post-deployment. This approach is adopted to mitigate the risk of unforeseen malfunctions due to aging factors [15,16].

Nowadays, due to a large number of field returns (i.e., chips that stopped working properly), retrieving as much diagnostic information as possible from the in-field context is always becoming more important. The configurational flexibility of LBIST engines would greatly help in this direction. However, it is usually significantly reduced when the chip is in its mission environment, permitting only limited configurations. These limitations are primarily motivated by concerns that specific parameters may overstress the components and induce premature degradation in the device under test (DUT), attributable, in part, to the energy demands of particular scan patterns. As such, SoC manufacturers can apply constraints on LBIST reconfigurability when operating in-field, i.e., not allowing the modification of the LFSR seed or the reprogramming and resetting of the MISR functionalities that would otherwise be accessible via test access ports. The proposed approach seeks to capitalize on the restricted "in-field" parameters to extract crucial insights regarding failures throughout the operational lifespan of the device and to exploit them to pinpoint a reduced set of candidate faults in case of field return.

Thus, this study proposes the following:

- A comprehensive methodology for **in-field data collection** of failure information **from constrained LBIST engines** activated along the mission mode of the device [10];
- A **logic diagnosis strategy** to facilitate failure analysis of field returns by **leveraging in-field failure data** collected on the chip.

The paper extends the concepts previously published in [10]. The latter proposed a methodology aiming to identify and store the first failing LBIST pattern's index and its corresponding signature by running the self-test multiple times when a signature mismatch is observed. As an extension, this paper proposes a logic diagnosis strategy, which s the main goal of this work, that exploits such collected information to reduce at most the final

number of candidates. Thus, this paper provides completely new experimental results for the diagnosis of both real failed devices and benchmarks circuits.

The identification of the first failing pattern along the LBIST run is possible with a limited set of LBIST diagnostic features; the constrained LBIST, launched by CPUs on key-on/key-off, should own the ability to set the number of patterns to apply and read out a resulting signature instead than a single GO/NOGO signal. Additionally, the LBIST run should be controllable regarding application frequency to identify and store the highest working frequency at which tests pass for the failing devices. These additional data are crucial for providing TRN faults timing information back to the manufacturers.

As such, the proposed methodology introduces two innovative contributions: firstly, the emphasis on TRN faults to monitor aging problems [15,16], and secondly, the system's applicability to real-world industrial devices, given that no architectural changes are needed and a limited set of configurations is usually left to the final user of the chip.

All the collected data are stored in the device's nonvolatile memory space. Even if cloud-based data sharing [17,18] could be an option, the proposed approach prioritizes data security and privacy, ensuring that the collected data remain protected until the broken device returns to the fab. In-field collected data are helpful, as they could help reduce the incidence of "No Trouble Found" (NTF) scenarios, a new growing issue for companies [19]. If a device returns from the field for some misbehavior and, when tested again, passes all the tests, it is tough to shorten the list of candidates to be the root cause of failure. Indeed, once the device is returned to manufacturers, the collected data can be exploited to perform logic diagnosis. To overcome NTF issues and massive field return devices, the interest in Silicon Lifecycle Management (SLM) [15,20] is growing. Recent studies [21–23] propose ways of generating compacted tests, also with the LBIST, to enhance the final coverage of such tests. Nevertheless, only very recently has a study been published regarding also the diagnosis [24], showing the new relevance that the topic of diagnosis for in-field testing is gaining.

In the state-of-the-art, various methodologies [25–32] have been proposed for diagnosing failures caught by LBIST. In [28], the authors propose strategies that combine the in-field self-tests execution and the diagnosis process, but their applicability relies on custom LBIST architectures. Moreover, more recently, ref. [24] proposed a specific LBIST scheme that enhances the common ATPG diagnosis process by exploiting the on-chip pattern generation method introduced in [23]. There are, then, a few examples [31,32] proposing logic diagnosis strategies that do not require any diagnostic feature and are solely based on the failing LBIST signatures analysis. Similarly, but with higher efficiency, the proposed logic diagnosis methodology introduces a way to diagnose LBIST failures collected in-field, without requiring architectural changes or specific on-chip pattern generation. By exploiting the first failing pattern of the LBIST run, the candidate set can be drastically reduced compared to [31,32]. The comprehensive proposed methodology is, thus, capable of filling the gap between what happens in the operational device lifespan and the off-line diagnosis process, while it often happens that a chip that failed in-field is not failing anymore once back to the failure analysis lab, originating an NFT situation.

The proposed methodology is exemplified through both highly available benchmark circuits [33], an industrial case study produced by STMicroelectronics, which offers practical insights into harnessing the power of LBIST to ensure device dependability. The latter is an automotive SoC, usually adopted for safety-critical car elements such as controlling the airbag behavior. The experimental results show that the average number of fault candidates obtained from the logic diagnosis process is approximately 100. Intriguing results are illustrated for some real failing devices; for some, the proposed logic diagnostic process leads to a diagnostic resolution (DR) as low as two candidate faults. Moreover, the results are also compared to other state-of-the-art methodologies [31,32] addressing the LBIST logic diagnosis only exploiting the failing signatures, and to the logic diagnosis methodology proposed by [23,24], based on a storage-based LBIST scheme. Comparative analysis shows that the proposed approach manages to reach better results for the majority

of the failed devices. In the end, simulation-based results are also reported for completeness, considering how the proposed logic diagnosis strategy would behave with the specific architecture proposed by [28].

The remainder of this article is organized as follows. Section 2 gives the reader an essential technical background and describes related state-of-the-art work. Section 3 describes the proposed methodology, including the in-field data collection system and the strategy for successive logic diagnosis. Section 4 reports the experimental results for a batch of failed automotive devices produced by STMicroelectronics, and gives a comparative analysis with other state-of-the-art methodologies. Finally, Section 5 draws some conclusions.

2. Background

This section provides a brief overview of the foundational aspects on which this work is based and discusses and compares related state-of-the-art work.

2.1. Silicon Lifecycle Management

In safety-critical sectors, the device manufacturing process does not mark the completion of the testing flow. Instead, continuous monitoring of the device throughout its lifecycle is essential to ensure that its reliability does not deteriorate. Consequently, over the past few decades, SLM has emerged as a significant area of focus for industries. SLM encompasses various phases that span the device's entire lifespan, starting from its conception to deployment in end-user systems. The main goal of SLM is to gather and analyze valuable data during the operational lifetime of a device, thereby acquiring knowledge that can be utilized to enhance its performance and reliability [15].

Within the realm of SLM, in-field testing plays a crucial role in maintaining the reliability of SoC devices during their operational mode. Meeting the reliability requirements of diverse application domains has become increasingly critical. Various solutions can be developed and implemented to address this issue. Particularly in safety-critical domains, regulations and standards such as ISO26262 may impose specific fault coverage targets based on varying safety levels. Depending on the application context, in-field testing may be conducted during key-on and key-off phases of the device or when the application is executing [34].

In particular, on-chip LBISTs (described in detail in Section 2.2) are exploited to detect device failures and enable SLM, as discussed in [35,36].

2.2. Built-In Self-Test

Testing has become more sophisticated with the introduction of very-large-scale integration (VLSI) technology. To address this issue during the manufacturing phase, boundary scan techniques were employed as a design for testability (DfT) solution to identify as many defects as possible. This approach utilizes specialized DfT hardware, called scan chains, to stimulate the DUT. Test patterns, or vectors, are applied to the input of these structures. The following responses are then observed and compared to nonfaulty behavior. The activation of these structures is highly complex; therefore, ATEs are tasked with managing this complexity. However, the testing landscape has become exceedingly challenging due to the continual advancements in VLSI technology. Consequently, the development and implementation of tests have struggled to keep pace with the growing complexity of modern devices. This challenge is particularly pronounced in safety-critical industries such as the automotive industry, where stringent standards govern reliability assessments. With the escalation of operating frequencies, pin-outs, and costs, standard ATE becomes impractical. To address these challenges, manufacturers have begun integrating auxiliary modules within devices that are capable of conducting internal self-testing, thereby eliminating dependence on external equipment. This technology, known as BIST, enhances reliability and is frequently employed as a cost-effective alternative to more sophisticated and expensive ATEs. Moreover, BISTs require minimal additional hardware as they reuse

scan chains that may remain unused after manufacturing, allowing circuit testing without disclosing sensitive design information to users. However, device manufacturers must provide instructions for running the test and interpreting the results.

In the BIST methodology, the unit under test (UUT) can be programmed to operate in either the functional or test mode. In the test mode, the primary inputs (PIs) and primary outputs (POs) of the device are disconnected, and a BIST controller feeds test patterns generated by a test-pattern generator (TPG) into the device inputs. The outputs are connected to an output data evaluator (ODE), which checks whether the obtained signature matches the expected signature. The ODE typically uses an MISR to update the signature after each BIST cycle. This signature depends not only on the current test outputs but also on previous test outcomes. Therefore, instead of comparing each output with the expected result, it is sufficient to determine the correct signature and verify its match with that produced by the ODE. The above description refers to the standard STUMPS architecture [6–8], which is shown in Figure 1.

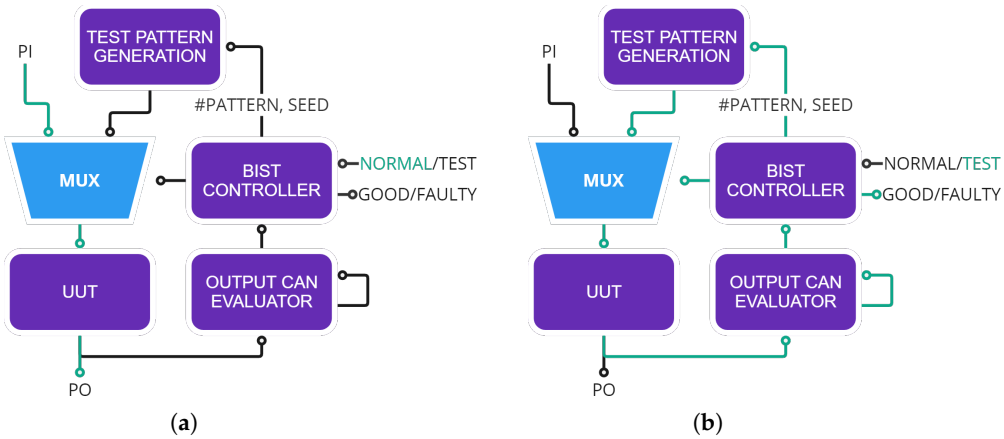

Figure 1. Standard STUMPS BIST architecture composed of its major functional blocks [6–8]. (**a**) Normal functional paths are enabled, PIs reach the UUT. (**b**) Self-test paths are enabled, the BIST controller feeds test vectors to the UUT and captures its response in the ODE.

LBIST refers to the application of BIST to test logic modules. Its implementation usually involves the following modules:

- LBIST Controller: It selects the test or functional mode and handles LBIST configuration;
- Pseudo Random Number Generator (PRNG): It generates patterns for the UUT;
- MISR: It collects the output signature obtained after executing the LBIST. The signature is updated at each LBIST cycle, hence it depends not only on the final test but also on the previous tests.

The LBIST Controller can be programmed with specific testing parameters reported below:

- Pattern-Count-Stop (PCS) indicates the number of patterns that the TPG generates;
- Initial seed of the PRNG for the pattern generation;
- Frequency of application, which can be controlled and configured from high to low, depending on the requirements.

After being programmed, the LBIST starts the self-test by iterating the following:

1. LFSR generates a random number starting from the given seed;
2. Shift values in/out of the scan chain/s and update MISR;
3. if $\#shift < total\_shifts$ go to 1;
4. Apply pattern by pulsing capture clocks cycles;
5. if $\#pattern < PCS$ go to 1;
6. Self-test ends and MISR can be observed and compared to the golden value.

2.3. Logic Diagnosis

In today's semiconductor industry, reducing the yield loss during the manufacturing of integrated circuits is one of the primary concerns [20,37]. Consequently, the investigation of the relationship between defects and faults in digital circuits has become one of the most researched topics.

Logic diagnosis is a critical process in electronic circuit testing and fault analysis. This involves identifying and isolating the underlying faults that cause erroneous behavior or failures in digital logic circuits. The goal of logic diagnosis is to accurately pinpoint the root causes of these faults, enabling efficient and effective repairs or improvements to be made. The process typically begins with the observation of abnormal circuit behavior or detection of failing tests during functional or manufacturing testing. Logic diagnosis then involves systematically analyzing the circuit responses to various input stimuli, often utilizing test patterns specifically designed to target potential faults. By carefully examining the circuit outputs and comparing them with the expected results, logic diagnosis techniques can help identify the faulty components, connections, or design flaws responsible for the observed discrepancies. The insights gained from logic diagnosis aid in enhancing the reliability and performance of digital circuits, supporting efficient debugging, and enabling targeted repairs or optimizations.

The process of logic diagnosis is both computationally difficult and memory-expensive, and generating structures that can store all the necessary information to isolate a fault is a deeply researched subject. These structures, known as *fault dictionaries*, are critical for identifying faults or a set of candidate faults that may cause the observed behavior. A subset of information is selected within the circuit faulty responses, allowing for fault identification. Various decision-making processes have been proposed, leading to different dictionary organizations.

The generation of fault dictionaries has been the subject of considerable research [38,39]. Researchers have explored various methods and techniques to efficiently create these structures. One of the critical factors that impact the design of fault dictionaries is the number of faults that must be considered. The larger the number of faults, the more challenging it becomes to design a fault dictionary that is both memory-efficient and computationally feasible.

Another significant challenge in the generation of fault dictionaries is the trade-off between the accuracy of fault identification and dictionary size. As the accuracy of fault identification increases, the size of the fault dictionary also increases. This increase in size can affect the memory requirements and computational complexity of the fault diagnosis process. Thus, a balance must be maintained between the accuracy of fault identification and dictionary size.

Boppana et al. [40] proposed a tree-like structure to represent faults. This tree structure is appealing because it maintains the fault classification provided by the pattern set. The set of faults that cause the faulty behavior of the circuit can be identified by starting at the root of the tree and traversing down to a leaf. Each leaf corresponds to a set of faults (equivalent class) that produces the same faulty responses for every applied pattern. Figure 2 shows an example of such a structure, assuming three patterns ($T1$, $T2$, and $T3$) and a fault universe composed of letters from A to I. The proposed approach uses a modified version of such a tree-based fault dictionary. Since this work aims to compute logic diagnosis for failures caught by LBIST, a suitable version for LBIST of such representation will be detailed in Section 2.3.

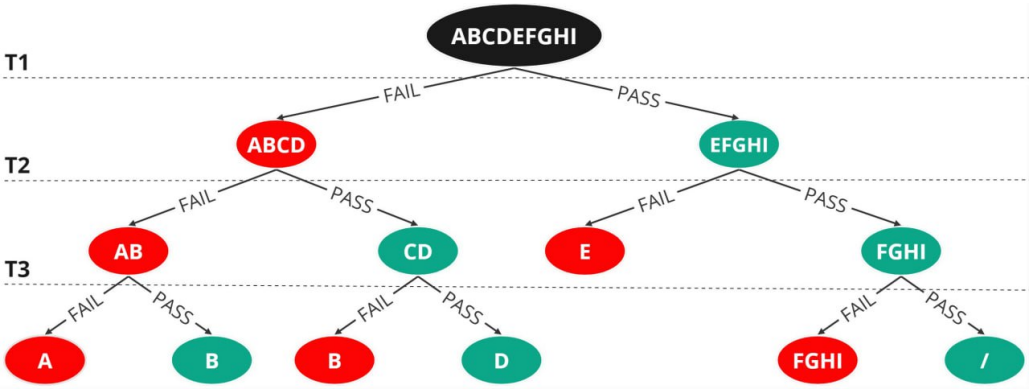

Figure 2. Tree-based fault dictionary with an example of three patterns.

2.4. Related Work

This section reports the current state of the literature for in-field data collection and logic diagnosis methodologies. Moreover, it briefly describes the contribution of this work and its differences to the state-of-the-art.

In-field data collection approaches mainly differentiate upon the collected data, the information the extracted data hold, and the location where data are stored.

A few examples in the literature describe methodologies about mostly in-field monitoring. Tenentes et al. [17] proposed a methodology for Internet of Things (IoT) SoCs to collect diagnostic data over the Internet, storing them in a cloud environment.

Vrachkov et al. [18] presented a diagnostic data collection scheme for the automotive sector which manages to comply with modern safety standards.

Table 1 summarizes the previously discussed approaches:

- The name of the methodology;
- A brief description of the approach;
- The reported case of study, if given;
- The type of collected data;
- The methodology benefits;
- The methodology limitations.

Table 1. In-field data collection methodologies.

Method	Brief Description	Reported Case of Study	Collected Data	Benefits	Limitations
Collective-Aware System-on-Chips for Dependable IoT Applications [17]	Storing data about device health and performance in the cloud to build dynamic models ultimately improving device lifespan	Low-cost IoT SoCs	Diagnostic	Improved life-time estimation	Requiring internet connection
Real-Time Diagnostics in the Automotive Industry over the Internet [18]	Continuously transmitting diagnostic data from SoCs, typically in automotive applications, for remote monitoring and analysis	NA, generalize for Automotive IoT SoCs	Diagnostic	Remote monitoring, data preservation	Requiring internet connection
Proposed approach	Obtaining in-field the first failing LBIST pattern's index to improve logic diagnosis if the device returns back	**Automotive SOC 40 nm ≈ 20M gates**	Failure data	Feedback to manufacturers that can be exploited by logic diagnosis	Memory source limited to the available FLASH memory

The proposed work branches differently from the presented work by firstly asserting in-field device dependability and, in case of failure, a way to collect failure data for later use.

Logic diagnosis based on the LBIST is a strong topic with many presented works in the literature. Bayraktaroglu et al. [25] investigated the use of the superposition principle to reduce the diagnosis time in a scan-based BIST. In addition [26], they introduced a low-cost deterministic partitioning technique and demonstrated that these techniques outperform LFSR-based methods by offering a faster DR. Bayraktaroglu et al. [27] proposed an alternative methodology that extends sets of scan cells to include a set of certainly faulty cells. Initially, all scan cells are considered ambiguous, but they can migrate to the nonfaulty or certainly faulty sets based on specific conditions. Their goal is to reduce the diagnosis time by utilizing the failure information in the signatures.

Elm et al. [28] proposed a novel architecture called Built-In Self-Diagnosis (BISD) that leverages the existing BIST to perform in-field data collection and diagnosis. In particular, after collecting failure signatures in a dedicated memory, the strategy performs diagnosis by providing a ranking over the whole fault universe. The rank of a fault depends upon the bitwise similarity between the collected signatures and the *reference faulty signature* previously computed for that fault.

Jayalakshmi et al. [29] proposed a methodology that manages to cut tester time and memory requirements for logic diagnosis on high-volume manufacturing (HVM). The proposal is to use a 2-pass flow procedure with two different types of patterns. The first type is GO/NOGO set of patterns that highlight the necessity for the second diagnostic set.

Ubar et al. [30] proposed a bisection methodology that takes into account the diagnostic weight for logic diagnosis in BIST environments. The approach resorts to deterministic patterns saved into a dedicated memory. The number of patterns results from a trade-off between required memory and the final diagnostic resolutions.

Cheng et al. [31] proposed a methodology based on the computation of error propagation functions, that maps scan cell failures in MISR. Candidates are found in the intersection of the logic cones of scan cells derived from error functions corresponding to *erroneous* MISR bits. The method does not require any architectural change and it is solely based on the resulting MISR signature. In [32], another methodology based on logic cones of scan cells is proposed. Such a study enhances the methodology described in [31] by further decreasing the number of candidates crossing information of also the logic cones for the scan cell that maps in *correct* MISR bits.

Gopalsamy et al. [24] proposed a logic diagnosis strategy enhanced by the use of tests generated on-chip by following the study in [23]. In [23], the authors proposed a way of constructing deterministic test sets making permutations on the existing test vectors. The proposed logic diagnosis [24] method exploits the construction of such test sets to reduce the total candidate sets, providing a better accuracy in academic benchmarks.

The logic diagnosis phase proposed in [28] is based on the BISD custom architecture, which was explained previously. In particular, after collecting failure signatures in a dedicated memory, the strategy performs diagnosis by providing a ranking over the whole fault universe. The rank of a fault depends upon the bitwise similarity between the collected signatures and the *reference faulty signature* previously computed for that fault.

Table 2 summarizes the previously discussed approaches:

- The name of the methodology;
- A brief description of the approach;
- The reported case of study is given;
- The application study is finally reported.

Previous studies mainly focus on architectural LBIST changes and only consider failing patterns. Moreover, many of them could not provide helpful insight into NTF situations. The proposed approach overcame such issues by proposing a methodology that does not require any architectural change. It considers both successful patterns and the first failing one to enhance the logical diagnosis process. Additionally, it primarily addresses TRN faults, the primary source of aging issues.

Table 2. Logic diagnosis through LBIST methodologies.

Method	Brief Description	Reported Case of Study	Application
Improved fault diagnosis in scan-based BIST via superposition [25] & Deterministic partitioning techniques for fault diagnosis in scan-based BIST [26]	Investigates the superposition principle to reduce diagnosis time and introduces a low-cost deterministic partitioning technique for faster DR	s5378 benchmark from ISCAS'89 [41] ≈ 1K gates	Logic Diagnosis resolution improvement
Diagnosis for Scan-Based BIST: Reaching Deep into the Signatures [27]	Proposes an alternative approach that extended the sets of scan cells to include certainly faulty cells, such can migrate when conditions are meet	s5378 benchmark from ISCAS'89 [41] ≈ 1K gates	Logic Diagnosis time reduction
BISD: Scan-based Built-In self-diagnosis [28]	Proposes a novel architecture, named BISD, that with architectural changes, manages to grade the evidence of faults from saved faulty signatures	NXP industrial circuits ≈ 100–350K gates	In-field Diagnosis
A methodology for LBIST logic diagnosis in high volume manufacturing [29]	Proposes a methodology for fail data collection that optimizes tester time & memory usage while still maintaining a high diagnosis quality	NA	Logic diagnosis, tester time & memory reduction
Fault Diagnosis in Integrated Circuits with BIST [30]	Utilizes an optimized bisection strategy based on diagnostic information inhered from test patterns	ISCAS'85 benchmarks [42] ≈ 100–3K gates	Logic diagnosis
Signature Based Diagnosis for LBIST [31,32]	Derive error propagation function for mapping scan capture failures to MISR. Generate fault list using only erroneous patterns	Industrial Circuits ≈ 1.3, 1.9M gates	Logic Diagnosis
A Storage Based LBIST Scheme for Logic Diagnosis [23,24]	Proposes a way to enhance the diagnosis by adding test sets obtained through the computation of subsets of scan vectors and permutations	ISCAS'89, ITC'99, IWLS'05 benchmarks and logic blocks of the OpenSPARC T1	Logic Diagnosis
Proposed approach	Obtain from in-field the first failing LBIST pattern's index. Through fault simulation obtain candidates list using both succeeding patterns and the first failing	ITC'99 benchmakrs [33] and **Automotive SOC 40 nm ≈ 20M gates**	In-field Data Collection & Logic Diagnosis

3. The Proposed Approach

This paper firstly proposes a methodology for collecting failure data in-field and then leveraging them to conduct logic diagnosis based solely on such collected information obtained from the constrained LBIST execution by CPUs.

The flow of the proposed approach is illustrated in Figure 3. It begins at step (0), where the FAB designs and commercializes a device. When the device is ready to be shipped to customers, the FAB creates the fault dictionaries as proposed by this work. In step (1), on the vehicle's key-on/off events during the regular operation of the device, the LBIST is executed under mission constraints, i.e., the LFSR seed cannot be changed and the MISR cannot be reprogrammed to avoid potential issues to the DUT while in-field. Step (1) results in a signature being reported (2). If this signature matches the expected one, the vehicle can be deemed safe, and the process is repeated. Conversely, a signature mismatch indicates a failure in the logic of the UUT, meaning that the vehicle is unsafe and further investigation is required.

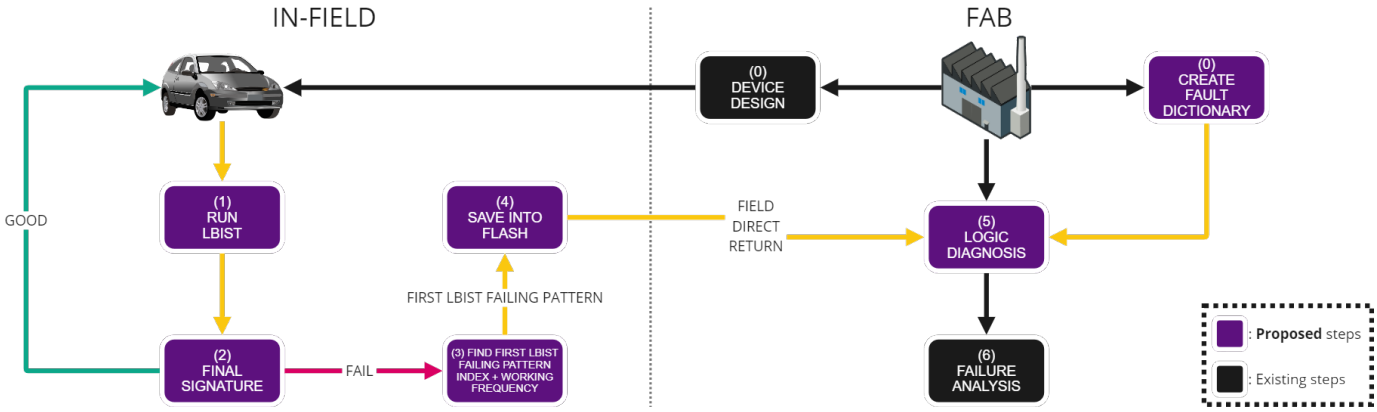

Figure 3. Flow of the proposed approach. Extended figure from [10].

The reported signature in step (2) provides limited information because it does not reveal which pattern's index caused the UUT to fail, thus limiting the scope of logic diagnosis investigations. However, the index of the first failing pattern can be used to add meaningful information to the final signature.

A dichotomic search algorithm (3) is used to retrieve the first failing pattern's index. Moreover, in this step, the working frequency at which the tests pass is extracted. This step requires the ability to control precisely the number of patterns to apply and to read out the final signature.

Once the pattern index is identified, the resulting signature and frequency are stored (4) in the device's flash memory, thus allowing a refined logic diagnosis once the device is returned to the manufacturers (5). Lastly, at (6), a possible failure analysis can be conducted by the FAB.

The faults covered by every pattern can be deduced by means of fault simulation. Given these prerequisites, candidates for a faulty device rely on the difference between those sets. In particular, the first failing pattern should cover faults that were not detected by previous patterns. Thus, these are the candidates for our diagnosis. Because the previous patterns passed successfully, the faults covered before the first failing pattern could be deleted from the set of candidates, leaving the unique coverage of the first failing pattern.

Thus, the proposed approach consists of two phases:

A. **In-field data collection**: Every time the vehicle is turned on/off, LBIST is executed at the maximum allowed frequency to specific target TRN faults. At the test end, if the resulting signature does not match the golden one, the frequency at which LBIST reports a correct signature is computed by decreasing it during LBIST runs. Once computed, the index of the first failed LBIST pattern is identified by means of a dichotomic search algorithm, as described in Algorithm 1, and it is then stored with the corresponding signature.

B. **Logic diagnosis of field return**: If the device returns to manufacturers from in-field, the information stored inside the FLASH memory is exploited to aid in diagnosing possible failures. Fault dictionaries are used to find the failing class of the device, and the information about the index of the first failing pattern makes it possible to obtain a noncorrupted candidate set.

3.1. In-Field Data Collection

This section describes how the in-field data collection works [10]. First, the application of a dichotomic search to collect the first failing LBIST pattern's index and how to compute the golden signature on-the-fly are discussed. Second, the storage required for the in-field approach is reported.

3.1.1. First Failing LBIST Signature Collection

When an LBIST's signature does not match with the expected one, it becomes nontrivial to identify the index of the first failing pattern. The signature of the LBIST is calculated in such a way that it utilizes previous data to generate a new signature, causing the deviation to propagate until the end, when the resulting signature can be read (as shown in Figure 4). Therefore, tests must be conducted for each pattern until the desired pattern is found.

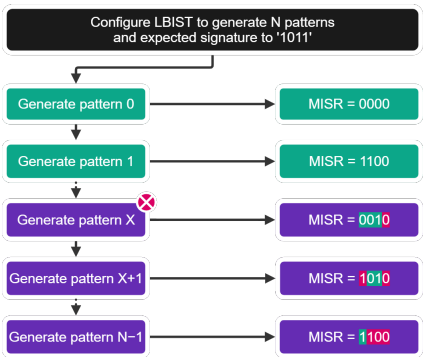

Figure 4. LBIST generates N patterns, the X_{th} fails and propagates the error till self-test end in the MISR.

This study leverages a widely known algorithm in computer science, the dichotomic approach, which has been extensively utilized in various research works [43–45]. Its versatility makes it well suited for numerous applications. The innovation in this work, when compared to others, lies in applying the algorithm to identify the index of the first failing LBIST pattern in a faulty device. This significantly reduces the number of required tests, bringing it down to the logarithmic scale relative to the total number of patterns. The pseudo-code for the dichotomic search algorithm is provided below:

Algorithm 1 Pseudo-code of dichotomic search

Require: k, cardinality of the set
Require: $f : K \rightarrow \{False, True\}$
$low \leftarrow 0$
$high \leftarrow k - 1$
while $high > low$ **do**
    $current \leftarrow (high + low) \gg 1$
    if $f(current) = True$ **then**
        $low \leftarrow current - 1$
    else
        $high \leftarrow current - 1$
    end if
end while

The LBIST signature satisfies all requirements of the dichotomic search algorithm described in Algorithm 1, making it applicable in this context. These requirements include the following:

- The signature relies on past signatures, ensuring that any deviation propagates throughout the entire test;
- The cardinality corresponds to the number of generated patterns, which can be programmed using the LBIST Controller;
- The necessary function, which compares the expected signature with the obtained signature, outputs a boolean value.

The flow, illustrated in Figure 5, is designed based on the step-wise programmability of the LBIST. To target TRN faults more effectively, each LBIST execution is initially performed at the maximum operating frequency.

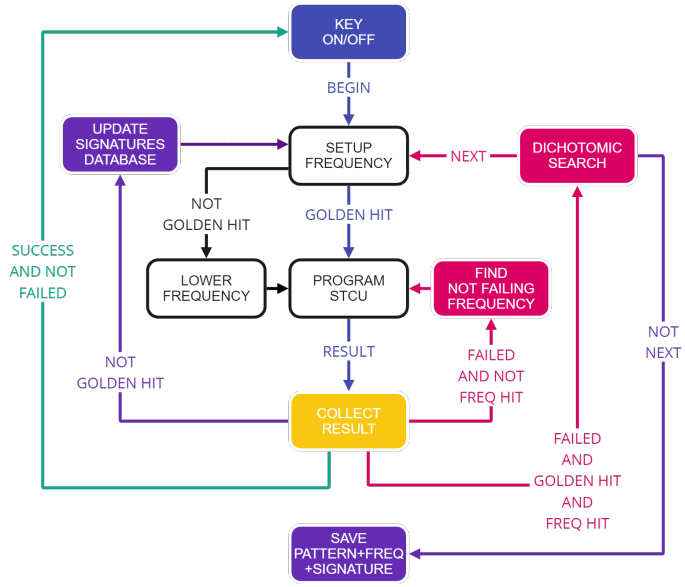

Figure 5. Simplified finite state machine (FSM) version of the logic to be implemented for the proposed work. Extended from [10].

The sequence of steps is as follows:

1. It begins by executing LBISTs from firmware using the maximum number of patterns *#total* at the highest frequency;
2. If the obtained signature matches the expected one, all patterns generated between 0 and *#total* can be classified as successful (marked as green). Consequently, the testing concludes, and the device is deemed to function correctly, as shown in Figure 6.

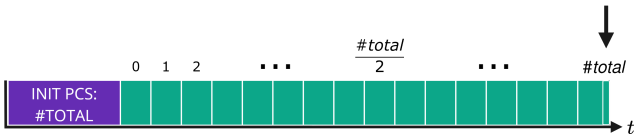

Figure 6. If the signature is good, the test ends immediately. From [10].

3. If there is a signature mismatch, the proposed approach firstly aims to identify at which frequency tests pass by decreasing the working frequency by a given Δ. The Δ parameter is influenced by the architectural limitation of the range of the registers and can be determined by the minimum working frequency steps at which devices produce failures. Subsequently, the dichotomic algorithm is initiated by configuring the LBIST with half of the total patterns $\frac{\#total\ patterns}{2}$. If the signature is still erroneous, all signatures generated using a number of patterns between $\frac{\#total\ patterns}{2}$ and $\#total\ pattern$ patterns can be discarded as a pattern prior, or equal to $\frac{\#total\ patterns}{2}$, producing a wrong signature. Consequently, these signatures can be marked in red without executing the LBIST.
4. This process continues by repeating the logic described in Algorithm 1. It concludes when no further divisions can be made, indicating that only a single number of patterns remains, which is the first failing pattern. This scenario is illustrated in Figure 7.

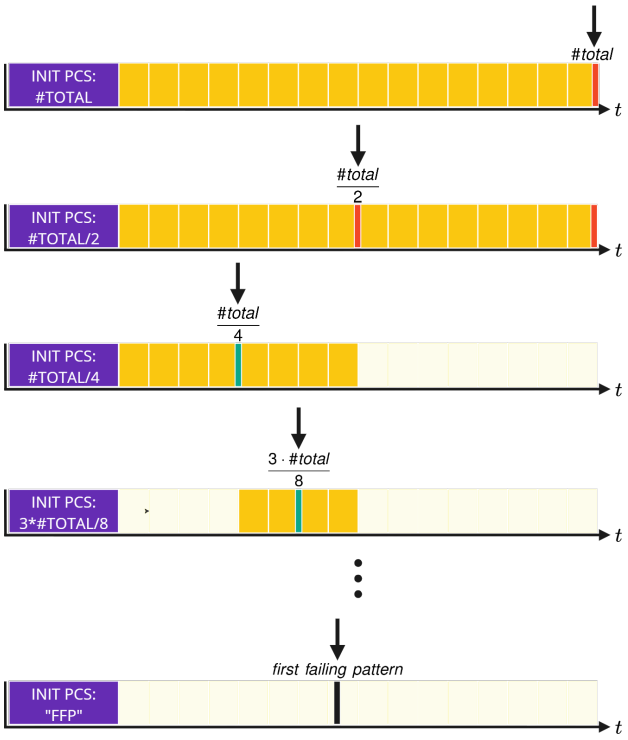

Figure 7. If the signature is wrong, the dichotomic search algorithm is applied, stopping when two different dichotomies cannot be distinguished. From [10].

The index of the first failing pattern, referred to as p_{ff}, obtained by means of a dichotomic search, holds key information. It enables refined logic diagnosis by considering the coverage of prior patterns:

1. Every pattern with indexes prior to p_{ff} $(p_0, \ldots, p_{ff-1})$ must cover a given set of faults;
2. This is true also for p_{ff}.

If p_{ff} is the first failed, it means that the set composed of patterns between $[p_0, p_{ff})$ does not cover the observed failure, but p_{ff} must cover it.

Regarding golden signatures for tests comparison, signatures are retrieved from a database stored in the FLASH memory. However, if a signature is not found in the database, a secure procedure is initiated to extract the necessary signature. The process of obtaining a golden signature and arrangement of the flash memory are described in the following sections.

3.1.2. Golden Signature Computation

Golden signatures are essential in LBIST for anomaly detection during the comparison process, as explained in Section 2.2. The tests, depicted in Figure 5, are conducted at a high frequency to effectively identify TRN faults. However, running tests at such frequencies increases the likelihood of the final signature being compromised. If a golden signature is not available in FLASH, it is still possible to store it securely by the following steps:

- Suspend current test;
- Lower the frequency to a safe LBIST execution range;
- Collect golden signature and save it in FLASH at self-test end;
- Restart the initial suspended test.

The role of the frequency switch is to minimize the chances of capturing any potential failures in the signature, which could otherwise lead to corruption in the test result and subsequent comparison.

This study primarily focuses on TRN faults, but it is possible to apply a dichotomic search to LBIST's signature if stuck-at (SA) faults need to be addressed. However, the exist-

ing method of capturing golden signatures from an in-field device must be replaced with a prefilled signature database because an SA fault corrupts the signature regardless of the working frequency.

However, it is important to emphasize that based on the bath curve [46], which describes the occurrence of faults in devices during manufacturing and after prolonged usage, and considering burn-in (BI) [14,47] to detect production failures, it is reasonable to assume that a failure shall initially manifest as a TRN fault before transforming into an SA fault. Nevertheless, the main objective of this study is to identify anomalies as soon as they arise, making TRN coverage a primary focus.

3.1.3. Data Storage

The layout of the nonvolatile memory required by the approach, illustrated in Figure 8, is structured into three distinct regions:

1. The *frequency region*, where the working frequency at which tests pass is stored;
2. The *golden region*, responsible for storing the golden signatures;
3. The *failure region*, where information about the first failing pattern obtained during the dichotomic search is stored.

The second and last regions share the same structure, composed of

- The index representing the number of patterns for the test;
- The resulting LBIST signature.

Depending on the fault model being considered, the required footprint changes.

For TRN fault modeling at device shipping, the structure is initially empty and becomes populated during device operation. In the *golden region*, only the first entry is occupied with the good signature, because no anomalies have been detected at that point. However, when LBIST detects a misbehavior, the *frequency region* will hold the safe working frequency, at which the tests pass. All entries in the *golden region* are filled using the dichotomic search algorithm (Algorithm 1), and the entry in the *failure region* will contain information about the first failing pattern. The number of entries in each region is determined by the number of times the total number of patterns can be divided into, given by the formula $\#entries = log_2(\#total\ patterns) + 1$.

For SA modeling, it is necessary to prefill the *golden region* because no computation involving golden signatures takes place. Consequently, the number of entries in the region is determined as $\#entries = \#total\ patterns + 1$.

Additionally, the *frequency region* and the indexes of the golden signatures can be omitted as the frequency is not involved and the indexes can be derived as an offset from a base address in FLASH.

Irrespective of the fault model under consideration, it is essential to replicate this scheme for each LBIST partition.

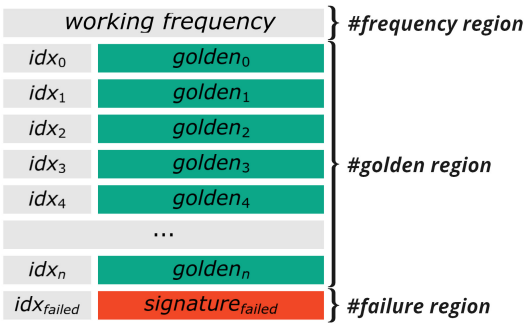

Figure 8. Memory layout for the proposed approach. From [10].

3.2. Logic Diagnosis of Field Return

The number of field return devices is a growing problem for manufacturing companies. Identifying the source of failure has become more complex considering the hugeness of modern SoCs in safety-critical sectors.

In this scenario, the proposed in-field data collection system facilitates the logic diagnosis process for manufacturers by identifying and storing the index and the resulting signature of the first failing pattern, in case of failure during key-on and key-off tests. Indeed, the way the LBIST works makes all the information resulting from patterns that fail after the first one corrupted, because MISR signatures are calculated from previous states. Thus, all signatures subsequent to the first failing one would be failures in any case; moreover, they do not add information to the diagnosis process. On the contrary, they could corrupt information for diagnosis (see Figure 9).

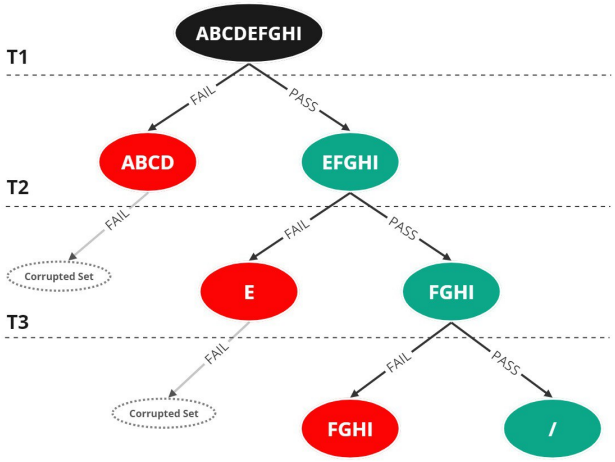

Figure 9. Tree-based fault dictionary for LBIST with an example of three patterns.

For this reason, as Figure 9 shows, the fault dictionary computed and used to diagnose the failures collected through the proposed methodology is incomplete because, for each applied pattern, we stop at only the first two *pass*/*fail* nodes and continue investigating only the *pass* node case. Thus, each *failed* node is actually a leaf because, from that state, it is no longer possible to accurately identify a fail or pass because of what was said above about how MISR works.

The steps of the process for diagnosing failures are shown in Figure 10, in which the diagnosis is composed of two phases:

- **Fault dictionary calculation**: The tree-based fault dictionary is calculated using fault simulations to extrapolate the unique coverages of each pattern generated by the LFSR, and the information is stored to be used in case of returns.
- **Logic diagnosis of in-field return**: Once some device is returned to manufacturers for defects, the fault dictionary is exploited to find the candidate faults using the information stored inside the FLASH memory of the device thanks to the in-field data collection system. Thus, by knowing which is the first failing pattern's index, the corresponding set of candidates can be extracted from the tree-based fault dictionary previously computed.

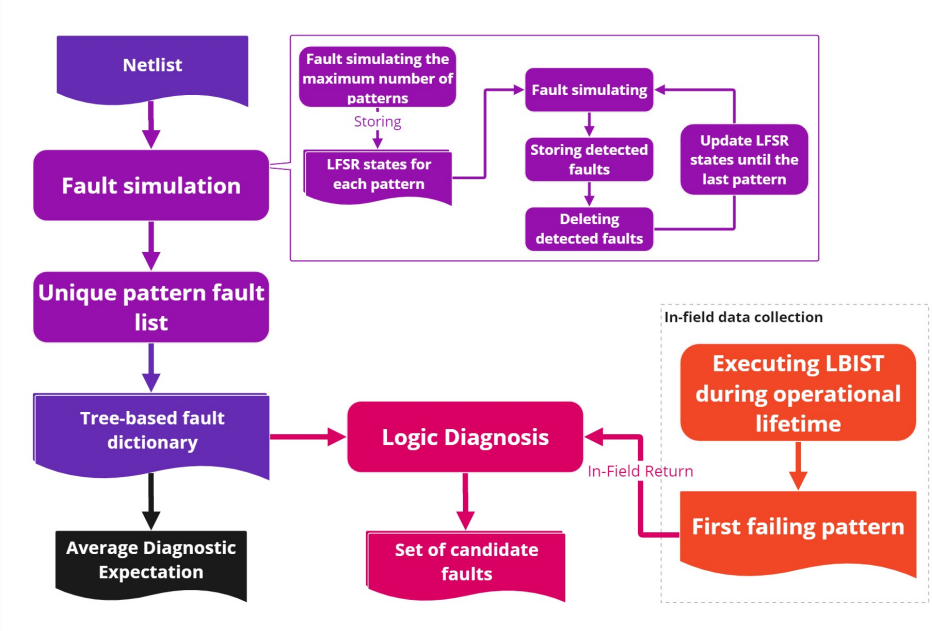

Figure 10. General overview of the logic diagnosis of field return devices.

To calculate the fault dictionary for the LBIST, a fault simulation campaign must be carried out a priori to extrapolate the unique coverage of each pattern, which can be generated by LFSR and applied during the in-field data collection of the device.

The steps of the algorithm that can be used to extract the unique coverage can be summarized as follows:

1. Execute a fault simulation with the maximum number of applicable patterns by also tracing the state of the LFSR for each applied pattern;
2. The information about the states of the LFSR is used to apply the different patterns generated on silicon;
3. For each pattern:
   (a) Set the LFSR with the corresponding seed;
   (b) Fault simulate over the fault list;
   (c) Update the fault list by removing the detected faults;
   (d) Store the information to build the tree-based fault dictionary.

Thus, it is possible to obtain all extractable information from the faults collected through in-field data collection. Changing the seed of the LFSR makes it possible to fault simulate each pattern independently, thereby extracting its unique coverage. Once the unique coverages have been extracted, the tree-based fault dictionary for the LBIST can be computed following the algorithm detailed in Algorithm 2. Indeed, the proposed approach uses the tree methodology [40,48] to create the fault dictionary containing all possible patterns that the LBIST can generate. Figure 9 shows a possible implementation for a case in which the maximum number of patterns to be applied via the LBIST is three ($T1$, $T2$, and $T3$) and the fault universe (FU) is a set defined as $FU = \{A, B, C, D, E, F, G, H, I\}$, where each letter represents a possible fault. At each applied pattern, only two new nodes are created, contrary to what happens in non-LBIST scenarios, as described in Section 2.3 and shown in Figure 2. In more words, considering a tree-based fault dictionary for LBIST, the *pass* node is forked for each pattern applied to two *fail*/*pass* nodes. Meanwhile, the *fail* node does not undergo forking; instead, it represents all the equivalence classes within the fault dictionary. The motivation behind this is that information extrapolated from the LBIST makes it impossible to pinpoint which other test patterns may have also failed after the first fail.

This limitation arises because, upon a failure, the MISR becomes compromised due to the initial failure it encounters. Consequently, it does not retain untainted information

about subsequent patterns that may have also failed. As a result, the data obtained in real-world conditions concerning the very first failing pattern hold significant importance. This information is crucial in identifying candidate faults using the precomputed fault dictionary.

Specifically, it is the only index that can provide valuable insights necessary for effective diagnosis in such cases.

Algorithm 2 Tree-based fault dictionary for LBIST

Require: $fault\_universe$, set containing all the faults.
Require: $unique\_coverages$, list of sets containing the unique covered faults for each pattern.
1: $prev \leftarrow fault\_universe$
2: $fault\_dictionary \leftarrow [\varnothing]$
3: $i \leftarrow 0$
4: $N \leftarrow$ length($unique\_coverages$)
5: **while** $i < N$ **do** ▷ for each pattern
6: $pass\_node \leftarrow prev - unique\_coverages[i]$
7: $fail\_node \leftarrow prev \bigcap unique\_coverages[i]$
8: $fault\_dictionary[i].pass \leftarrow pass\_node$
9: $fault\_dictionary[i].fail \leftarrow fail\_node$
10: $prev \leftarrow pass\_node$
11: $i \leftarrow i + 1$
12: **end while**

Considering the tree-based fault dictionary shown in Figure 9, if a device returns back to manufacturers with the information that the first failing pattern was the third one ($T3$), then the set of candidate faults is the *fail* node corresponding to that pattern, that is, $\{F, G, H, I\}$. Thus, after having computed the fault dictionary, the straightforward way to terminate logic diagnosis is simply accessing the *fail* node at the index of the first failing pattern, as shown in Algorithm 3.

Algorithm 3 Logic diagnosis exploiting both tree-based fault dictionary and in-field collected information

Require: $fault\_dictionary$, tree-based fault dictionary
Require: p_{ff}, index of the first failing pattern
1: $candidate\_faults \leftarrow fault\_dictionary[p_{ff}].fail$
2: **return** $candidate\_faults$

4. Experimental Results

This section presents the experimental results of the methodology applied to simulated circuits from academic benchmarks ITC'99 [33] and an industrial case of study. Moreover, the experimental setup developed to test the proposed approach simulating the in-field scenario is described together with the diagnostic expectation (DE) values obtained by testing the entire flow on faulty devices returned during their lifetime. Additionally, the results obtained from real faulty devices with the proposed methodology are compared with the methodologies [31,32] described in Section 2. Such studies were chosen for comparative analysis because, like the proposed one, they rely solely on LBIST signatures, without any architectural change, to reprove the cause of failure. Thus, they have been also adopted to the real failed industrial devices to analyze the resulting DR. In addition to this, the proposed methodology is also compared, through simulation, to [24], which exploits a storage-based LBIST scheme, for the benchmarks circuits and for the simulated computation of the DE of the industrial case study. Finally, simulation-based results are given for how the methodology presented in [28] would behave when making an architectural change in the referenced industrial case study. Moreover, at the end of the section, a high-level comparison between the considered state-of-the-art methodologies is reported, highlighting

the main differences and the possible scenarios in which the proposed approach is the best suited.

4.1. Experimental Setup

The case studies used to validate the methodology are circuits from ITC'99 [33] and an industrial automotive SoC produced by STMicroelectronics. The industrial DUT has the following specifications:

- About 20 million gates;
- About 700 thousand flip-flops;
- Multicore architecture;
- ASIL-D compliant;
- 7 LBIST partitions;
- 92 MBIST partitions.

To verify the proposed methodology for circuits from ITC'99 [33] benchmarks, we used the same technology file of the industrial case study and performed the LBIST insertion through commercial DfT tool. Table 3 summarizes the configuration for the scan chains and the resulting total number of faults for the considered circuits.

Table 3. Faults summary for considered benchmarks circuits.

Circuit	Number of Scan Cells	Number of Scan Chains	Number of SA + TRN Faults
b15	525	22	66K
b17	1455	32	180K
b18	3129	40	470K
b20	486	22	102K
b22	701	32	146K

Moreover, to further prove the effectiveness of the proposal using the industrial case study, we developed an experimental setup composed of the following:

- The physical board containing the SoC;
- An external tool responsible for managing LBIST parameters and facilitating serial communication with the board.

During normal device operation, the firmware programs and executes LBISTs. The primary component of the external software tool is responsible for coordinating the LBIST experiments, implementing a dichotomic search for the proposed method, and managing communication with the board and power supply.

Environmental factors like temperature and operational conditions can significantly impact the collected diagnostic information, hence the final diagnosis. To overcome this, experiments were first characterized on a nonfaulty chip over multiple hours of nonstop BIST execution at different parameters. This characterization allows the identification of a set of parameters where tests should not fail and ones that should be avoided because operations of stress/temperature (DUT becoming hot due to BIST execution) would cause the nonfaulty device to output the wrong signature.

Table 4 reports the 7 LBIST partitions with the corresponding number of scan cells and scan chains, and faults summary, including SA and TRN faults.

Table 4. Faults summary for LBIST partitions of the industrial case of the study.

LBIST Partition	Number of Scan Cells	Number of Scan Chains	Number of SA + TRN Faults
0	28K	800	4M
1	82K	1K	15M
2	56K	800	6M
3	61K	1K	5M
4	72K	1K	5M
5	69K	1K	6M
6	68K	800	9M

4.1.1. FLASH Footprint

Each of the 7 LBIST partitions on board provides 16 bits of programmable PCS, resulting in an initial programming of LBISTs with a starting PCS of $0xFFFF$. This value is then halved or adjusted through a dichotomic search in each iteration, as described in Algorithm 1.

In the scheme described in Section 3.1.3, the entry in the *frequency region* is stored as a 4 bytes float. Each entry of the *golden and failure* regions has a width of 80 bits: 16 bits for the index and 64 bits for the signature. The footprint of FLASH memory varies depending on the fault model used. For TRN modeling, the entire structure occupies $32 + 7 \times 80 \times ((\log_2(0xFFFF) + 1)) = 9552$ bits or approximately 9 Kb. On the other hand, for the SA model, because the *frequency region* and indexes for the *golden region* can be omitted, the structure size increases to $7 \times 64 \times (0xFFFF + 1) + 16 = 29{,}360{,}144$ bits, or approximately 29 Mb. To reduce the memory footprint of SA fault modeling, a "skip" step technique can be employed for rows in the golden signature table. However, if a signature mismatch occurs, this approach results in a trade-off with potentially lower DR. Nevertheless if golden signatures are saved strategically, for example, following the fault coverage curve, the loss in resolution can be minimized.

It is important to note that the methodology described in this study is highly effective in collecting information regarding TRN fault detection, which is the primary objective.

4.1.2. Signature Collection Time

In the case of the study, when LBISTs are programmed to concurrently test the maximum number of patterns $0xFFFF$, it takes approximately 100 ms to complete the self-test procedure. Upon completion, the device initiates a functional reset that resets the cores but keeps the STCU active for signature collection. Therefore, during device power-on/off, if the device is not faulty, only one execution of LBISTs is necessary, resulting in a procedure duration of 100 ms.

In the event of an initial signature mismatch, the working frequency is initially calculated. This procedure involves the execution of LBIST, with the maximum number of patterns, starting from the maximum frequency and decreasing it by a given Δ until the tests pass. Considering a starting frequency of 200 MHz and Δ = 10 MHz, the best case is that the working frequency is $max_f - \Delta$; as such, only one iteration is required, resulting in an execution time of 100 ms. However, reaching Δ is the worst-case, resulting in an execution time of $100 \times \frac{200}{\Delta} = 2000$ ms.

At this point, a dichotomic search, described in Algorithm 1, is initiated, which involves executing LBISTs a total of 16 times with a decreasing or increasing number of patterns. In the worst-case scenario, where the dichotomic search ends with $0xFFFF - 1$ patterns, the total execution time required is approximately $100 + \sum_{i=1}^{15} \frac{100 \cdot (2^i - 1)}{2^i} \sim 1500$ ms. This estimation accounts for the fact that the upper dichotomy is always selected. Conversely, in the best-case scenario, where the pattern's index to be found is 1, indicating that the lower dichotomy is consistently chosen, the required time is approximately $\sum_{i=0}^{15} \frac{100}{2^i} \sim 200$ ms. It

is worth noting that if golden signatures are not present in the database, the execution time will double because a golden signature needs to be extracted for each execution.

The presented measurements focus solely on the LBIST execution time and do not include the time taken for the FLASH erase and program operations. However, even if these operations are performed in every iteration, their respective durations of 0.0192 ms and 0.00475 ms are considered negligible compared to the LBIST execution time.

4.1.3. Collected Data from Faulty Devices

The data collection of LBISTs signatures in the experimental setup was performed at the core reference voltage with rising frequencies to find a better range to target TRN faults. The number of frequencies used depends upon a parameter named Δ, which indicates the quantity by which the frequency is decreased each time, equal to 10 MHz. In the proposed case of study, the minimum allowed and legal, with regards to registers, frequency ramp up/down is 2 MHz. The first step of the proposed approach is to detect at which frequency the DUT does not fail. With experimental observations in our case of study, we determined that it was faster to initially decrease by an initial 10 MHz until the DUT does not fail and then start a dichotomic search to fine-tune the computed frequency based on the 2 MHz increase/decrease. Figure 11 shows the results for 23 faulty devices.

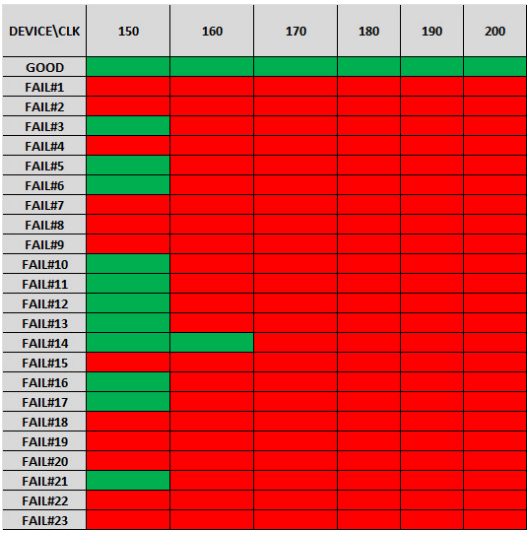

Figure 11. Failures information for the industrial case study for field return devices depending on the BIST execution frequency. Green cells indicate no failure being observed, meanwhile a red cell indicates a fail.

4.2. Logic Diagnosis

In this section, the experimental results of the logic diagnosis process are reported, including the calculation of the average DE for each LBIST partition and the DR for a batch of faulty field return devices.

4.2.1. Diagnostic Expectation

Following the construction of the tree-based fault dictionary outlined in Section 3.2, the average DE is computed.

Table 5 shows the results of average DE of the proposed logic diagnosis methodology, in comparison with other state-of-the-art studies, for benchmarks circuits from ITC'99 [33].

The method proposed in [32] gives the best outcome in the majority of the benchmarks circuits considered. Nevertheless, the proposed methodology achieves good and similar results to [32].

Table 5. Average DE for ITC'99 [33] benchmarks.

Circuit	Avg. DE	Avg. DE [24]	Avg. DE [31]	Avg. DE [32]
b15	1.77	7.64	6.12	1.57
b17	1.87	7.55	3.47	1.11
b18	5.43	10.54	24.96	2.13
b20	1.37	14.43	7.74	1.21
b22	2.16	16.84	18.51	2.34

Table 6 presents the average DE for each partition of the LBIST. The reported values represent the average number of candidate faults once a misbehavior is captured. Using a tree-based fault dictionary, the average DE is calculated as the total number of candidate faults inside leaf nodes divided by the total number of patterns applied.

As previously explained in Section 3.2, constructing the fault dictionary for the LBIST involves excluding the failure branch of a previously *failed* node, thus reducing diagnostic information to only the first failing pattern.

In the worst-case scenario, in which the first failing pattern coincides with one of the initial patterns or occurs shortly thereafter, the average DE tends to be high.

For instance, LBIST partition 5, in the industrial case study, on average, manifests approximately 109 candidate faults. This value, although relatively high, remains reasonable when considered in conjunction with the overall fault list size for the partition, which is approximately 6 million.

However, focusing on LBIST partition 5, as the failing pattern is discovered beyond the 64th position, the average DE experiences a notable decrease. In particular, it is approximately 37.

The relationship between the position of the first failing pattern and the resulting DE is depicted in Figure 12, which shows that DE is markedly influenced by the position of the first failing pattern.

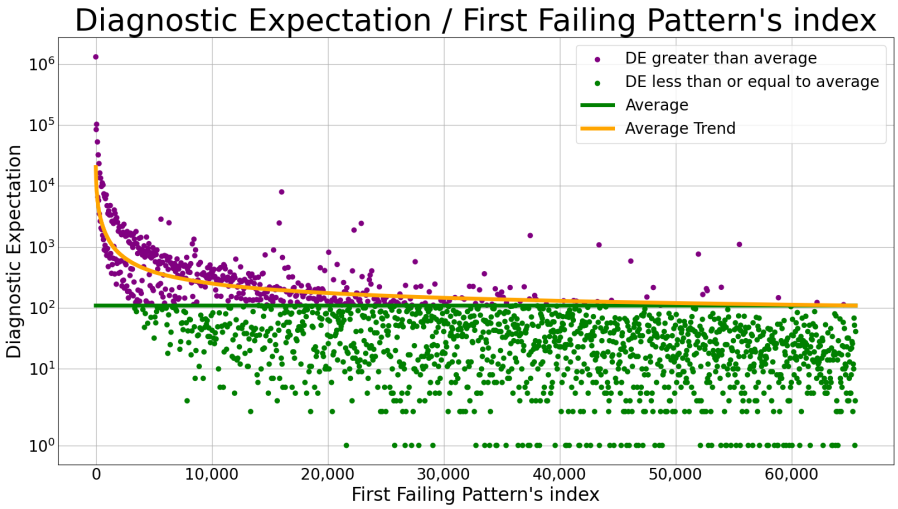

Figure 12. Depiction of how the DE changes, in the industrial case study, based on the position of the first failing pattern for LBIST partition 5.

Indeed, Figure 12 shows the trend of the DR depending on the first failing pattern. The y-axis, on a logarithmic scale, represents the DE; the x-axis represents the index of the first failing pattern. The green horizontal line represents the average constant value of DE, which is approximately 109, whereas the orange line depicts the average trend of the DE. The logarithmic scale on the y-axis provides a more nuanced view of this relationship over a wide range of values.

All points with a DE greater than the average value are colored purple. Instead, those with a DE less than the average are colored in green.

In summary, this graph illustrates that as the index of the first failing pattern increases (moving right along the x-axis), there is a decrease in the DE (moving down along the y-axis). This decrease in DE corresponds to an increased diagnostic accuracy due to fewer possible candidates.

Table 6 provides a summary of the average DE for the industrial case study for each LBIST partition compared with that obtained using the [24,31,32] methodologies. The average DE offers critical insights into the diagnostic efficiency of different partitions. For example, in LBIST partition 1, which encompasses a comprehensive fault list of 15 million faults, the proposed approach reaches an average DE of 207, indicating a relatively high DR. By contrast, LBIST partition 0, with a fault list size of 4 million, demonstrates an average DE of 92, suggesting a more efficient diagnostic process.

Table 6. Average DE for each LBIST partition of the industrial case study.

LBIST Partition	Avg. DE	Avg. DE [24]	Avg. DE [31]	Avg. DE [32]
0	92	44	12,362	7305
1	207	81	38,253	29,846
2	103	52	18,872	10,751
3	100	65	23,114	17,539
4	101	58	22,020	12,878
5	109	56	25,595	17,231
6	123	77	26,412	19,945

The storage-based LBIST scheme proposed in [24] was also used for comparison for the industrial case study. The experimental procedure described in the study was reproduced to compute the results, that is, the software procedure for diagnostic patterns generation and then the logic diagnosis methodology on 2000 random distinct faults injection campaign.

The approach described in [31] exploits the error propagation function of MISR to determine candidates for faulty behavior and the logic cones of the scan cells capturing the wrong values.

To calculate the values for the DE outcome using the [31] approach, the same method described in the cited paper was used. That is, 2000 faults were randomly injected into the combinational logic for each LBIST partition of the case study. The resulting faulty signatures were diagnosed using the previously calculated error propagation function of the MISR.

A similar methodology was used for making the comparison with [32]. The latter is an improved version of what was presented in the first work [31]. Therefore, 2000 faults were injected randomly and exploited the propagation errors of the MISR; the candidates were calculated starting from the cones of the logic of the identified scan cells. The difference with the first method is that in this improved version, some pruning is performed based on the correct bits of the MISR. Some candidates are removed from the initial candidates, which would have propagated their error even to MISR bits that, in the end, turned out to be correct.

Table 6 reports the average number of candidates obtained using the proposed methodology and those described in [31,32] for each LBIST partition. The latter two approaches do not reach low DE values for the industrial case study. The reasoning for [31] is because it is based solely on the MISR of the failing pattern after a single LBIST run, thus without pruning the possible candidates of successfully applied patterns, as proposed in this work. The enhanced methodology [32] makes pruning for the correct MISR bits but still gives a high number of candidates on average. The reason is that the case study, and typically every large complex SoC, has many scan chains, multiple LBIST partitions, and a compacted MISR; therefore, considerable error information is lost during compaction. Thus, simply

analyzing the propagation of errors is not sufficient to accurately diagnose failures for such very large and complex SoCs. Moreover, suppose the case study has multiple scan chains with different lengths. In that case, the total number of shifts will be greater than the total length for some scan chains, and this behavior would result in a more complex error propagation function, as we have in the reported case study.

This difference is significant for [31,32] for the motivations explained above. If we consider LBIST partition 5, we observe an impressive difference in the outcome. This is because partition 5 has approximately 6 million faults in total, 1000 scan chains (see Table 4), and only 64 bits of MISR. Meanwhile, considering the DE value for LBIST partition 0, the difference between the methodologies is still high, but both [31,32] work better for this partition than the other partitions because it has fewer faults and fewer scan chains, but still a 64 bit MISR, resulting in a smaller test compaction.

The results from [24] appear to be the best for the industrial case study. Its effectiveness is reasonable, considering that the patterns used are deterministic and diagnostic, not pseudo-random, and that the information exploited for the diagnosis is more detailed. Indeed, the proposed approach only considers the first failing pattern because, as already explained in the previous sections, the LBIST in-field has typical constraints and the proposal aims at capitalizing what is possible to retrieve. Meanwhile, the [24] aims at diagnosing failed device considering all the failing patterns, thus implying the possibility to retest the device to extract the needed information for the full diagnostic report.

In Section 4.4, a final comparative summary explaining the different advantages of the considered methodologies is reported.

4.2.2. Logic Diagnosis of Faulty Devices

The experimental setup was validated by using a set of failed devices retrieved from the field (as shown in Figure 11). Table 7 presents the total number of candidates per failed device, obtained by exploiting the index of the first failing pattern to perform the diagnosis, which is this work proposal. Indeed, the latter is utilized to identify the specific node within the tree-based fault dictionary, thereby permitting the determination of the set of candidate faults.

Table 7. Logic DR for the industrial case study, obtained through the data collected in-field for a batch of faulty devices field return.

Faulty Chip # *	DR	DR [31]	DR [32]
FAIL#2	2	40,116	16,950
FAIL#5	25,299	4105	3280
FAIL#14	198	15,990	11,925
FAIL#15	41	11,764	9530
FAIL#18	1910	91,140	71,378
FAIL#19	1910	7948	5347
FAIL#20	4158	5844	4670
FAIL#23	586	13,553	8132

* Faulty chip numbers refer to Figure 11.

The total number of failed devices is twenty-three (as shown in Figure 11). Out of these, eight are conclusively diagnosed with combinational logic failures (as shown in Table 7), each associated with a specific set of candidate faults. In addition, our methodology identified fifteen devices that exhibited chain failures. Indeed, the LBIST can be programmed to perform a certain number of patterns as scan chain integrity tests (without capture) before applying actual logic tests. Because our methodology finds the index of the first failing pattern, we can determine which devices manifested scan chain failures unrelated to combinational logic.

The logic diagnosis results are presented in Table 7, where each row represents a faulty chip, indicating the DR achieved, that, here, is the number of candidate faults. The DR provides crucial insights into the effectiveness of the diagnostic process for each faulty chip. For instance, FAIL#2 displayes a DR of 2, suggesting a high level of diagnostic precision, whereas FAIL#5 exhibits a high number of candidate faults, considering only the information that can be retrieved from the vehicle.

Moreover, in Table 7, the last two columns indicate the DR obtained using the methodologies described in [31,32]. Our proposed method has a significant advantage for most analyzed devices. The second method enhances, on average, the DR obtained by the first one, but it still does not reach the proposed method's values. The only case in which our method lacks accuracy compared with the other two is FAIL#5. Indeed, this device reported a failure in the 16th pattern. As depicted in Figure 12, the position of the first failing pattern (in this case, the 16th) determines the value of the DR. The lower the position, the higher the value of the DR. In addition, ref. [32] quite reaches the DR of the proposed approach for the FAIL#20, improving the one achieved by [31]. Nevertheless, considering the average (see Table 6), our method still achieves higher accuracy.

4.3. Using the Architecture Scheme by [28]

Many works in the state-of-the-art, as seen in Table 2, propose changes or extensions to the standard STUMPS BIST architecture to provide a better and more efficient diagnosis process. A comparison with this work would not be appropriate, as the latter's purpose is to provide a way for manufacturers to adapt the proposed methodology to any architecture scheme without requiring a specific one.

However, a simulation based on an architectural change can be adapted to assess its effectiveness in our industrial case study. We choose the BISD architecture [28] because it has a similar goal to ours, as described in Section 2.4.

Firstly, we assessed the requirements of the selected study regarding needed memory:

- Failing memory depth of 50;
- Test set size of 65,535, hence pattern indexes are on 16 bits;
- Single-Input-Signature-Register (SISR) length of $log_2(\#shifts)$ bits.

Considering all LBIST partitions in Table 4, the required memory is 3,146,528 bits or approximately 3 Mb. The reported memory is lower than the one reported in Section 4.1.1 for the stuck-at fault model; however, the target fault model for this work is the transition delay, which takes just a fraction of the calculated one. Moreover, such memory needs to be accounted for the area overhead, as the considered approach does not use already available memories but, instead, creates a new one local to the BISD.

Regarding the logic diagnosis phase, the approach [28] begins with the computation through simulation of faulty signatures for each fault in the DUT, which raises scalability concerns depending on the size of the fault universe. Once such a dictionary is constructed, faulty signatures, if any, can be obtained by executing the BIDS in-field. The outcome of the diagnosis process, described in the referred article, is a final ranking of all the faults ordered so that if a fault ranks first, that is, indeed, the candidate. The authors in [28] present experimental results based on total 800 fault injections. They compute the diagnostic resolution by considering the number of times the first fault in the rank was the injected one.

For our industrial case study, we conducted an injection campaign of about 1000 faults for each LBIST partition in the DUT, and in 23% of the cases, the first fault in the rank was, indeed, the injected one. However in the remaining cases, where the top-ranked fault is not the injected one, the number of candidates yielded by the [28] is the maximum, with the set of candidate faults being the whole LBIST partition's faults.

Moreover, the proposed approach circumscribes the set of candidates in any case, unlike the [28], which provides a ranking of all the faults in the circuit. Therefore, in the case of real faulty devices, the calculated ranking can be followed during failure analysis, but the number of possible candidates is not reduced.

4.4. High-Level Summary Comparisons

Table 8 presents a comparative analysis of the analyzed state-of-the-art logic diagnosis methodologies and the proposed one. It highlights the different data required from the faulty device, the precompilation steps needed, the information provided in the case of NTF devices, and the outcome.

Table 8. High-level summary among different LBIST diagnostic methodologies.

Method	Required Data of the Failed Device	Precomputation	Insights in Case of NTF	Limitations	Outcome
BISD: Scan-based Built-In self-diagnosis [28]	Signatures	Mapping of each fault to its signature through simulation	Yes	It does not provide a final list of candidates but only a ranking	Ranking of the fault universe
A Storage Based LBIST Scheme for Logic Diagnosis [24]	Diagnostic report with all the failed patterns	Software procedure to determine the patterns	No	Diagnostic features are required for the used LBIST scheme and fully deterministic patterns precomputed are necessary	List of candidates
Signature based diagnosis for logic BIST [31]	Signatures	Logic cones of the scan cells linked to failing MISR bits	Yes	The accuracy strictly depends on the aliasing of the MISR	List of candidates
Improving the performance of signature based diagnosis for Logic BIST [32]	Signatures	Logic cones of all the scan cells	Yes	The accuracy strictly depends on the aliasing of the MISR	List of candidates
Proposed	Index of the first failing LBIST pattern	Tree-based fault dictionary	Yes	If the first failing pattern is at the beginning of the test set, the candidate set size could be large	List of candidates

Experimental results showed that the proposal is more efficient for highly complex real-world devices than [28,31,32].

The first [28] requires more precomputation than the others because it is based on the complete faulty signatures set, retrievable only through repetitive fault injections of the fault universe. Moreover, the outcome of the method is the ranking of the fault universe and not a list of candidates. Thus, the set of possible causes of the failure is not actually reduced. The other two [31,32] have good results for benchmark circuits but fail to reach accurate diagnosis in very complex case studies, as shown in Section 4.

Nonetheless, ref. [24] proved to be more efficient in the industrial case study, but it does not give any insights in the case of NTF, as Table 8 shows. Indeed, ref. [24] proposes a way to generate deterministic patterns with the aim of diagnosis and considers all the failing patterns in the diagnosis process, leading then to a more accurate diagnosis. However, using all the failing patterns to retrieve the full diagnostic report is not always possible, especially with NTF devices.

The selection of the most suitable method is crucial. It should align with specific requirements such as the availability of data, the ability to retest the failed device in debug mode, computational resources, and the desired level of detail. In this light, the proposed

method excels by leveraging minimal information from the device, particularly the index of the first failing pattern, making it highly compatible with any LBIST scheme.

Remarkably, it yields a list of candidate faults even if the device does not fail again after being returned to the manufacturers. In instances where the complete diagnostic report cannot be retrieved and it is not possible to change the LBIST architecture of the device, this methodology emerges as the most suitable and efficient option compared to others.

5. Conclusions and Future Work

In safety-critical fields, it is crucial to guarantee high-quality standards during the entire operational lifetime of devices. Therefore, the testing phase must be extended to the in-field life of the devices to ensure that safety does not decrease with time. Moreover, companies have been having many in-field returns, and analyzing failures without having information about what happened during the life of the referred device is complex. Companies especially have issues regarding the NTF situation in which field return devices do not manifest failures once returned to the manufacturer.

Therefore, the proposed methodology addresses this problem. It is based on two phases: data collection during the in-field operational life of devices, and the exploitation of such data to diagnose failures in the case of in-field return to manufacturers. The approach exploits the LBIST programmed by firmware to store the corrupted MISR signatures that manufacturers can use to diagnose potential failures.

For some failed devices, it may be necessary to use more information than just the first failing pattern stored in-field during mission mode, especially if the failure has been caught at the beginning of the test set. Thus, for future work, we are considering adding other steps to the off-line diagnosis process: exploiting the collected in-field information and decreasing the number of candidates by using scan-based diagnostic patterns targeting the candidates found during the device's mission mode.

Author Contributions: All authors contributed to the conceptualization and methodology; software and validation, G.F. and G.I.; formal analysis, G.F. and G.I.; investigation, G.F. and G.I.; resources, C.B. and V.T.; data curation, G.F. and G.I.; writing—original draft preparation, P.B., G.F. and G.I.; writing—review and editing, P.B., C.B. and V.T.; visualization, P.B., G.F. and G.I.; supervision, P.B., C.B. and V.T.; project administration, P.B., C.B. and V.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Ministero dell'Università e della Ricerca grant number DM1061.

Data Availability Statement: All the results discussed in this paper come from real case studies derived from both production and analysis lots. Unfortunately, STMicroelectronics cannot share full data about those experiments, only the data already shown in the paper. Nevertheless, readers can directly contact the authors to seek more details.

Conflicts of Interest: Authors Claudia Bertani and Vincenzo Tancorre are employed by the company STMicroelectronics. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as potential conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CPU	Central processing unit
SoC	System-on-chip
ATE	Automatic test equipment
LBIST	Logic Built-In Self-Test
SLM	Silicon Lifecycle Management
MISR	Multi Input Shift Register
LFSR	Linear Feedback Shift Register
NTF	No trouble found

SRSG	Shift register sequence generator
STUMPS	Self-Testing Using MISR and Parallel SRSG
STCU	Self-Test-Control-Unit
DUT	Device under test
VLSI	Very-large-scale integration
DfT	Design for testability
UUT	Unit under test
PI	Primary input
PO	Primary output
TGP	Test pattern generation
ODE	Output data evaluator
PRNG	Pseudo random number generator
PCS	Pattern-Count-Stop
IoT	Internet of Things
FSM	Finite State Machine
SA	Stuck-At
TRN	Transition delay
DE	Diagnostic expectation
DR	Diagnostic resolution
BI	Burn-In
BISD	Built-In Self-Diagnosis
HVM	High-volume manufacturing

References

1. Srivastava, R.; Mudgil, N.; Gupta, G.; Mondal, H. SoC Time to Market Improvement through Device Driver Reuse: An Industrial Experience. In Proceedings of the 2012 International Symposium on Electronic System Design (ISED), Kolkata, India, 19–22 December 2012; pp. 56–61. [CrossRef]
2. Vassighi, A.; Semenov, O.; Sachdev, M.; Keshavarzi, A.; Hawkins, C. CMOS IC technology scaling and its impact on burn-in. *IEEE Trans. Device Mater. Reliab.* **2004**, *4*, 208–221. [CrossRef]
3. Agrawal, M.; Chakrabarty, K. Test-Cost Modeling and Optimal Test-Flow Selection of 3-D-Stacked ICs. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2015**, *34*, 1523–1536. [CrossRef]
4. Carey, D.R. Introduction to Automated Test Systems—Back to Basics. In Proceedings of the 2019 IEEE AUTOTESTCON, Harbor, MD, USA, 26–29 August 2019; pp. 1–7. [CrossRef]
5. *IEEE 1149.1-2013*; IEEE Standard for Test Access Port and Boundary-Scan Architecture—Redline. IEEE Computer Society: Washington, DC, USA, 2013; pp. 1–899.
6. Bardell, P.H.; McAnney, W.H.; Savir, J. *Built-In Test for VLSI: Pseudorandom Techniques*; Wiley-Interscience: New York, NY, USA, 1987.
7. Abramovici, M.; Breuer, M.A.; Friedman, A.D. *Digital Systems Testing and Testable Design*; Computer Science Press: New York, NY, USA, 1990.
8. Bardell, P.H.; McAnney, W.H. Self-Testing of Multichip Logic Modules. In Proceedings of the International Test Conference, Philadelphia, PA, USA, 15–18 November 1982.
9. Huang, Y.; Guo, R.; Cheng, W.T.; Li, J.C.M. Survey of Scan Chain Diagnosis. *IEEE Des. Test Comput.* **2008**, *25*, 240–248. [CrossRef]
10. Bernardi, P.; Filipponi, G.; Reorda, M.S.; Appello, D.; Bertani, C.; Tancorre, V. Collecting diagnostic information through dichotomic search from Logic BIST of failing in-field automotive SoCs with delay faults. In Proceedings of the 2023 26th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS), Tallinn, Serbia, 3–5 May 2023; pp. 21–26. [CrossRef]
11. *IEC 61508*; Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems. International Electrotechnical Commission: Geneva, Switzerland, 2010.
12. *ISO 26262-1:2018*; Road Vehicles—Functional Safety—Part 1: Vocabulary. International Organization for Standardization: Geneva, Switzerland, 2018.
13. Polian, I.; Anders, J.; Becker, S.; Bernardi, P.; Chakrabarty, K.; ElHamawy, N.; Sauer, M.; Singh, A.; Reorda, M.S.; Wagner, S. Exploring the Mysteries of System-Level Test. In Proceedings of the 2020 IEEE 29th Asian Test Symposium (ATS), Penang, Malaysia, 23–26 November 2020; pp. 1–6. [CrossRef]
14. He, C.; Yu, Y. Wafer Level Stress: Enabling Zero Defect Quality for Automotive Microcontrollers without Package Burn-In. In Proceedings of the IEEE International Test Conference (ITC), Washington, DC, USA, 1–6 November 2020.
15. Kashyap, R. Silicon lifecycle management (SLM) with in-chip monitoring. In Proceedings of the 2021 IEEE International Reliability Physics Symposium (IRPS), Monterey, CA, USA, 21–25 March 2021; pp. 1–4. [CrossRef]
16. Jang, B.; Lee, J.K.; Choi, M.; Kim, K.K. On-chip aging prediction circuit in nanometer digital circuits. In Proceedings of the 2014 International SoC Design Conference (ISOCC), Jeju, Republic of Korea, 3–6 November 2014; pp. 68–69. [CrossRef]

17. Tenentes, V.; Rossi, D.; Al-Hashimi, B.M. Collective-Aware System-on-Chips for Dependable IoT Applications. In Proceedings of the 2018 IEEE 24th International Symposium on On-Line Testing and Robust System Design (IOLTS), Platja d'Aro, Spain, 2–4 July 2018; pp. 57–60. [CrossRef]
18. Vrachkov, D.G.; Todorov, D.G. Real Time Diagnostics in the Automotive Industry over the Internet. In Proceedings of the 2018 IX National Conference with International Participation (ELECTRONICA), Sofia, Bulgaria, 17–18 May 2018; pp. 1–3. [CrossRef]
19. Tran, T.; Gundala, S.R.; Soni, K.; Baker, A.; Fogle, A.; Chandrashekhar, S. No Trouble Found (NTF) Customer Return Analysis. In Proceedings of the 2020 IEEE International Reliability Physics Symposium (IRPS), Dallas, TX, USA, 28 April–30 May 2020; pp. 1–6. [CrossRef]
20. Rajski, J.; Chickermane, V.; Côté, J.F.; Eggersglüß, S.; Mukherjee, N.; Tyszer, J. The Future of Design for Test and Silicon Lifecycle Management. *IEEE Des. Test* **2024**, *41*, 35–49. [CrossRef]
21. Shenoy, J.; Ockunzzi, K.; Singh, V. A Novel Test Data Compaction Method with Improved Debug Capabilities of the Signatures. In Proceedings of the 2023 IEEE International Test Conference India (ITC India), Bangalore, India, 23–25 July 2023; pp. 1–6. [CrossRef]
22. Kaczmarek, B.; Mrugalski, G.; Mukherjee, N.; Pogiel, A.; Rajski, J.; Rybak, L.; Tyszer, J. LBIST for Automotive ICs with Enhanced Test Generation. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2022**, *41*, 2290–2300. [CrossRef]
23. Gopalsamy, S.; Pomeranz, I. Fully Deterministic Storage Based Logic Built-In Self-Test. In Proceedings of the 2023 IEEE 41st VLSI Test Symposium (VTS), San Diego, CA, USA, 24–26 April 2023; pp. 1–7. [CrossRef]
24. Gopalsamy, S.; Pomeranz, I. A Storage Based LBIST Scheme for Logic Diagnosis. In Proceedings of the 2024 IEEE 42nd VLSI Test Symposium (VTS), Tempe, AZ, USA, 22–24 April 2024; pp. 1–7. [CrossRef]
25. Bayraktaroglu, I.; Orailoğlu, A. Improved Fault Diagnosis in Scan-Based BIST via Superposition. In Proceedings of the 37th Annual Design Automation Conference, Los Angeles, CA, USA, 5–9 June 2000; pp. 55–58. [CrossRef]
26. Bayraktaroglu, I.; Orailoglu, A. Deterministic partitioning techniques for fault diagnosis in scan-based BIST. In Proceedings of the International Test Conference 2000 (IEEE Cat. No.00CH37159), Atlantic City, NJ, USA, 3–5 October 2000; pp. 273–282. [CrossRef]
27. Bayraktaroglu, I.; Orailoglu, A. Diagnosis for scan-based BIST: Reaching deep into the signatures. In Proceedings of the Design, Automation and Test in Europe. Conference and Exhibition 2001, Munich, Germany, 13–16 March 2001; pp. 102–109. [CrossRef]
28. Elm, M.; Wunderlich, H.J. BISD: Scan-based Built-In self-diagnosis. In Proceedings of the 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010), Dresden, Germany, 8–10 March 2010; pp. 1243–1248. [CrossRef]
29. Jayalakshmi, A.; Cheong, T.E. A methodology for LBIST logic diagnosis in high volume manufacturing. In Proceedings of the 2012 4th Asia Symposium on Quality Electronic Design (ASQED), Penang, Malaysia, 10–11 July 2012; pp. 249–253. [CrossRef]
30. Ubar, R.; Kostin, S.; Raik, J.; Evartson, T.; Lensen, H. Fault Diagnosis in Integrated Circuits with BIST. In Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007), Lubeck, Germany, 29–31 August 2007; pp. 604–610. [CrossRef]
31. Cheng, W.T.; Sharma, M.; Rinderknecht, T.; Lai, L.; Hill, C. Signature based diagnosis for logic BIST. In Proceedings of the 2007 IEEE International Test Conference, Santa Clara, CA, USA, 21–26 October 2007; pp. 1–9. [CrossRef]
32. Sharma, M.; Cheng, W.T.; Rinderknecht, T.H. Improving the Performance of Signature Based Diagnosis for Logic BIST. PCT/US2009/034933, 23 February 2009.
33. Corno, F.; Reorda, M.; Squillero, G. RT-level ITC'99 benchmarks and first ATPG results. *IEEE Des. Test Comput.* **2000**, *17*, 44–53. [CrossRef]
34. Abraham, J.A.; Gu, X.; MacLaurin, T.; Rajski, J.; Ryan, P.G.; Gizopoulos, D.; Reorda, M.S. Special session 8B—Panel: In-field testing of SoC devices: Which solutions by which players? In Proceedings of the 2014 IEEE 32nd VLSI Test Symposium (VTS), Napa, CA, USA, 13–17 April 2014; pp. 1–2. [CrossRef]
35. Manzone, A.; Bernardi, P.; Grosso, M.; Rebaudengo, M.; Sanchez, E.; Reorda, M. Integrating BIST techniques for on-line SoC testing. In Proceedings of the 11th IEEE International On-Line Testing Symposium, French Riviera, France, 6–8 July 2005; pp. 235–240. [CrossRef]
36. Filipponi, G.; Iaria, G.; Reorda, M.S.; Appello, D.; Garozzo, G.; Tancorre, V. In-field Data Collection System through Logic BIST for large Automotive Systems-on-Chip. In Proceedings of the 2022 IEEE International Test Conference (ITC), Anaheim, CA, USA, 23–30 September 2022; pp. 646–649. [CrossRef]
37. Casarsa, M.; Harutyunyan, G.; Zorian, Y. Test and Diagnosis Solution for Functional Safety. In Proceedings of the 2020 IEEE International Test Conference (ITC), Washington, DC, USA, 1–6 November 2020; pp. 1–5. [CrossRef]
38. Pomeranz, I.; Reddy, S.M. A Same/Different Fault Dictionary: An Extended Pass/Fail Fault Dictionary with Improved Diagnostic Resolution. In Proceedings of the 2008 Design, Automation and Test in Europe, Munich, Germany, 10–14 March 2008; pp. 1474–1479. [CrossRef]
39. Bernardi, P.; Grosso, M.; Reorda, M.S. An adaptive tester architecture for volume diagnosis. In Proceedings of the 2010 15th IEEE European Test Symposium, Prague, Czech Republic, 24–28 May 2010; pp. 227–232. [CrossRef]
40. Boppana, V.; Hartanto, I.; Fuchs, W. Full fault dictionary storage based on labeled tree encoding. In Proceedings of the 14th VLSI Test Symposium, Princeton, NJ, USA, 28 April–1 May 1996; pp. 174–179. [CrossRef]
41. Brglez, F.; Bryan, D.; Kozminski, K. Combinational profiles of sequential benchmark circuits. In Proceedings of the 1989 IEEE International Symposium on Circuits and Systems (ISCAS), Portland, OR, USA, 8–11 May 1989; Volume 3, pp. 1929–1934. [CrossRef]

42. Brglez, F.; Fujiwara, H. A neutral netlist of 10 combinatorial benchmark circuits and a target translator in FORTRAN. In Proceedings of the Symposium on Circuits and Systems, Special Session on ATPG and Fault Simulation, Kyoto, Japan, 5–7 June 1985.
43. Kong, B.Y.; Yoo, H.; Lee, Y. An Automated Synthesis Framework for Fast Evaluation of Maximum Operating Frequency. In Proceedings of the 2023 International Conference on Electronics, Information, and Communication (ICEIC), Singapore, 5–8 February 2023; pp. 1–4. [CrossRef]
44. Zhang, X.; Li, H.; Jiang, L.; Xu, Q. A Low-Cost TSV Test and Diagnosis Scheme Based on Binary Search Method. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2015**, *23*, 2639–2647. [CrossRef]
45. Bernardi, P.; Guerriero, A.M.; Insinga, G.; Paganini, G.; Carnevale, G.; Coppetta, M.; Mischo, W.; Ullmann, R. Built-In Self-Test Architecture Enabling Diagnosis for Massive Embedded Memory Banks in Large SoCs. *Electronics* **2024**, *13*, 303. [CrossRef]
46. Klutke, G.; Kiessler, P.; Wortman, M. A critical look at the bathtub curve. *IEEE Trans. Reliab.* **2003**, *52*, 125–129. [CrossRef]
47. Angione, F.; Appello, D.; Bernardi, P.; Calabrese, A.; Quer, S.; Reorda, M.S.; Tancorre, V.; Ugioli, R. A Toolchain to Quantify Burn-In Stress Effectiveness on Large Automotive System-on-Chips. *IEEE Access* **2023**, *11*, 105655–105676. [CrossRef]
48. Bernardi, P.; Grosso, M.; Rebaudengo, M.; Sonza Reorda, M. A pattern ordering algorithm for reducing the size of fault dictionaries. In Proceedings of the 24th IEEE VLSI Test Symposium, Berkeley, CA, USA, 30 April–4 May 2006. [CrossRef]